# Oracle® Customer Interaction History

Implementation Guide

Release 11*i*

**Part No. B10643-02**

August 2004

ORACLE®

Oracle Customer Interaction History Implementation Guide, Release 11*i*

Part No. B10643-02



Primary Authors: Stephanie Smith, Michael Phelan.

Contributors: Michael Atkinson, Richard Day.

# Contents

## 6 Using Oracle Customer Interaction History

## D Data Validations



## E Seeded Data

# Send Us Your Comments

**Oracle Customer Interaction History Implementation Guide, Release 11*i***

**Part No. B10643-02**

Oracle welcomes your comments and suggestions on the quality and usefulness of this document. 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?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, 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 Applications Documentation Manager
- Postal service:
  Oracle Corporation
  Oracle Applications Documentation Manager
  500 Oracle Parkway
  Redwood Shores, CA 94065
  USA

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

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 Customer Interaction History Implementation Guide.

This guide assumes you have a working knowledge of the following:

- The principles and customary practices of your business area.
- Oracle Customer Interaction History

  If you have never used Oracle Customer Interaction History, Oracle suggests you attend one or more of the Oracle Customer Interaction History training classes available through Oracle University.
- The Oracle Applications graphical user interface.

  To learn more about the Oracle Applications graphical user interface, read the *Oracle Applications User's Guide*.

See Other Information Sources for more information about Oracle Applications product information.

## How To Use This Guide

This document contains the information you need to understand and use Oracle Customer Interaction History.

Chapter 1 describes the Interaction Center product family. It also describes what is new in the current release.

Chapter 2 provides a detailed description of Oracle Customer Interaction History.

Chapter 3 describes the system requirements and product dependencies for Oracle Customer Interaction History. It also describes how to access the administrative console for Oracle Customer Interaction History.

Chapter 4 provides a high-level overview of the implementation task sequence and references to procedures that are performed during the course of an implementation.

Chapter 5 describes task-based procedures for administering Oracle Customer Interaction History.

Chapter 6 describes task-based procedures for using Oracle Customer Interaction History.

Appendix A describes the concurrent programs for Oracle Customer Interaction History.

## 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. 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 additional information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/

**Accessibility of Code Examples in Documentation** JAWS, a Windows screen reader, 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, JAWS 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 Corporation does not own or control. Oracle Corporation neither evaluates nor makes any representations regarding the accessibility of these Web sites.

# Other Information Sources

You can choose from many sources of information, including online documentation, training, and support services, to increase your knowledge and understanding of Oracle Customer Interaction History.

If this guide refers you to other Oracle Applications documentation, use only the Release 11*i* versions of those guides.

## Online Documentation

All Oracle Applications documentation is available online (HTML or PDF). Online help patches are available on MetaLink.

## Related Documentation

Oracle Customer Interaction History shares business and setup information with other Oracle Applications products. Therefore, you may want to refer to other product documentation when you set up and use Oracle Customer Interaction History.

You can read the documents online by choosing Library from the expandable menu on your HTML help window, by reading from the Oracle Applications Document Library CD included in your media pack, or by using a Web browser with a URL that your system administrator provides.

If you require printed guides, you can purchase them from the Oracle Store at http://oraclestore.oracle.com.

## Documents Related to All Products

### Oracle Applications User's Guide

This guide explains how to enter data, query, run reports, and navigate using the graphical user interface (GUI) available with this release. This guide also includes information on setting user profiles, as well as running and reviewing reports and concurrent processes.

You can access this guide online by choosing "Getting Started with Oracle Applications" from any Oracle Applications help file.

## Installation and System Administration

### Oracle Applications Concepts

This guide provides an introduction to the concepts, features, technology stack, architecture, and terminology for Oracle Applications Release 11*i*. It provides a useful first book to read before an installation of Oracle Applications. This guide also introduces the concepts behind Applications-wide features such as Oracle Business Intelligence System (BIS), languages and character sets, and Self-Service Web Applications.

### Installing Oracle Applications

This guide provides instructions for managing the installation of Oracle Applications products. In Release 11*i*, much of the installation process is handled using Oracle Rapid Install, which minimizes the time to install Oracle Applications and the technology stack by automating many of the required steps. This guide contains instructions for using Oracle Rapid Install and lists the tasks you need to perform to finish your installation. You should use this guide in conjunction with individual product user's guides and implementation guides.

### Upgrading Oracle Applications

Refer to this guide if you are upgrading your Oracle Applications Release 10.7 or Release 11.0 products to Release 11*i*. This guide describes the upgrade process and lists database and product-specific upgrade tasks. You must be either at Release 10.7 (NCA, SmartClient, or character mode) or Release 11.0, to upgrade to Release 11*i*. You cannot upgrade to Release 11*i* directly from releases prior to 10.7.

### Maintaining Oracle Applications

Use this guide to help you run the various AD utilities, such as AutoUpgrade, AutoPatch, AD Administration, AD Controller, AD Relink, License Manager, and others. It contains how-to steps, screenshots, and other information that you need to run the AD utilities. This guide also provides information on maintaining the Oracle applications file system and database.

### Oracle Applications System Administrator's Guide

This guide provides planning and reference information for the Oracle Applications System Administrator. It contains information on how to define security, customize menus and online help, and manage concurrent processing.

### Oracle Alert User's Guide

This guide explains how to define periodic and event alerts to monitor the status of your Oracle Applications data.

### Oracle Applications Developer's Guide

This guide contains the coding standards followed by the Oracle Applications development staff. It describes the Oracle Application Object Library components needed to implement the Oracle Applications user interface described in the *Oracle Applications User Interface Standards for Forms-Based Products*. It also provides information to help you build your custom Oracle9*i* Forms Developer forms so that they integrate with Oracle Applications.

### Oracle Applications User Interface Standards for Forms-Based Products

This guide contains the user interface (UI) standards followed by the Oracle Applications development staff. It describes the UI for the Oracle Applications products and how to apply this UI to the design of an application built by using Oracle Forms.

## Other Implementation Documentation

### Multiple Reporting Currencies in Oracle Applications

This manual details additional steps and setup considerations for implementing the Multiple Reporting Currencies feature.

### Multiple Organizations in Oracle Applications

This guide describes how to set up and use the Multiple Organization support feature, so you can define and support different organization structures when running a single installation of Oracle Applications.

### Oracle Workflow Guide

This guide explains how to define new workflow business processes as well as customize existing Oracle Applications-embedded workflow processes.You also use this guide to complete the setup steps necessary for any Oracle Applications product that includes workflow-enabled processes.

### Oracle Applications Flexfields Guide

This guide provides flexfields planning, setup and reference information. This manual also provides information on creating custom reports on flexfields data.

### Oracle eTechnical Reference Manuals

Each eTechnical Reference Manual (eTRM) contains database diagrams and a detailed description of database tables, forms, reports, and programs for a specific Oracle Applications product. This information helps you convert data from your existing applications, integrate Oracle Applications data with non-Oracle applications, and write custom reports for Oracle Applications products. Oracle eTRM is available on Oracle*Metalink*.

### Oracle Manufacturing APIs and Open Interfaces Manual

This manual contains up-to-date information about integrating with other Oracle Manufacturing applications and with your other systems. This documentation includes APIs and open interfaces found in Oracle Manufacturing.

### Oracle Order Management Suite APIs and Open Interfaces Manual

This manual contains up-to-date information about integrating with other Oracle Manufacturing applications and with your other systems. This documentation includes APIs and open interfaces found in Oracle Order Management Suite.

### Oracle Common Application Components Implementation Guide

Many CRM products use components from CRM Application Foundation. Use this guide to correctly implement CRM Application Foundation.

## Training and Support

### Training

Oracle offers training courses to help you and your staff master Oracle Customer Interaction History and reach full productivity quickly. You have a choice of educational environments. You can attend courses offered by Oracle University at any one of our many Education Centers, you can arrange for our trainers to teach at your facility, or you can use Oracle Learning Network (OLN), Oracle University's online education utility. In addition, Oracle training professionals can tailor standard courses or develop custom courses to meet your needs. For example, you may want to use your organization’s structure, terminology, and data as examples in a customized training session delivered at your own facility.

### Support

From on-site support to central support, our team of experienced professionals provides the help and information you need to keep Oracle Customer Interaction History working for you. This team includes your Technical Representative,

Account Manager, and Oracle's large staff of consultants and support specialists with expertise in your business area, managing an Oracle8*i* server, and your hardware and software environment.

### Oracle*MetaLink*

Oracl*eMetaLink* is your self-service support connection with web, telephone menu, and e-mail alternatives. Oracle supplies these technologies for your convenience, available 24 hours a day, 7 days a week. With Oracl*eMetaLin*k, you can obtain information and advice from technical libraries and forums, download patches, download the latest documentation, look at bug details, and create or update TARs. To use MetaLink, register at (http://metalink.oracle.com).

**Alerts:** You should check Oracl*eMetaLink* alerts before you begin to install or upgrade any of your Oracle Applications. Navigate to the Alerts page as follows: Technical Libraries/ERP Applications/Applications Installation and Upgrade/Alerts.

**Self-Service Toolkit:** You may also find information by navigating to the Self-Service Toolkit page as follows: Technical Libraries/ERP Applications/Applications Installation and Upgrade.

## 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 Oracle Applications 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.

## About Oracle

Oracle Corporation develops and markets an integrated line of software products for database management, applications development, decision support, and office automation, as well as Oracle Applications, an integrated suite of more than 160 software modules for financial management, supply chain management, manufacturing, project systems, human resources and customer relationship management.

Oracle products are available for mainframes, minicomputers, personal computers, network computers and personal digital assistants, allowing organizations to integrate different computers, different operating systems, different networks, and even different database management systems, into a single, unified computing and information resource.

Oracle is the world's leading supplier of software for information management, and the world's second largest software company. Oracle offers its database, tools, and applications products, along with related consulting, education, and support services, in over 145 countries around the world.

# 1 Introduction

Topics include:

- Section 1.1, "Oracle Interaction Center Overview"
- Section 1.2, "New in this Release"

## 1.1 Oracle Interaction Center Overview

Every customer interaction — a telephone call, an e-mail message, or a Web chat conversation — presents an opportunity to win new business or improve customer satisfaction. Oracle Interaction Center supports the management and processing of customer relationship activity across all channels of customer contact.

Oracle Interaction Center integrates with several customer relationship business applications in the Oracle eBusiness Suite. This allows access to centralized customer information and business application functionality.

Oracle Interaction Center consists of several products. The modules relating to inbound telephony and outbound telephony are bundled separately. Oracle Interaction Center products include:

- Section 1.1.1, "Oracle Advanced Inbound Telephony"
- Section 1.1.2, "Oracle Advanced Outbound Telephony"
- Section 1.1.3, "Oracle Customer Interaction History"
- Section 1.1.4, "Oracle Email Center"
- Section 1.1.5, "Oracle Interaction Blending"
- Section 1.1.6, "Oracle Interaction Center Intelligence"
- Section 1.1.7, "Oracle One-to-One Fulfillment"

- Section 1.1.8, "Oracle Scripting"
- Section 1.1.9, "Oracle Universal Work Queue"

### See Also

- Section 1.2, "New in this Release"
- Section 1.3, "Modified in this Release"
- Section 1.4, "Obsolete in this Release"

## 1.1.1 Oracle Advanced Inbound Telephony

Oracle Advanced Inbound Telephony is designed to consistently and effectively handle customer interactions by intelligently routing, queuing and distributing media items. Oracle Advanced Inbound Telephony offers CTI support for market-leading traditional ACD/PBX and IP Telephony platforms, and provides enhanced screen pops on customer data into the Oracle E-Business Suite application. Oracle Advanced Inbound Telephony is fully integrated with Oracle TeleSales, Oracle TeleService and Oracle Collections, thereby minimizing integration time and deployment costs. Oracle Advanced Inbound Telephony also provides the Oracle Telephony Adapter SDK, which can be used to integrate other PBX/ACD and CTI middleware combinations that are not supported by an Oracle telephony adapter.

Oracle Advanced Inbound Telephony is required to telephony enable business applications in the Oracle E-Business Suite. "Telephony-enabled" means that the application can communicate with a telephone system for inbound calls, outbound calls, or both by way of the CTI middleware that handles the messaging between the customer's PBX/ACD and the business application.

The Oracle Advanced Inbound Telephony bundle consists of the following products: Oracle Interaction Center Server Manager, Oracle Universal Work Queue, Oracle Telephony Manager, Oracle Interaction Center Intelligence and Oracle Interaction Blending. Oracle Interaction Center Server Manager is a set of shell scripts and batch files that start, stop and monitor the server processes in an interaction center server group.

### See Also

- Section 1.1.2, "Oracle Advanced Outbound Telephony"
- Section 1.1.3, "Oracle Customer Interaction History"
- Section 1.1.4, "Oracle Email Center"

- Section 1.1.5, "Oracle Interaction Blending"
- Section 1.1.6, "Oracle Interaction Center Intelligence"
- Section 1.1.7, "Oracle One-to-One Fulfillment"
- Section 1.1.8, "Oracle Scripting"
- Section 1.1.9, "Oracle Universal Work Queue"

## 1.1.2 Oracle Advanced Outbound Telephony

Oracle Advanced Outbound Telephony is another key part of the Oracle E-Business Suite of applications. It is the module of Oracle Interaction Center that addresses outbound telephony. Oracle Advanced Outbound Telephony consists of two main components:

- A tactical list manager, which determines who to call and when to call them
- An outbound dialing engine, which dials numbers and transfers live contacts to call center agents

Oracle Advanced Outbound Telephony integrates with and relies on Oracle Marketing to create campaigns and lists to execute. Oracle Advanced Outbound Telephony serves as the execution arm for these marketing lists to maximize both outbound list penetration and agent productivity. Oracle Advanced Outbound Telephony also integrates with desktop applications like Oracle TeleSales and Oracle Collections to handle the actual customer interactions. Oracle Advanced Outbound Telephony can be used any time agents need to contact parties via the telephone.

Oracle Advanced Outbound Telephony also integrates with Oracle Customer Interaction History to provide feedback that marketers can use to analyze and measure the success of the marketing campaign, thereby providing a closed-loop marketing process.

Oracle Advanced Outbound Telephony does not include any other telephony management modules, and thus requires the use of Oracle Advanced Inbound Telephony.

### See Also

- Section 1.1.1, "Oracle Advanced Inbound Telephony"
- Section 1.1.3, "Oracle Customer Interaction History"
- Section 1.1.4, "Oracle Email Center"

- Section 1.1.5, "Oracle Interaction Blending"
- Section 1.1.6, "Oracle Interaction Center Intelligence"
- Section 1.1.7, "Oracle One-to-One Fulfillment"
- Section 1.1.8, "Oracle Scripting"
- Section 1.1.9, "Oracle Universal Work Queue"

### 1.1.3 Oracle Customer Interaction History

Oracle Customer Interaction History provides applications with a common framework for capturing and accessing all "interaction" data associated with customer contacts. Oracle Customer Interaction History acts as a central repository and provides a consistent framework for tracking all automated or agent-based customer interactions.

Applications record interactions through the Oracle Customer Interaction History framework itself, or through other applications that use Oracle Customer Interaction History. This information can be accessed by using the Oracle Customer Interaction History user interface, by using the user interface of an application that is integrated with Oracle Customer Interaction History, or by calling the Oracle Customer Interaction History APIs.

#### See Also

- Section 1.1.1, "Oracle Advanced Inbound Telephony"
- Section 1.1.2, "Oracle Advanced Outbound Telephony"
- Section 1.1.4, "Oracle Email Center"
- Section 1.1.5, "Oracle Interaction Blending"
- Section 1.1.6, "Oracle Interaction Center Intelligence"
- Section 1.1.7, "Oracle One-to-One Fulfillment"
- Section 1.1.8, "Oracle Scripting"
- Section 1.1.9, "Oracle Universal Work Queue"

### 1.1.4 Oracle Email Center

Oracle Email Center is a comprehensive solution for managing high volumes of inbound and outbound e-mails. Oracle Email Center reduces the cost per email interaction by automatically replying to certain email inquiries as well as routing

others to a skilled set of agents and providing them with a full featured console with cross application functionality.

Oracle Email Center increases customer satisfaction and reduces customer attrition by providing quick, accurate and consistent responses. It also increases agent's efficiency through the use of a full featured, Email Center agent console thereby reducing agent turnover.

An easy-to-use graphical user interface called the Self Service Administration console enables the administrator to configure the system, define processing rules and publish business data, while the Supervisor console enables the supervisor to manage email queues and balance workload. In addition, Oracle Interaction Center Intelligence provides a comprehensive set of reports that enable directors and managers to track email activity and relate it to business events. Through its integration with other Oracle E-Business applications, Email Center provides its agents with cross-application functionality and at the same time provides a utility called Message Component to the business applications for viewing, composing and responding to email messages.

### See Also

- Section 1.1.1, "Oracle Advanced Inbound Telephony"
- Section 1.1.2, "Oracle Advanced Outbound Telephony"
- Section 1.1.3, "Oracle Customer Interaction History"
- Section 1.1.5, "Oracle Interaction Blending"
- Section 1.1.6, "Oracle Interaction Center Intelligence"
- Section 1.1.7, "Oracle One-to-One Fulfillment"
- Section 1.1.8, "Oracle Scripting"
- Section 1.1.9, "Oracle Universal Work Queue"

## 1.1.5 Oracle Interaction Blending

Oracle Interaction Blending is a server in an interaction center server group that delivers media (inbound telephony, web callbacks, and outbound telephony) to agents through the Oracle Universal Work Queue work selector according to service levels set in a service plan.

Service levels define the goals for servicing contact channels. Service plans specify the desired level of service for a particular time of day. Service levels are assigned to time intervals in a service plan.

### See Also

- Section 1.1.1, "Oracle Advanced Inbound Telephony"
- Section 1.1.2, "Oracle Advanced Outbound Telephony"
- Section 1.1.3, "Oracle Customer Interaction History"
- Section 1.1.4, "Oracle Email Center"
- Section 1.1.6, "Oracle Interaction Center Intelligence"
- Section 1.1.7, "Oracle One-to-One Fulfillment"
- Section 1.1.8, "Oracle Scripting"
- Section 1.1.9, "Oracle Universal Work Queue"

## 1.1.6 Oracle Interaction Center Intelligence

Oracle Interaction Center Intelligence is a Web-based reporting solution that provides intelligent reports that facilitate day-to-day operational and long-term strategic decisions.

The data is presented to the user in a easy-to-use portal format. This format gives the user a unified, role-based, and easily customized view of Interaction Center information. Data presented includes session information, agent productivity metrics and key performance indicators (e.g., speed to answer and abandon rate).

Oracle Interaction Center Intelligence is built on an Oracle proprietary Java-based technology stack (Oracle CRM Foundation, also known as JTT) and a three-tier architecture:

- The first tier consists of the front end (client) which presents the application through an Oracle Applications-compliant Web browser.
- The middle tier is comprised of the Apache Web server and application server, which included as part of the installation of Oracle Applications release 11*i*.
- The third tier represents the database, which is comprised of an Oracle 8*i* or 9*i* database.

### See Also

- Section 1.1.1, "Oracle Advanced Inbound Telephony"
- Section 1.1.2, "Oracle Advanced Outbound Telephony"
- Section 1.1.3, "Oracle Customer Interaction History"

- Section 1.1.4, "Oracle Email Center"
- Section 1.1.5, "Oracle Interaction Blending"
- Section 1.1.7, "Oracle One-to-One Fulfillment"
- Section 1.1.8, "Oracle Scripting"
- Section 1.1.9, "Oracle Universal Work Queue"

## 1.1.7 Oracle One-to-One Fulfillment

Oracle One-to-One Fulfillment provides Oracle E-Business Suite applications with a centralized mechanism for managing fulfillment. Fulfillment is the process of compiling and distributing information to customers.

Oracle One-to-One Fulfillment consists of an API, a server, an administration user interface called the Administration Console, and an agent interface used by Oracle Sales Online. The Oracle One-to-One Fulfillment API is used by the E-Business Suite applications to initiate a request for fulfillment processing. The fulfillment request identifies the agent or server, the content, and the channel. The Oracle One-to-One Fulfillment server processes the request. It compiles the personalized content and determines the recipients of e-mail, fax, print and physical collateral.

Each business application has a unique interface for making a fulfillment request on behalf of one or more parties. Examples of outbound correspondence in E-Business Suite include:

- Notification letters
- Dunning letters
- Lease terms
- Product information
- Survey e-mail invitations

Sales and marketing collateral

### See Also

- Section 1.1.1, "Oracle Advanced Inbound Telephony"
- Section 1.1.2, "Oracle Advanced Outbound Telephony"
- Section 1.1.3, "Oracle Customer Interaction History"
- Section 1.1.4, "Oracle Email Center"

- Section 1.1.5, "Oracle Interaction Blending"
- Section 1.1.6, "Oracle Interaction Center Intelligence"
- Section 1.1.8, "Oracle Scripting"
- Section 1.1.9, "Oracle Universal Work Queue"

### 1.1.8 Oracle Scripting

Oracle Scripting provides enterprises with scripts which guide customers, agents and employees through decision flows based on a series of questions and answers. Scripts can be used for a variety of purposes: as traditional interaction center scripts that guide agents through collection of information and provide proactive alerts; as Web-based surveys for customers, prospects, and employees; and as self-service Web scripts that integrate with enterprise Web pages and provide a mechanism to guide Web customers through decision processes. Oracle Scripting is composed of several components: Script Author, the Scripting Engine, the Scripting Administration console, and the Survey Administration console.

Script Author is the development tool with which customized business requirements are translated into miniature programs known as "scripts." Each implementation of Oracle Scripting employs at least one customized script built by Oracle Consulting, consulting partners, or the enterprise. There are various ways in which scripts can be employed to gather or distribute data for an enterprise. For example, a script can serve to unify an agent's desktop by integrating aspects of various applications, or as a survey questionnaire to solicit specific information from the sample or target population. Script Author offers two ways to create a script; a graphical layout tool and a Script Wizard component.

The Scripting Engine is responsible for displaying the script to the end user, interpreting the end user's responses to questions and answers, and processing custom code developed in support of the script. The Scripting Engine includes two interfaces (one for agents, and one for executing a script using a Web browser). Any script executed in the Web interface requires survey campaign administration.

The Scripting Administration console provides the user interface with which script developers can launch Script Author as a Java applet, and script administrators can administer Oracle Scripting files, as well as generate, view and analyze a panel footprint report.

The Survey Administration console provides the user interface with which survey administrators establish and maintain survey campaign information, define and manage survey deployments, and view responses from data received.

### See Also

- Section 1.1.1, "Oracle Advanced Inbound Telephony"
- Section 1.1.2, "Oracle Advanced Outbound Telephony"
- Section 1.1.3, "Oracle Customer Interaction History"
- Section 1.1.4, "Oracle Email Center"
- Section 1.1.5, "Oracle Interaction Blending"
- Section 1.1.6, "Oracle Interaction Center Intelligence"
- Section 1.1.7, "Oracle One-to-One Fulfillment"
- Section 1.1.9, "Oracle Universal Work Queue"

## 1.1.9 Oracle Universal Work Queue

Oracle Universal Work Queue provides one click access to work. It brings media queues, prioritized application work queues, work items and messages together in a single access point. Applications integrate with Oracle Universal Work Queue to enable customer channels and to centralize access to application work items. Customer channels deliver media (inbound telephone calls, outbound telephone calls, web callbacks, web collaboration requests and e-mail) to applications through Oracle Universal Work Queue. Applications generate work items (e.g., service requests, leads, opportunities, tasks) and distribute them through assignment, queues and work pools. Oracle Universal Work Queue provides the unifying link between media queues, prioritized application work queues, application work items and Oracle E-Business applications.

### See Also

- Section 1.1.1, "Oracle Advanced Inbound Telephony"
- Section 1.1.2, "Oracle Advanced Outbound Telephony"
- Section 1.1.3, "Oracle Customer Interaction History"
- Section 1.1.4, "Oracle Email Center"
- Section 1.1.5, "Oracle Interaction Blending"
- Section 1.1.6, "Oracle Interaction Center Intelligence"
- Section 1.1.7, "Oracle One-to-One Fulfillment"
- Section 1.1.8, "Oracle Scripting"

## 1.2 New in this Release

The following features are new in this release.

- Section 1.2.1, "New Responsibilities"
- Section 1.2.2, "Customer Interaction History Viewer Context"
- Section 1.2.3, "Interaction History Bulk API and Processor"
- Section 1.2.4, "Customer Contact Information Captured and Displayed"

### See Also

- Section 1.1, "Oracle Interaction Center Overview"
- Section 1.3, "Modified in this Release"
- Section 1.4, "Obsolete in this Release"

### 1.2.1 New Responsibilities

The following responsibilities have been added in this release:

- Interaction History Migration: From the E-Business Home Page or the CRM Home Page, use the Interaction History Migration responsibility to migrate existing data from the outcome-result pair, result-reason pair tables and the Interaction and Activities history tables to the wrap-up table.
- Interaction History Self Service: From the E-Business Home Page or the CRM Home Page, use the Interaction History Self Service responsibility to view logged interactions in the Oracle Applications Framework interface.

### 1.2.2 Customer Interaction History Viewer Context

In this release all new and existing end-user interfaces have been modified to require the user to first select a customer or account context in which they will view interactions and activities. This is only required when access the interfaces from the Oracle Customer Interaction History responsibilities. This as been done to mimic the customer context approached used in the applications that embed these interfaces: Oracle TeleSales, Oracle Customer Care, Oracle Sales On-line, etc. Presenting the user with a page or form in which they must select the context criteria before they are allowed to proceed to the search interface does this. The context customer or account is then used in all subsequent searches. Performed until the user selects to close the search and re-start with a different context. The forms based user interface allows the user to turn off the context once they have

entered the search form to allow searches across multiple customers and accounts. This is allowed to maintain functional compatibility with previous releases.

### See Also

Chapter 6, "Using Oracle Customer Interaction History"

### 1.2.3 Interaction History Bulk API and Processor

The Interaction History Bulk API and Processor are used to improve performance when logging large batches of interaction. This interface is used exclusively by Oracle One-to-One Fulfillment. For information about setting up Oracle One-to-One Fulfillment, see *Oracle One-to-One Fulfillment Implementation Guide*.

If you use One-To-One Fulfillment, use the CRM Administrator responsibility to access the Interaction History Bulk Processor concurrent program to log sent fulfillments as interactions. You can also schedule the bulk API processor to run periodically to automatically log these interactions without further administrator intervention. Any errors encountered during the running of the concurrent program can be viewed in the Oracle Customer Interaction History administration console on the Bulk Processing Errors subtab.

### 1.2.4 Customer Contact Information Captured and Displayed

The customer contact party information is now captured on interactions and is displayed in interaction user interfaces. The Oracle Customer Interaction History search user interfaces have been modified to allow the user to search and sort by contact name.

## 1.3 Modified in this Release

The following features are modified in this release.

- Section 1.3.1, "Wrap-Up Administration"
- Section 1.3.2, "Wrap-Up Migration"

### See Also

- Section 1.1, "Oracle Interaction Center Overview"
- Section 1.2, "New in this Release"
- Section 1.4, "Obsolete in this Release"

### 1.3.1 Wrap-Up Administration

The wrap-ups administration has been enhanced to allow the user to filter the list of wrap-ups and to copy a set of defined wrap-ups from one Marketing Source (campaign, offer, etc.) to another. Further, when creating new wrap-ups, the List Of Values (LOV) interface for Marketing Source has been enhanced to include the name value for each Marketing Source and to allow sorting of the list of values.

### 1.3.2 Wrap-Up Migration

The wrap-up migration logic has been modified to look at the existing interactions and activities in history and use the unique Outcome, Result and Reason combinations recorded in history to create wrap-ups. This change is made to ease the burden on the administrator in having to create all of the combinations desired via the wrap-ups administration interface.

## 1.4 Obsolete in this Release

The following features are obsolete in the release.

- Outcome-Result Administration
- Result-Reason Administration
- Forms-Based Administration

#### See Also

- Section 1.1, "Oracle Interaction Center Overview"
- Section 1.2, "New in this Release"
- Section 1.3, "Modified in this Release"

### 1.4.1 Outcome-Result Administration

Outcome-result pairs are obsolete. They have been replaced with wrap-ups. Outcome-result pairs no longer appear in Oracle Customer Interaction History administration. Use the Interaction History Migration responsibility to convert existing outcome-result pairs to wrap-ups.

### 1.4.2 Result-Reason Administration

Result-reason pairs are obsolete. They have been replaced with wrap-ups. Result-reason pairs no longer appear in Oracle Customer Interaction History

administration. Use the Interaction History Migration responsibility to convert existing result-reason pairs to wrap-ups.

### 1.4.3 Forms-Based Administration

Forms-based administration for Oracle Customer Interaction History, which was obsoleted in 11.5.8, is no longer accessible.

# 2

# Detailed Product Description

Topics include:



## 2.1 Oracle Customer Interaction History Overview

Oracle Customer Interaction History provides applications with a common framework for capturing and accessing all "interaction" data associated with customer contacts. Oracle Customer Interaction History acts as a central repository and provides a consistent framework for tracking all automated or agent-based customer interactions.

Applications record interactions through the Oracle Customer Interaction History framework itself, or through other applications that use Oracle Customer Interaction History. This information can be accessed by using the Oracle Customer Interaction History user interface, by using the user interface of an application that is integrated with Oracle Customer Interaction History, or by calling the Oracle Customer Interaction History APIs.

Within Oracle Customer Interaction History, you can view customer-agent interactions, activities and notes. Interactions and activities can be filtered by different criteria.

## 2.2 Concepts

Topics include:

- Section 2.2.1, "Interaction"
- Section 2.2.5, "Outcome"
- Section 2.2.6, "Result"
- Section 2.2.7, "Reason"
- Section 2.2.4, "Activity Type"
- Section 2.2.3, "Action"
- Section 2.2.8, "Wrap Up"

## 2.2.1 Interaction

An interaction is a point of contact or touchpoint between a human or automated agent and a party such as a customer, a customer system, or a potential customer. An example of a touchpoint is a phone call between an agent and a customer. Interactions include activities, media, and media items.

An interaction is a timed entity with an outcome and a result that can be tracked. When an interaction is closed, it becomes an historical record that subsequently cannot be altered or modified. Multiple forms of communication or media items between the party and the agent can be included in a single interaction.

## 2.2.2 Activity

An activity describes the elements of an event that take place during an interaction. An activity includes an action (such as, creating or sending) and an activity type (such as, a service request or collateral). An interaction must have at least one activity.

### Example

A customer calls an agent to request service. The agent create a service request, sends a piece of collateral, and updates the notes in the customer record.

The activities are:

- Creating a service request.
- Sending collateral.
- Updating a note.

### 2.2.3 Action

An action is an act that is performed during an interaction (such as, creating or sending). An action is one element of an activity. An activity is defined by an action and an activity type.

#### Example

In the following list, creating, sending, and updating are actions.

- Creating a service request.
- Sending collateral.
- Updating a note.

### 2.2.4 Activity Type

An activity type is an object that is acted upon during an interaction (such as, a service request or a piece of collateral). An activity is one element of an activity. An activity is defined by an activity type and an action.

#### Example

In the following list, service request, collateral, and note are activity types.

- Creating a service request.
- Sending collateral.
- Updating a note.

### 2.2.5 Outcome

An outcome describes the outcome of a customer interaction (for example, making contact with the customer or reaching an answering machine). An outcome can be assigned manually by the agent or automatically by the application.

You can require the assignment of a result when a particular outcome is assigned to an interaction. In addition, the assignment of an outcome can generate a callback so that an agent calls the customer back at another time.

### 2.2.6 Result

A result describes the consequence of an outcome. Using a wrap up, you can relate an outcome to one or more results (for example, a outcome of "contact" with a result

of "no sale"). Also, you can require the assignment of a reason when a particular result is assigned to an interaction.

### 2.2.7 Reason

A reason provides an explanation for the result. Using a wrap up, you can relate a result to one or more reasons (for example, a contact outcome with a result of "no sale" and a reason of "no money").

### 2.2.8 Wrap Up

A wrap up relates outcomes to results and reasons. You can limit the availability of a wrap up to a specific campaign type or code. When the wrap up is selected for an interaction, the outcome, result, and reason in the wrap up definition are assigned to the interaction.

# 3

# Before You Begin

This chapter describes the system requirements and product dependencies for Oracle Customer Interaction History. It also describes how to access the administrative console for Oracle Customer Interaction History.

Topics include:



## 3.1 Mandatory Dependencies

Mandatory dependencies are required in order for Oracle Customer Interaction History to function. Oracle Customer Interaction History relies on several products for data and functionality. The following products must be installed and implemented before you begin the implementation of Oracle Customer Interaction History:

Oracle Common Application Components

### Oracle Common Application Components

Resource Manager, Task Manager, and Notes are required for Oracle Customer Interaction History to function.

For information about Oracle Common Application Components, see *Oracle Common Application Components Implementation Guide* and *Oracle Common Application Components User Guide*.

## 3.2 Related Documentation

The latest product documentation is available on Oracle*MetaLink* at http://metalink.oracle.com.

## 3.3 Installing Oracle Customer Interaction History

You have the following options for installing Oracle Customer Interaction History:

- Oracle Applications Rapid Install

  The Rapid Install is intended for customers who are installing Oracle Applications for the first time or upgrading to Release 11i from to Release 11.0 or Release 10.7. It contains the family packs or product minipacks for all products in Oracle Applications.

  The Rapid Install is provided on CD-ROMs and is available from Oracle Store at http://oraclestore.oracle.com. For information about installing Oracle Applications using Rapid Install, see *Installing Oracle Applications*. For information about upgrading Oracle Applications using Rapid Install, see *Upgrading Oracle Applications*.

- Oracle Applications Maintenance Pack

  The Maintenance Pack is intended for customers who have already installed Oracle Applications Release 11i. It contains the family packs or product minipacks for all products in Oracle Applications.

  The Maintenance Pack is provided as a patch and is available on Oracle*MetaLink* at http://metalink.oracle.com. For information about upgrading Oracle Applications Release 11i using the Maintenance Pack, see Maintenance Pack Release Instructions on Oracle*MetaLink* at http://metalink.oracle.com. (Perform an advanced search for Document ID 232834.1. Enter the document ID in the Search Field and select the Doc ID option.)

- Oracle Customer Interaction History Minipack (11i.JTH.R)

  The Oracle Customer Interaction History Minipack is intended for customers who have already installed or upgraded to Oracle Applications Release 11i and wish to upgrade only Oracle Customer Interaction History. The minipack is cumulative and contains only the patches for Oracle Customer Interaction History.

**Note:** Oracle Customer Interaction History integrates with other products in Oracle Applications. Therefore, you may have to install family packs, product minipacks, or individual product patches for *other* products before installing the Oracle Customer Interaction History Minipack.

The Oracle Customer Interaction History Minipack is provided as a patch (3100686) and is available on OracleMetaLink at http://metalink.oracle.com. The readme and patch list for Oracle Customer Interaction History Minipack are available on OracleMetaLink at http://metalink.oracle.com. (Perform an advanced search for Document ID 266172.1. Enter the document ID in the Search Field and select the Doc ID option.)

## 3.4 Accessing Oracle Customer Interaction History

The product interfaces are accessed by providing the Uniform Resource Locator (URL) for the environment in an Oracle Applications 11i-compliant Web browser and navigating to the hyperlink for the login page for the specific technology stack. You can also provide the URL for the specific login page. This URL is referred to as your login URL.

**Oracle Applications URL** Use this URL to navigate to the E-Business Home Page URL or the CRM Home page URL.

```
http://<host>:<port>/
```

- To navigate to the E-Business Home Page URL, choose **Apps Logon Links** > **E-Business Home Page**.
- To navigate to the CRM Home Page URL, choose **Apps Logon Links** > **CRM Home Page**.

**CRM Home Page URL** This URL is sometimes referred to as the Apache or JTF login URL. Use this URL to open the login page for HTML-based applications.

```
http://<host>:<port>/OA_HTML/jtflogin.jsp
```

**E-Business Home Page URL:** This URL is sometimes referred to as the Self-Service Web Applications or SSWA login URL. Use this URL to open the login window for Oracle Applications via the E-Business Home Page. You can access Forms-based or HTML-based applications from the Personal Home Page.

```
http://<host>:<port>/OA_HTML/US/ICXINDEX.htm
```

### 3.4.1 User Accounts

An application user is an authorized user of Oracle Applications and is uniquely identified by a username. After the user account has been defined, the application user can sign on to Oracle Applications at the E-Business Home Page or CRM Home Page login.

---

**Note:** Oracle Applications is installed with a system defined username and password.

- Username: sysadmin
- Password: sysadmin

---

An application user enters a username along with a password to sign on to Oracle Applications. The password assigned by the system administrator is temporary. When signing on for the first time, the application user will be prompted to change the password. Access to specific functionality and data will be determined by the responsibilities assigned to your user account.

### 3.4.2 Responsibilities

A system administrator assigns one or more responsibilities to an application user. A responsibility is a level of authority that allows a user to access specific functionality and data in Oracle Applications. Oracle Applications is installed with predefined responsibilities. A system administrator can modify a predefined responsibility or create custom responsibilities.

The following table describes the predefined responsibilities that are used to implement Oracle Customer Interaction History.

| Responsibility | Function | Interface |
|---|---|---|
| CRM Administrator | Schedule the Interaction History Bulk Processor concurrent program. | E-Business Home Page |
| | Schedule the Interaction History Data Import concurrent program in order to import interaction data from a third-party or legacy system into Oracle Applications | |

| Responsibility | Function | Interface |
|---|---|---|
| Interaction History | View interactions in the Forms interface:<br>■ Launch the Customer Interaction History Test page form.<br>■ Select a customer party or account to be passed to the Customer Interaction History Search form. | E-Business Home Page |
| Interaction History Data Import | Schedule the Interaction History Data Import concurrent program in order to import interaction data from a third-party or legacy system into Oracle Applications | E-Business Home Page |
| Interaction History Data Purge | Schedule the Interaction History Data Purge concurrent program in order to purge interaction data from Oracle Applications. | E-Business Home Page |
| Interaction History JSP Admin | Access the administration console:<br>■ Create outcomes, results, reasons, activities, actions, and wrap ups.<br>■ Associate actions and activities.<br>■ View bulk processing errors. | E-Business Home Page<br>or<br>CRM Home Page |
| Interaction History JSP User | View interactions in the JSP interface:<br>■ Launch the Interaction History Viewer page.<br>■ View interactions and activities.<br>■ Search for interactions and activities. | E-Business Home Page<br>or<br>CRM Home Page |
| Interaction History Migration | Verify and set up data in old tables in order to migrate outcome-result pairs and result-reasons pairs to wrap ups. | E-Business Home Page<br>or<br>CRM Home Page |
| Interaction History Self Service | View interactions in the Oracle Applications Framework interface. | E-Business Home Page<br>or<br>CRM Home Page |
| System Administrator | Create a user for administering Oracle Customer Interaction History.<br>Set values for profile options. | E-Business Home Page<br>or<br>CRM Home Page |

In the E-Business Home Page, after the user signs on, a list of available responsibilities appears. To switch responsibilities in the E-Business Home Page,

click a responsibility. To switch responsibilities in the Forms interface, choose Switch Responsibility from the File menu.

In the CRM Home Page, after the user signs on, the user must select a default responsibility (even if the user has only one responsibility). The next time the user signs on, the tabs related to the default responsibility appear. To switch responsibilities, go to Navigation Preferences in your profile (Profile icon). In the Switch Responsibilities section, select another responsibility from the Current Responsibility list.

# 4

# Implementation Tasks

There are a number of procedures which must be completed in order to implement Oracle Customer Interaction History. This chapter provides a high-level overview of the implementation task sequence and references to procedures that are performed during the course of an implementation.

Topics include:



## 4.1 Implementation Task Sequence

Perform the steps in the following table to implement Oracle Customer Interaction History. The Number column indicates the step order. The Required column indicates whether a step is required. The Description column describes a high-level step and, where applicable, provides a reference to a more detailed topic in this document. The Responsibility column indicates the Oracle Applications user account responsibility required to complete the step.

| Step | Required | Description | Responsibility |
|---|---|---|---|
| 1 | Yes | **Create an Oracle Customer Interaction History administrator.**<br>You will use this user account to perform the remaining steps.<br>See: Section 4.2, "Defining an Administrator" | System Administrator |
| 2 | No | **Migrate outcomes-result pairs and result-reason pairs to wrap-ups tables.**<br>See: Appendix A, "Data Migration" | Interaction History Migration |

| Step | Required | Description | Responsibility |
|---|---|---|---|
| 3 | No | **Define interaction outcomes.**<br>See: Section 5.1, "Outcome" | Interaction History JSP Administrator |
| 4 | No | **Define interaction results.**<br>See: Section 5.2, "Result" | Interaction History JSP Administrator |
| 5 | No | **Define interaction reason codes.**<br>See: Section 5.3, "Reason" | Interaction History JSP Administrator |
| 6 | No | **Define interaction activity types.**<br>See: Section 5.4, "Activity Type" | Interaction History JSP Administrator |
| 7 | No | **Define interaction actions.**<br>See: Section 5.5, "Action" | Interaction History JSP Administrator |
| 8 | No | **Define interaction wrap ups.**<br>See: Section 5.6, "Wrap Up" | Interaction History JSP Administrator |
| 9 | No | **Associate an action with an activity type.**<br>See: Section 5.7, "Action-Activity Type" | Interaction History JSP Administrator |
| 10 | For applications that use One-to-One Fulfillment | **Set up Interaction History Bulk Processor.**<br>See: Section B.1, "Interaction History Bulk Processor" | CRM Administrator |

### See Also

Section 4.2, "Defining an Administrator"

## 4.2 Defining an Administrator

Use the following procedure to create a user account for administering Oracle Customer Interaction History.

### Login

E-Business Home Page

### Responsibility

System Administrator

### Prerequisites

None

### Steps

1. In the Navigator window, on the Functions tab, choose **Security** > **User** > **Define**.

   The User window appears.

   Use the following guidelines to define Oracle Applications user names:

   - Use only one word.
   - Use only alphanumeric characters ('A' through 'Z', and '0' through '9').
   - Use only the set of characters that your operating system supports for filenames.

2. In the User Name field, enter the name of the user.

   The password is temporary. When the user signs on to Oracle Applications for the first time, the message "Your password has expired" appears and the user is prompted to set a new password.

   Use the following guidelines to define Oracle Applications passwords:

   - Use at least five characters and no more than 100 characters.
   - Use only alphanumeric characters ('A' through 'Z', and '0' through '9').

3. In the Password field, enter the password for the user account and then press Tab.

   The cursor remains in the Password field.

4. Enter the password again to verify it.

5. If you want to use this account to submit concurrent programs for Oracle Customer Interaction History, then select an employee to associate with this user account from the Person field.

6. In the Responsibilities tab, add the following responsibilities:

| Responsibility | Function | Interface |
|---|---|---|
| CRM Administrator | Schedule the Interaction History Bulk Processor concurrent program.<br>Schedule the Interaction History Data Import concurrent program in order to import interaction data from a third-party or legacy system into Oracle Applications | E-Business Home Page |
| Interaction History | View interactions in the Forms interface:<br>■ Launch the Customer Interaction History Test page form.<br>■ Select a customer party or account to be passed to the Customer Interaction History Search form.<br>Access the administration console in the JSP interface:<br>■ Create outcomes, results, reasons, activities, actions, and wrap ups.<br>■ Associate actions and activities.<br>■ View bulk processing errors. | E-Business Home Page (Forms or JSP)<br>CRM Home Page (JSP) |
| Interaction History Data Import | Schedule the Interaction History Data Import concurrent program in order to import interaction data from a third-party or legacy system into Oracle Applications | E-Business Home Page |
| Interaction History Data Purge | Schedule the Interaction History Data Purge concurrent program in order to purge interaction data from Oracle Applications. | E-Business Home Page |
| Interaction History JSP Admin | Access the administration console:<br>■ Create outcomes, results, reasons, activities, actions, and wrap ups.<br>■ Associate actions and activities.<br>■ View bulk processing errors. | E-Business Home Page<br>or<br>CRM Home Page |
| Interaction History JSP User | View interactions in the JSP interface:<br>■ Launch the Interaction History Viewer page.<br>■ View interactions and activities.<br>■ Search for interactions and activities. | E-Business Home Page<br>or<br>CRM Home Page |

| Responsibility | Function | Interface |
|---|---|---|
| Interaction History Migration | Verify and set up data in old tables in order to migrate outcome-result pairs and result-reasons pairs to wrap ups. | E-Business Home Page<br>or<br>CRM Home Page |
| Interaction History Self Service | View interactions in the Oracle Applications Framework interface. | E-Business Home Page<br>or<br>CRM Home Page |
| System Administrator | Create a user for administering Oracle Customer Interaction History.<br>Set values for profile options. | E-Business Home Page<br>or<br>CRM Home Page |

Once the user record has been saved, you cannot delete an assigned responsibility. Oracle Applications maintains audit data for assigned responsibilities.

To deactivate an assigned responsibility, set the effective end date (in the Effective Dates - To field) of the assigned responsibility to the current date. To activate an assigned responsibility, clear or reset the effective end date.

7. From the **File** menu, choose **Save**.

   You may close the Users window.

## See Also

Section 4.1, "Implementation Task Sequence"

# 5

# Administration Tasks

This chapter describes task-based procedures for administering Oracle Customer Interaction History. Topics include:

- Section 5.1, "Outcome"
- Section 5.2, "Result"
- Section 5.3, "Reason"
- Section 5.4, "Activity Type"
- Section 5.5, "Action"
- Section 5.6, "Wrap Up"
- Section 5.7, "Action-Activity Type"

## 5.1 Outcome

Use the Outcome hyperlink to maintain a list of interaction outcomes.

### Tasks

You can perform the following tasks:

- Section 5.1.1, "Creating an Outcome"
- Section 5.1.2, "Updating an Outcome"
- Section 5.1.3, "Removing an Outcome"

### 5.1.1 Creating an Outcome

Use this procedure to define an outcome.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History or Interaction History JSP Admin

### Prerequisites

None

### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Outcome**.

   The Outcomes page appears.
4. Click **Create**.

   The Create Outcome window opens.
5. In the Code field, type a code for the outcome.

   You cannot change this field after the outcome is saved.
6. In the Short Description field, type a short description of the outcome.

   The short description appears in the list of outcomes.
7. Optionally, in the Long Description field, type a more detailed description of the outcome.

   This field is for informational purposes only.
8. If you want to indicate that the outcome is positive, then select the **Positive Outcome** box.
9. If you want to require a result with this outcome, then select the **Result Required** box.
10. If you want to indicate that the outcome is related to a phone call, then select the **Telephony Related** box.
11. In the Generate Callback field, select the type of callback that you want the outcome to generate.

    You have the following options:

- Private: Generate a callback that must be made by the original agent.
- Public: Generate a callback that can be made by any agent.
- None: Do not generate a callback.

12. In the Success field, indicate whether the outcome is a successful outcome.

    You cannot change this field after the server is saved.

13. If you want to use the outcome immediately, then select the **Active** box.
14. Click **Create**.

### See Also

- Section 5.1.2, "Updating an Outcome"
- Section 5.1.3, "Removing an Outcome"

## 5.1.2 Updating an Outcome

Use this procedure to update an outcome.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History or Interaction History JSP Admin

### Prerequisites

None

### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Outcome**.

   The Outcomes page appears.

4. Click an ID hyperlink.

   The Edit Outcome page appears.

5. In the Short Description field, enter a short description of the outcome.

   The short description appears in the list of outcomes.

6. Optionally, in the Long Description field, enter a more detailed description of the outcome.

   This field is for informational purposes only.

7. If you want to indicate that the outcome is positive, then select the **Positive Outcome** box.

8. If you want to require a result with this outcome, then select the **Result Required** box.

9. In the Generate Callback field, select the type of callback that you want the outcome to generate.

   You have the following options:

   - Private: Generate a callback that must be made by the original agent.
   - Public: Generate a callback that can be made by any agent.
   - None: Do not generate a callback.

10. If you want to indicate that the outcome is related to a phone call, then select the **Telephony Recycling Code** box.

11. If you want to use the outcome immediately, then select the **Active** box.

12. Click **Update**.

### See Also

- Section 5.1.1, "Creating an Outcome"
- Section 5.1.3, "Removing an Outcome"

## 5.1.3 Removing an Outcome

Use this procedure to delete a user-defined outcome. You cannot delete seeded outcomes.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History or Interaction History JSP Admin

### Prerequisites

None

### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Outcome**.

   The Outcomes page appears.
4. Select the **Select** box for the outcome that you want to delete.

   You cannot delete seeded outcomes.
5. Click **Delete**.

   The selected outcome is removed from the list and the Outcomes page refreshes.

### See Also

- Section 5.1.1, "Creating an Outcome"
- Section 5.1.2, "Updating an Outcome"

## 5.2 Result

Use the Result hyperlink to maintain a list of results.

### Tasks

You can perform the following tasks:

- Section 5.2.1, "Creating a Result"
- Section 5.2.2, "Updating a Result"
- Section 5.2.3, "Removing a Result"

## 5.2.1 Creating a Result

Use this procedure to define a result.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History or Interaction History JSP Admin

### Prerequisites

None

### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Result**.

   The Results page appears.
4. Click **Create**.

   The Create Result window opens.
5. In the Code field, type a code for the result.

   You cannot change this field after the result is saved.
6. In the Short Description field, enter a short description of the result.

   The short description appears in the list of results.
7. Optionally, in the Long Description field, enter a more detailed description of the result.

   This field is for informational purposes only.
8. If you want to indicate that the result is positive, then select the **Positive Result** box.
9. If you want to require a reason with this result, then select the **Reason Required** box.
10. If you want to use the result immediately, then select the **Active** box.

11. Click **Create**.

### See Also

- Section 5.2.2, "Updating a Result"
- Section 5.2.3, "Removing a Result"

## 5.2.2 Updating a Result

Use this procedure to update a result.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History or Interaction History JSP Admin

### Prerequisites

None

### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Result**.

   The Results page appears.
4. Click an ID hyperlink.

   The Edit Result page appears.
5. In the Short Description field, enter a short description of the result.

   The short description appears in the list of results.
6. Optionally, in the Long Description field, enter a more detailed description of the result.

   This field is for informational purposes only.
7. If you want to indicate that the result is positive, then select the **Positive Result** box.

8. If you want to require a reason with this result, then select the **Reason Required** box.
9. If you want to use the result immediately, then select the **Active** box.
10. Click **Update**.

### See Also

- Section 5.2.1, "Creating a Result"
- Section 5.2.3, "Removing a Result"

## 5.2.3 Removing a Result

Use this procedure to delete a user-defined result. You cannot delete seeded results.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History or Interaction History JSP Admin

### Prerequisites

None

### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Result**.

   The Results page appears.
4. Select the **Select** box for the result that you want to delete.

   You cannot delete seeded results.
5. Click **Delete**.

   The selected result is removed from the list and the Results page refreshes.

### See Also

- Section 5.2.1, "Creating a Result"

- Section 5.2.2, "Updating a Result"

## 5.3 Reason

Use the Reason hyperlink to maintain a list of reasons.

### Tasks

You can perform the following tasks:

- Section 5.3.1, "Creating a Reason"
- Section 5.3.2, "Updating a Reason"
- Section 5.3.3, "Removing a Reason"

### 5.3.1 Creating a Reason

Use this procedure to define a reason.

#### Login

E-Business Home Page or CRM Home Page

#### Responsibility

Interaction History or Interaction History JSP Admin

#### Prerequisites

None

#### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Reason**.

   The Reasons page appears.
4. Click **Create**.

   The Create Reason window opens.
5. In the Code field, type a code for the reason.

   You cannot change this field after the reason is saved.

6. In the Short Description field, enter a short description of the reason.

   The short description appears in the list of reasons.

7. Optionally, in the Long Description field, enter a more detailed description of the reason.

   This field is for informational purposes only.

8. If you want to use the reason immediately, then select the **Active** box.

9. Click **Create**.

### See Also

- Section 5.3.2, "Updating a Reason"
- Section 5.3.3, "Removing a Reason"

## 5.3.2 Updating a Reason

Use this procedure to update a reason.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History or Interaction History JSP Admin

### Prerequisites

None

### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Reason**.

   The Reasons page appears.

4. Click an ID hyperlink.

   The Edit Reason page appears.

5. In the Short Description field, enter a short description of the reason.

The short description appears in the list of reasons.

6. Optionally, in the Long Description field, enter a more detailed description of the reason.

   This field is for informational purposes only.

7. If you want to use the reason immediately, then select the **Active** box.
8. Click **Update**.

### See Also

- Section 5.3.1, "Creating a Reason"
- Section 5.3.3, "Removing a Reason"

## 5.3.3 Removing a Reason

Use this procedure to delete a user-defined reason. You cannot delete seeded reasons.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History or Interaction History JSP Admin

### Prerequisites

None

### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Reason**.

   The Reasons page appears.

4. Select the **Select** box for the reason that you want to delete.

   You cannot delete seeded reasons.

5. Click **Delete**.

The selected reason is removed from the list and the Reasons page refreshes.

### See Also

- Section 5.3.1, "Creating a Reason"
- Section 5.3.2, "Updating a Reason"

## 5.4 Activity Type

Use the Activity Type hyperlink to maintain a list of activity types.

### Tasks

You can perform the following tasks:

- Section 5.4.1, "Creating an Activity Type"
- Section 5.4.2, "Updating an Activity Type"
- Section 5.4.3, "Removing an Activity Type"

### 5.4.1 Creating an Activity Type

Use this procedure to define an activity type.

#### Login

E-Business Home Page or CRM Home Page

#### Responsibility

Interaction History or Interaction History JSP Admin

#### Prerequisites

None

#### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Activity Type**.

   The Activity Types page appears.

4. Click **Create**.

   The Create Activity Type window opens.

5. In the Code field, type a code for the activity type.

   You cannot change this field after the activity type is saved.

6. In the Short Description field, enter a short description of the activity type.

   The short description appears in the list of activity types.

7. If you want to use the activity type immediately, then select the **Active** box.

8. Click **Create**.

### See Also

- Section 5.4.2, "Updating an Activity Type"
- Section 5.4.3, "Removing an Activity Type"

## 5.4.2 Updating an Activity Type

Use this procedure to update an activity type.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History or Interaction History JSP Admin

### Prerequisites

None

### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Activity Type**.

   The Activity Types page appears.

4. Click an ID hyperlink.

The Edit Activity Type page appears.

5. In the Short Description field, enter a short description of the reason.

   The short description appears in the list of reasons.

6. If you want to use the activity type immediately, then select the **Active** box.
7. Click **Update**.

### See Also

- Section 5.4.1, "Creating an Activity Type"
- Section 5.4.3, "Removing an Activity Type"

## 5.4.3 Removing an Activity Type

Use this procedure to delete a user-defined activity type. You cannot delete seeded activity types.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History or Interaction History JSP Admin

### Prerequisites

None

### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Activity Types**.

   The Activity Types page appears.

4. Select the **Select** box for the activity type that you want to delete.

   You cannot delete seeded activity types.

5. Click **Delete**.

The selected activity type is removed from the list and the Activity Types page refreshes.

### See Also

- Section 5.4.1, "Creating an Activity Type"
- Section 5.4.2, "Updating an Activity Type"

## 5.5 Action

Use the Action hyperlink to maintain a list of actions.

### Tasks

You can perform the following tasks:

- Section 5.5.1, "Creating an Action"
- Section 5.5.2, "Updating an Action"
- Section 5.5.3, "Removing an Action"

### 5.5.1 Creating an Action

Use this procedure to define an action.

#### Login

E-Business Home Page or CRM Home Page

#### Responsibility

Interaction History or Interaction History JSP Admin

#### Prerequisites

None

#### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Action**.

   The Actions page appears.

4. Click **Create**.

   The Create Action window opens.

5. In the Code field, type a code for the action.

   You cannot change this field after the action is saved.

6. In the Short Description field, enter a short description of the action.

   The short description appears in the list of actions.

7. If you want to use the action immediately, then select the **Active** box.

8. Click **Create**.

### See Also

- Section 5.5.2, "Updating an Action"
- Section 5.5.3, "Removing an Action"

## 5.5.2 Updating an Action

Use this procedure to update an action.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History or Interaction History JSP Admin

### Prerequisites

None

### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Action**.

   The Actions page appears.

4. Click an ID hyperlink.

   The Edit Action page appears.

5. In the Short Description field, enter a short description of the reason.

   The short description appears in the list of reasons.

6. If you want to use the activity type immediately, then select the **Active** box.
7. Click **Update**.

### See Also

- Section 5.5.1, "Creating an Action"
- Section 5.5.3, "Removing an Action"

## 5.5.3 Removing an Action

Use this procedure to delete a user-defined action. You cannot delete seeded actions.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History or Interaction History JSP Admin

### Prerequisites

None

### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Actions**.

   The Actions page appears.

4. Select the **Select** box for the action that you want to delete.

   You cannot delete seeded actions.

5. Click **Delete**.

   The selected action is removed from the list and the Actions page refreshes.

### See Also

- Section 5.5.1, "Creating an Action"
- Section 5.5.2, "Updating an Action"

## 5.6 Wrap Up

Use the Wrap Up hyperlink to maintain a list of wrap ups.

### Tasks

You can perform the following tasks:

- Section 5.6.1, "Creating a Wrap Up"
- Section 5.6.2, "Updating a Wrap Up"
- Section 5.6.3, "Removing a Wrap Up"

### 5.6.1 Creating a Wrap Up

Use this procedure to maintain a list of a wrap ups.

#### Login

E-Business Home Page or CRM Home Page

#### Responsibility

Interaction History or Interaction History JSP Admin

#### Prerequisites

- Create an outcome.
- Create a result.
- Create a reason.

#### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Wrap Up**.

   The Wrap Ups page appears.

4. To filter the list of wrap ups, select a filter option.
5. To filter the list of wrap ups, select the filter criteria.

   You have the following options:

   - All
   - Base
6. To copy a wrap up from one marketing source to another, select the Marketing Source Name option, type the name of the marketing source, and then click Copy.
7. Click **Create**.

   The Create Wrap Up page appears.
8. Optionally, in the Marketing Source Code field, type a marketing source code or click **Go** to search for a marketing source code.
9. Optionally, from the Marketing Source Name field, type a marketing source name or click **Go** to search for a marketing source name.
10. From the Outcome field, select an outcome or click **Go** to search for an outcome.

    If you select an outcome that requires a result, then you must select a result from the Result field.
11. Optionally, from the Result field, type a result or click **Go** to search for a result.

    If you select an result that requires a reason, then you must select a reason from the Reason field.
12. Optionally, from the Reason field, type a reason or click **Go** to search for a reason.
13. In the Effective Start Date field, type or select the start date for the wrap up.
14. Optionally, in the Effective End Date field, type or select the end date for the wrap up.
15. In the Level field, select a wrap up level.

    You have the following options:

    - Interaction
    - Activity
    - Both (Default)

16. Click **Create**.

### See Also

- Section 5.6.2, "Updating a Wrap Up"
- Section 5.6.3, "Removing a Wrap Up"

## 5.6.2 Updating a Wrap Up

Use this procedure to update a wrap up.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History or Interaction History JSP Admin

### Prerequisites

- Create an outcome.
- Create a result.
- Create a reason.

### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Wrap Up**.

   The Wrap Ups page appears.
4. Click an ID hyperlink.

   The Edit Wrap Ups page appears.
5. Optionally, from the Marketing Source Code field, type a marketing source code or click **Go** to search for a marketing source code.
6. Optionally, in the Marketing Source Name field, type a marketing source name or click **Go** to search for a marketing source name.
7. From the Outcome field, select an outcome.

If you select an outcome that requires a result, then you must select a result from the Result field.

8. Optionally, from the Result field, type a result or click **Go** to search for a result.

   If you select an result that requires a reason, then you must select a reason from the Reason field.

9. Optionally, from the Reason field, type a reason or click **Go** to search for a reason.
10. In the Effective Start Date field, type or select the start date for the wrap up.
11. Optionally, in the Effective End Date field, type or select the end date for the wrap up.
12. In the Level field, select a wrap up level.

    You have the following options:

    - Interaction
    - Activity
    - Both

13. Click **Update**.

### See Also

- Section 5.6.1, "Creating a Wrap Up"
- Section 5.6.3, "Removing a Wrap Up"

## 5.6.3 Removing a Wrap Up

Use this procedure to delete a user-defined wrap up. You cannot delete seeded wrap ups.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History or Interaction History JSP Admin

### Prerequisites

None

### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Wrap Up**.

   The Wrap Ups page appears.
4. Select the **Select** box for the wrap up that you want to delete.

   You cannot delete seeded wrap ups.
5. Click **Delete**.

   The selected wrap up is removed from the list and the Wrap Ups page refreshes.

### See Also

Section 5.6.1, "Creating a Wrap Up"

## 5.7 Action-Activity Type

Use the Action-Activity Type hyperlink to maintain a list of action-activity type pair.

### Tasks

You can perform the following tasks:

- Section 5.7.1, "Creating an Action-Activity Type Pair"
- Section 5.7.2, "Updating an Action-Activity Type Pair"
- Section 5.7.3, "Removing an Action-Activity Type Pair"

### 5.7.1 Creating an Action-Activity Type Pair

Use this procedure to define an action-activity type pair.

#### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History or Interaction History JSP Admin

### Prerequisites

- Create an action.
- Create an activity type.
- If you want to define a default wrap up for the action-activity type pair, then create a wrap up.

### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Action-Activity Type**.

   The Action-Activity Type page appears.
4. Click **Create**.

   The Create Action-Activity Type page appears.
5. From the Action field, type an action or click **Go** to search for an action.
6. From the Activity Type field, type an activity type or click **Go** to search for an activity type.
7. Optionally, from the Default Wrap Up ID field, type a default wrap up or click **Go** to search for a default wrap up.
8. Click **Create**.

### See Also

- Section 5.7.2, "Updating an Action-Activity Type Pair"
- Section 5.7.3, "Removing an Action-Activity Type Pair"

## 5.7.2 Updating an Action-Activity Type Pair

Use this procedure to update an action-activity type pair.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History or Interaction History JSP Admin

### Prerequisites

- Create an action.
- Create an activity type.
- If you want to define a default wrap up for the action-activity type pair, then create a wrap up.

### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Action-Activity Type**.

   The Action-Activity Type page appears.
4. Click an action or activity type hyperlink.

   The Edit Action-Activity Type page appears.
5. From the Action field, select an action.
6. From the Activity Type field, select an activity type.
7. Optionally, from the Default Wrap Up ID field, select a default wrap up.
8. Click **Update**.

### See Also

- Section 5.7.1, "Creating an Action-Activity Type Pair"
- Section 5.7.3, "Removing an Action-Activity Type Pair"

## 5.7.3 Removing an Action-Activity Type Pair

Use this procedure to delete an action-activity type pair. You cannot delete seeded action-activity type pairs.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History or Interaction History JSP Admin

### Prerequisites

None

### Steps

1. Click the Interaction History tab.
2. Click the Configuration subtab.
3. In the side panel, click **Action-Activity Type**.

   The Action-Activity Type page appears.
4. Select the **Select** box for the action-activity type pair that you want to delete.

   You cannot delete seeded action-activity type pairs.
5. Click **Delete**.

   The selected action-activity type is removed from the list and the Action-Activity Type page refreshes.

### See Also

- Section 5.7.1, "Creating an Action-Activity Type Pair"
- Section 5.7.2, "Updating an Action-Activity Type Pair"

# 6

# Using Oracle Customer Interaction History

This chapter describes task-based procedures for using Oracle Customer Interaction History. Topics include:

- Section 6.1, "Interactions"
- Section 6.2, "Activities"

## 6.1 Interactions

Use the Interaction History tab to search for customer interactions.

### Tasks

You can perform the following tasks:

- Section 6.1.1, "Viewing Interactions (HTML)"
- Section 6.1.2, "Searching for Interactions (HTML)"
- Section 6.1.3, "Viewing Interactions (Self Service)"
- Section 6.1.4, "Filtering Interactions (Self Service)"
- Section 6.1.5, "Viewing Interactions (Forms)"
- Section 6.1.6, "Filtering Interactions and Activities (Forms)"

### 6.1.1 Viewing Interactions (HTML)

The end-user interfaces for Oracle Customer Interaction History display the interactions for a specific party or account. Typically, a user specifies the party or account in the business application that is calling Oracle Customer Interaction History. However, when you access interactions directly from the Oracle Customer

Interaction History user interface, you must specify the party or account using a "test page."

Use the following procedure to access customer interactions directly from the Oracle Customer Interaction History user interface.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History JSP User

### Prerequisites

None

### Steps

1. In the Context Value section, do one of the following:
   - In the Party Name field, type or search for a party name.
   - In the Account Number field, type or search for an account number.
2. Optionally, in the Initial Search Value area, in the Search Category field select a parameter type for narrowing your search results and in the Search Value field enter a value for the search parameter.
3. Click **Go**.

   The Interaction History Viewer page appears.
4. To view interactions, click the **Interactions** hyperlink in the side panel.
5. To view activities, click the **Activities** hyperlink in the side panel.

### See Also

- Section 6.1.2, "Searching for Interactions (HTML)"
- Section 6.2.1, "Viewing Activities (HTML)"
- Section 6.2.2, "Searching for Activities (HTML)"

## 6.1.2 Searching for Interactions (HTML)

Use the following procedure to search for customer interactions.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History JSP User

### Prerequisites

Select a party or account. Typically, a user specifies the party or account in the business application that is calling Oracle Customer Interaction History. However, when you access interactions directly from the Oracle Customer Interaction History user interface, you must specify the party or account using a "test page." (See Section 6.1.1, "Viewing Interactions (HTML)")

### Steps

1. Click the **Advanced Search** hyperlink.

   The Interaction Filtering Criteria page appears.

2. Select the search criteria:

   You may search for a criterion using character strings and the wildcard symbol (%). For example, to find customers starting with A, type A%.

   a. In the Customer field, type a customer name or click **Go** to search for a customer name.

   b. In the Agent field, type a agent name or click **Go** to search for a agent name.

   c. In the Campaign field, type a campaign name or click **Go** to search for a campaign.

   d. In the Account field, type an account name or click **Go** to search for an account.

   e. In the Date fields, select a start date and end date.

   If you want to clear the search criteria, then click **Clear**.

3. Click **Apply**.

### See Also

Section 6.1.1, "Viewing Interactions (HTML)"

## 6.1.3 Viewing Interactions (Self Service)

The end-user interfaces for Oracle Customer Interaction History display the interactions for a specific party or account. Typically, a user specifies the party or account in the business application that is calling Oracle Customer Interaction History. However, when you access interactions directly from the Oracle Customer Interaction History user interface, you must specify the party or account using a "test page."

Use the following procedure to access customer interactions directly from the Oracle Customer Interaction History user interface.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History Self Service

### Prerequisites

None

### Steps

1. In the Context Values section, do one of the following:
   - In the Account Number field, enter an account number.
   - In the Party Name field, enter a party name.
2. Optionally, in the Search field select a parameter type for narrowing your search results and in the Value field enter a value for the search parameter.
3. Click **Go**.

   The Customer History page appears.

### See Also

Section 6.1.4, "Filtering Interactions (Self Service)"

## 6.1.4 Filtering Interactions (Self Service)

The Customer History region list the interactions for a customer. Use the following procedure to display a subset of the interactions based on selected search parameters.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History Self Service

### Prerequisites

Select a party or account. Typically, a user specifies the party or account in the business application that is calling Oracle Customer Interaction History. However, when you access interactions directly from the Oracle Customer Interaction History user interface, you must specify the party or account using a "test page." (See Section 6.1.3, "Viewing Interactions (Self Service)")

### Steps

1. To perform a simple search, do the following:
   a. From the Search list, select a type of search parameter.
   b. In the Value field, type a value for the selected search parameter.

      You may search for a criterion using character strings and the wildcard symbol (%). For example, to find outcomes that start with A, type A%.
   c. Click **Go**.
2. To perform an advanced search, do the following.
   a. Click the **Advanced Search** hyperlink.
   b. To display interactions that match all of the search parameters, select **Search results where each must contain all values entered**.
   c. To display interactions that match any of the search parameters, select **Search results where each may contain any value entered**.
   d. Optionally, in the Start Date field, select the match criteria and enter a start date.
   e. Optionally, in the End Date field, select the match criteria and enter an end date.
   f. Optionally, in the Resource Name field, select the match criteria and enter a resource name.
   g. Optionally, in the Interaction Type field, select the match criteria and enter an interaction type.

h. If you want to add an additional search parameter, then select a parameter type from the Add Another field, click Add, and enter a value for the parameter.

i. Click **Go**.

### See Also

Section 6.1.3, "Viewing Interactions (Self Service)"

## 6.1.5 Viewing Interactions (Forms)

The end-user interfaces for Oracle Customer Interaction History display the interactions for a specific party or account. Typically, a user specifies the party or account in the business application that is calling Oracle Customer Interaction History. However, when you access interactions directly from the Oracle Customer Interaction History user interface, you must specify the party or account using a "test page."

Use the following procedure to access customer interactions directly from the Oracle Customer Interaction History user interface.

### Login

E-Business Home Page

### Responsibility

Interaction History

### Prerequisites

None

### Steps

1. Select **Interaction History Form**.

   The Customer Interaction History Test page appears.

2. In the Context Values section, do one of the following:

   - In the Customer field, type or search for a party name.
   - In the Account field, type or search for an account number.

3. Optionally, in the Contact field, enter a contact to narrow your search results by contact.

4. Click **Continue**.

   The Customer Interaction History window appears.

> **Note:** The Reuse Results Window box is used to reapply the search criteria without having to open a new test page. The Reuse Results Window box must be selected prior to opening the first instance of the form. Therefore, to enable this feature, you must first select the Reuse Results Window box, close the test page, and then reopen the test page.

### See Also

Section 6.1.6, "Filtering Interactions and Activities (Forms)"

## 6.1.6 Filtering Interactions and Activities (Forms)

The Customer Interaction History form list the interactions and activities for a customer. Use the following procedure to display a subset of the interactions or activities based on selected search parameters.

### Login

E-Business Home Page

### Responsibility

Interaction History

### Prerequisites

Select a party or account. Typically, a user specifies the party or account in the business application that is calling Oracle Customer Interaction History. However, when you access interactions directly from the Oracle Customer Interaction History user interface, you must specify the party or account using a "test page." (See Section 6.1.5, "Viewing Interactions (Forms)")

### Steps

1. To filter interactions, select the Interaction Filtering Criteria tab and then select the search criteria:

   You may search for a criterion using character strings and the wildcard symbol (%). For example, to find customers starting with A, type A%.

a. Optionally, in the Agent field, select a agent name.

b. Optionally, in the Activity Type field, select an activity type for the activities that you want to display.

c. Optionally, in the Media Type field, select a media type for the activities that you want to display.

d. Optionally, in the Account field, type an account for the activities that you want to display.

e. Optionally, in the Application field, select the application that created the activities that you want to display.

f. Optionally, in the Direction field, select the direction of the media for the activities that you want to display.

g. Optionally, in the Contact field, select a contact party.

h. Optionally, in the Start Date field, select the start date of the activities that you want to display.

i. Optionally, in the End Date field, select the end date of the interactions that you want to display.

j. If you want to clear the search criteria, then click **Clear**.

k. Click **Search**.

1. To filter activities, select the Activities Filtering Criteria tab and then select the search criteria:

   You may search for a criterion using character strings and the wildcard symbol (%). For example, to find customers starting with A, type A%.

a. Optionally, in the Agent field, select a agent name.

b. Optionally, in the Source Code field, select a campaign source code.

c. Optionally, in the Account field, type an account name.

d. Optionally, in the Contact field, select a contact party.

e. Optionally, in the Start Date field, select the start date of the interactions that you want to display.

f. Optionally, in the End Date field, select the end date of the interactions that you want to display.

   If you want to clear the search criteria, then click **Clear**.

g. Click **Search**.

### See Also

Section 6.1.5, "Viewing Interactions (Forms)"

## 6.2 Activities

Use the Activities hyperlink to search for customer activities.

### Tasks

You can perform the following tasks:

- Section 6.2.1, "Viewing Activities (HTML)"
- Section 6.2.2, "Searching for Activities (HTML)"

### 6.2.1 Viewing Activities (HTML)

Use the following procedure to view customer activities.

#### Login

E-Business Home Page or CRM Home Page

#### Responsibility

Interaction History JSP User

#### Prerequisites

None

#### Steps

1. Click the Activities tab.
2. To narrow the search, select the search criteria:
   a. From the Media Type list, select a type of media.
   b. From the Source list, select a source.
   c. From the Activity list, select an activity type.
   d. In the Start Date field, select a start date.

e. In the End Date field, select an end date.

3. Click **Apply**.

### See Also

Section 6.2.2, "Searching for Activities (HTML)"

## 6.2.2 Searching for Activities (HTML)

Use the following procedure to search for customer activities.

### Login

E-Business Home Page or CRM Home Page

### Responsibility

Interaction History JSP User

### Prerequisites

None

### Steps

1. Click the Activities tab.
2. Click the **Advanced Search** hyperlink.
3. Select the search criteria:

   You may search for a criterion using character strings and the wildcard symbol (%). For example, to find customers starting with A, type A%.

   a. In the Customer field, type a customer name or click **Go** to search for a customer name.

   b. In the Agent field, type a agent name or click **Go** to search for a agent name.

   c. In the Media Type field, type a media type name or click **Go** to search for a media type.

   d. In the Activity field, type an activity type name or click **Go** to search for an activity type.

   e. In the Direction field, type a direction or click **Go** to search for a direction.

   f. In the Handler field, type a handler or click **Go** to search for a handler.

g. In the Date fields, select a start date and end date.

   If you want to clear the search criteria, then click **Clear**.

4. Click **Apply**.

## See Also

Section 6.2.1, "Viewing Activities (HTML)"

# A

# Data Migration

The outcome-result and result-reason pairs are obsolete. These data pairs are replaced by the wrap-up.

A wrap up relates outcomes to results and reasons. You can limit the availability of a wrap up to a specific campaign type or code. When the wrap up is selected for an interaction, the outcome, result, and reason in the wrap up definition are assigned to the interaction.

If you have existing outcome-result and result-reason pairs, you will need to migrate the data to the wrap-up table. You can migrate data by using the data migration function or by manually creating wrap-ups.

This appendix covers the migration of existing outcome-result pairs and result-reason pair to wrap-ups using the data migration function. It includes the following topics:



## A.1 Understanding Data Migration

**Note:** In release 11.5.9, a script (JTFIHWRPCMP.sql) was provided for migrating outcome-result and result-reason pairs to wrap-ups. Do not use this script for 11.5.10. Use the functions provided by the Interaction History Migration responsibility.

The data migration function populates wrap-ups based on existing values for outcomes-result pairs and result-reason pairs. You can migrate data by using the data migration function or by manually creating wrap-ups.

The following rules are applied during the data migration in order to determine if a wrap-up should be created:

1. The following tables are scanned:
   a. Outcomes
   b. Outcome-Result
   c. Result-Reason
   d. Interaction History
   e. Activity History
2. If an outcome requires a result and a result has not been associated with the outcome, then a wrap-up is not created.
3. If a result requires a reason and a reason has not been associated with the result, then a wrap-up is not created.
4. If a linked outcome, result, and reason is set to inactive, then a wrap-up is created and the end date is set o the current date and time.
5. If a linked outcome, result, and reason is associated with a marketing campaign, then a wrap-up is created for the campaign.

### See Also

- Concepts
- Creating a Wrap Up
- Section A.2, "Using Interaction History Migration"
- Section A.3, "Outcome-Result Administration"
- Section A.4, "Result-Reason Administration"

## A.2 Using Interaction History Migration

Use this procedure to migrate outcome-result pairs and result-reason pairs to wrap-ups.

### Login

HTML Login URL

### Responsibility

Interaction History Migration

### Prerequisites

---

**Note:** The Active box in outcome, result, and reason definition is used to hide the item from a list of values. The definition can still be used in the background to log an interaction. Use this feature to reduce the number of items that appear in a list of values.

For example, the Sit Reorder outcome is used by the predictive dialer in Oracle Advanced Outbound Telephony to log interactions. However, that outcome would not be needed in a list of values for agents.

---

1. Review all outcome definitions.
   - If you do not want the outcome to appear in a list of values, clear the Active box.
   - If you want to require a result with the outcome, then select the Result Required box.

     If a result is required and the data migration function does not find an outcome-result pair, then a wrap-up will not be created.
2. Review all result definitions.
   - If you do not want the result to appear in a list of values, clear the Active box.
   - If you want to require a reason with the outcome, then select the Reason Required box.

     If a reason is required and the data migration function does not find an result-reason pair, then a wrap-up will not be created.
3. Review all reason definitions.
   - If you do not want the result to appear in a list of values, clear the Active box.

### Steps

1. Click the Interaction History tab.
2. Click the Migration Configuration subtab.
3. In the side panel, click **Wrap-up Migration**.
4. Review the list of wrap-ups create by the data migration function. Remove any unwanted wrap-ups.

### See Also

- Section A.1, "Understanding Data Migration"
- Section A.3, "Outcome-Result Administration"
- Section A.4, "Result-Reason Administration"

## A.3 Outcome-Result Administration

The outcome-result pairs are obsolete. However, outcome-result administration is still available through the Interaction History Migration responsibility to facilitate migration of the data to the wrap-up table.

Use the Outcome-Result hyperlink to maintain a list of outcome-result pairs.

### Tasks

You can perform the following tasks:

- Section A.3.1, "Creating an Outcome-Result Pair"
- Section A.3.2, "Removing an Outcome-Result Pair"

### See Also

- Section A.1, "Understanding Data Migration"
- Section A.2, "Using Interaction History Migration"
- Section A.4, "Result-Reason Administration"

### A.3.1 Creating an Outcome-Result Pair

Use this procedure to define a outcome-result pair.

### Login
HTML Login URL

### Responsibility
Interaction History Migration

### Prerequisites
- Create an outcome.
- Create a result.

### Steps
1. Click the Interaction History tab.
2. Click the Migration Configuration subtab.
3. In the side panel, click **Outcome-Result**.

   The Outcome-Result page appears.
4. Click **Create**.

   The Create Outcome-Result window opens.
5. From the Outcome list, select an outcome.
6. From the Result list, select a result.
7. Click **Create**.

### See Also
Section A.3.2, "Removing an Outcome-Result Pair"

## A.3.2 Removing an Outcome-Result Pair
Use this procedure to delete a user-defined outcome-result pair.

### Login
HTML Login URL

### Responsibility
Interaction History Migration

### Prerequisites

None

### Steps

1. Click the Interaction History tab.
2. Click the Migration Configuration subtab.
3. In the side panel, click **Outcome-Result**.

   The Outcome-Result page appears.
4. Select the **Select** box for the outcome-result pair that you want to delete.
5. Click **Delete**.

   The selected outcome-result pair is removed from the list and the Outcome-Result page refreshes.

### See Also

Section A.3.1, "Creating an Outcome-Result Pair"

## A.4 Result-Reason Administration

The result-reason pairs are obsolete. However, result-reason administration is still available through the Interaction History Migration responsibility to facilitate migration of the data to the wrap-up table.

Use the Result-Reason hyperlink to maintain a list of result-reason pairs.

> **Note:** Migration of existing result-reason mappings to wrap up mappings can be performed using the script JTFIHWRPCMP.sql.

### Tasks

You can perform the following tasks:

- Creating a Result-Reason Pair
- Section A.4.1, "Creating a Result-Reason Pair"
- Section A.4.2, "Removing a Result-Reason Pair"

### See Also

- Section A.1, "Understanding Data Migration"
- Section A.2, "Using Interaction History Migration"
- Section A.3, "Outcome-Result Administration"

## A.4.1 Creating a Result-Reason Pair

Use this procedure to define a result-reason pair.

### Login

HTML Login URL

### Responsibility

Interaction History Migration

### Prerequisites

- Create a result.
- Create a reason.

### Steps

1. Click the Interaction History tab.
2. Click the Migration Configuration subtab.
3. In the side panel, click **Result-Reason**.

   The Result-Reason page appears.
4. Click **Create**.

   The Create Result-Reason window opens.
5. From the Result list, select a result.
6. From the Reason list, select a reason.
7. Click **Create**.

### See Also

Section A.4.2, "Removing a Result-Reason Pair"

## A.4.2 Removing a Result-Reason Pair

Use this procedure to delete a user-defined reason-result pair.

### Login

HTML Login URL

### Responsibility

Interaction History Migration

### Prerequisites

None

### Steps

1. Click the Interaction History tab.
2. Click the Migration Configuration subtab.
3. In the side panel, click **Result-Reason**.

   The Result-Reasons page appears.
4. Select the **Select** box for the result-reason pair that you want to delete.
5. Click **Delete**.

   The selected result-reason pair is removed from the list and the Result-Reasons page refreshes.

### See Also

- Creating a Result-Reason Pair
- Section A.4.1, "Creating a Result-Reason Pair"

# B

# Concurrent Programs

This chapter covers the following topics:

- Section B.1, "Interaction History Bulk Processor"
- Section B.2, "Interaction History Import"
- Section B.3, "Interaction History Purge"

## B.1 Interaction History Bulk Processor

The Interaction History Bulk Processor concurrent program is used to log interaction records to Oracle Customer Interaction History tables in bulk, rather than after each interaction, following a large batch of interactions (such as a mass e-mail campaign). This program can be run as needed or set up to run periodically.

> **Note:** The Oracle Customer Interaction History Bulk API is currently used by Oracle One-to-One Fulfillment. For information about how to implement Oracle One-to-One Fulfillment, please see the *Oracle One-to-One Fulfillment Implementation Guide*.

This section covers the following topics:

- Scheduling the Interaction History Bulk Processor Concurrent Program
- Using the Error Viewer

### See Also

- Section B.2, "Interaction History Import"
- Section B.3, "Interaction History Purge"

## B.1.1 Scheduling the Interaction History Bulk Processor Concurrent Program

Use this procedure to schedule the Interaction History Bulk Processor concurrent program.

### Responsibility

CRM Administrator

### Login

E-Business Home Page

### Prerequisites

None

### Steps

1. Select **Requests > Run**.

   The Submit a New Request window appears.

2. Select **Single Request** and click **OK**.

3. From the Name list, select Interaction History Bulk Processor.

   **Note:** There are no parameters for the Interaction History Bulk Processor concurrent process.

4. To set the job frequency, click **Schedule**.

5. In the Submit Request window, click **Submit**.

### Guidelines

For more information about submitting concurrent requests, including defining a submission schedule, see *Oracle Applications User's Guide*.

## B.1.2 Using the Error Viewer

Use this procedure to view logging errors that occurred during the Interaction History Bulk Processor concurrent program.

### Responsibility

Interaction History or Interaction History JSP Admin

### Login

E-Business Home Page or CRM Home Page

### Prerequisites

Run the Interaction History Bulk Processor concurrent program.

### Steps

1. Click the Bulk Processing Errors tab.
2. To filter the list of errors, select the filter criteria.

   You have the following options:

   - Writer Code
   - Request Type
   - Request Id

## B.2 Interaction History Import

The Interaction History Import concurrent program is used to import interaction data from third-party and legacy systems. This program can be run as needed or set up to run periodically.

> **Note:** This concurrent program references PL/SQL procedure JTF_IH_IMPORT_GO_IMPORT.

To import data into the Oracle Customer Interaction History tables, do the following:

1. Verify the format of the data to be transferred. See Validating Data.
2. Use your tool of choice to move the data into the Oracle Customer Interaction History staging tables. See Loading the Data into Staging Tables.
3. Use the Interaction History Import concurrent program to move the data from the staging tables to the Oracle Customer Interaction History tables. See Scheduling the Interaction History Data Import Concurrent Program.

4. Delete the data in the Oracle Customer Interaction History staging tables. See Deleting the Staging Table Rows.

### See Also

- Section B.1, "Interaction History Bulk Processor"
- Section B.3, "Interaction History Purge"

## B.2.1 Validating Data

You can use standard SQL*Plus database commands to view details about the Oracle Customer Interaction History staging tables. Verify that the data to be transferred conforms to the format of the data expected in the staging tables.

The Oracle Customer Interaction History staging tables are:

- JTF_IH_INTERACTIONS_STG
- JTF_IH_ACTIVITIES_STG
- JTF_IH_MEDIA_ITEMS_STG

The Oracle Customer Interaction History staging tables are similar to the actual Oracle Customer Interaction History tables. However, the staging tables have three additional columns that hold data about the import process. The additional columns are:

- SESSION_NO: This is the sequential number of the import session.
- STATUS_FL: This is a flag that indicates state of a record after the import process. Values are:
  - Null - The record is new and has not been processed by the import program.
  - 0 - The record produced an error during the import process.
  - 1 - The record passed the referential integrity checks.
  - 2 - The record was successfully imported.
- SESSION_DATE: The is the date that the record was imported.

For information about Oracle Customer Interaction History tables, see Appendix D, "Data Validations".

## B.2.2 Loading the Data into Staging Tables

The method for loading the data into the staging tables will depend on the third party or legacy system with which you are working. You must have read and write access to the Oracle Customer Interaction History staging tables in order to complete this task.

Optionally, you can test that the data has been loaded properly in the staging tables by running stored procedure JTF_IH_IMPORT.GO_TEST. This procedure tests the data in staging tables without writing the data into the Oracle Customer Interaction History tables.

## B.2.3 Scheduling the Interaction History Data Import Concurrent Program

**Note:** If you encounter problems moving the data from the staging tables to the Interaction History tables by running the Concurrent Manager, you can start the Interaction History mass import procedure from SQL*Plus by calling the stored procedure JTF_IH_IMPORT.GO_IMPORT.

There is one optional parameter to this procedure: NCntTransRows (NUMBER) - the number of interactions in one database transaction.

Use this procedure to schedule the Interaction History Import concurrent program.

### Prerequisites

Load the data to be imported into the Oracle Customer Interaction History staging tables. See Loading the Data into Staging Tables.

### Responsibility

CRM Administrator or Interaction History Data Import

### Steps

1. Select **Requests > Run**.

   The Submit a New Request window opens.

2. Select the **Single Request** and click **OK**.

   The Submit Request window opens.

3. From the Name list, select Interaction History Data Import.

**Note:** There are no parameters for the Interaction History Data Import concurrent process.

4. To set the job frequency, click **Schedule**.
5. In the Submit Request window, click **Submit**.

### Guidelines

For more information about submitting concurrent requests, including defining a submission schedule, see *Oracle Applications User's Guide*.

## B.2.4 Deleting the Staging Table Rows

If you decide to reload data or to load additional data into the staging tables, you must first delete the current data before adding new data. You can use standard SQL*Plus database commands to delete the existing rows.

# B.3 Interaction History Purge

The Interaction History Purge concurrent program is used to remove interaction data based on a set of purge criteria. This program can be run as needed or set up to run periodically.

**Warning: There are historical reports provided with Oracle Interaction Center Intelligence that rely on Oracle Customer Interaction History data. In addition, purging interaction data can affect functionality in Oracle Email Center and Oracle Marketing. Use the Interaction History Purge concurrent program with great care. Review your historical reporting needs before making any decision to purge data.**

Use this procedure to schedule the Interaction History Purge concurrent program.

### Responsibility

Interaction History Data Purge

### Login

E-Business Home Page

### Prerequisites

None

### Steps

1. Select (Interaction History Data Purge) **Requests**.

   The Find Requests window appears.

2. Click **Submit A New Request**.

   The Submit a New Request window appears.

3. Select **Single Request** and click **OK**.

4. From the Name list, select Interaction History Data Purge.

   The Parameters windows appears.

   > **Note:** You can re-open the Parameters window by clicking in the Parameters field.

5. Enter the criteria for selecting the interaction data to be deleted.

6. Click **OK**.

7. To set the job frequency, click **Schedule**.

8. In the Submit Request window, click **Submit**.

### Guidelines

For more information about submitting concurrent requests, including defining a submission schedule, see *Oracle Applications User's Guide*.

### Parameters

Use these parameters to select that data that you want to purge. You must set at least one parameter. If you set more than one parameter, the program will select only the data that meets **all** of the selected parameters. To purge all interactions, set the Purge Type to ALL.

| Parameter | Required? | Definition |
|---|---|---|
| Party IDs | No | Specify a list of comma-separated numbers that correspond to the party_ id(s) that you wish to purge. If a single party id is to be purged, then no commas are required. |
| Party Type | No | Specify the Party Type of the parties associated with interactions to be purged. Valid Values: PERSON ORGANIZATION PARTY_ RELATIONSHIP |
| Start Date | No | The interaction start date from which to purge. Interactions with a start date greater than or equal to this value will be purged. |
| End Date | No | The interaction end date to purge up to. Interactions with an end date less than or equal to this value will be purged. |
| Safe Mode | Yes | Allows the purge to be run in a report only mode. TRUE (default) - No data deleted. Will report the number of Interactions, Activities and Media Items to be purged. No records are purged.FALSE - Data is deleted. |
| Purge Type | No | Specify 'ALL' to purge all interactions. When specifying other criteria (party ids, party type, start date and end date), leave this value empty. |
| Activity Outcomes | No | Specify a list of comma-separated numbers that correspond to the activity outcome id(s) that you wish to purge. If a single outcome id is to be purged, then no commas are required. If an activity is found associated with an interaction that has one of the outcome ids specified, then the interaction and all of it's activities are purged. |
| Activity Results | No | Specify a list of comma-separated numbers that correspond to the activity result id(s) that you wish to purge. If a single result id is to be purged, then no commas are required. If an activity is found associated with an interaction that has one of the result ids specified, then that interaction and all of it's activities are purged. |
| Activity Reasons | No | Specify a list of comma-separated numbers that correspond to the activity reason id(s) that you wish to purge. If a single result id is to be purged, then no commas are required. If an activity is found associated with an interaction that has one of the result ids specified, then that interaction and all of its activities are purged. |
| Interaction Outcomes | No | Specify a list of comma-separated numbers that correspond to the interaction outcome id(s) that you wish to purge. If a single outcome id is to be purged, then no commas are required. |
| Interaction Results | No | Specify a list of comma-separated numbers that correspond to the interaction result id(s) that you wish to purge. If a single result id is to be purged, then no commas are required. |
| Interaction Reasons | No | Specify a list of comma-separated numbers that correspond to the interaction reason id(s) that you wish to purge. If a single result id is to be purged, then no commas are required. |

### See Also

- Section B.1, "Interaction History Bulk Processor"
- Section B.2, "Interaction History Import"

# C

# API Reference

Oracle Applications contains the following types of APIs:

- **Private APIs** are for internal, development use only. Details are not provided to anyone outside of the immediate development environment, nor are they intended for use by anyone outside of the Oracle Applications development environment.
- **Public APIs** are designed for customers and Oracle consultants to integrate non-Oracle systems into Oracle Applications or to extend the functionality of the base products. Oracle does not support public APIs unless they are published in a reference manual such as this one. The user accepts all risk and responsibility for working with non-published public APIs.
- **Public, published APIs** are guaranteed by Oracle to remain valid from release to release and that patches will not alter the API behavior. Public, published APIs are supported by Oracle to the same extent as released software.

For non-published APIs, Oracle expressly does not provide any guarantees regarding consistency of naming, usage, or behavior of any API (public or private) between releases. It is also possible that a patch could alter any characteristic of any non-published CRM API. As such, those who choose to use these APIs do so at their own risk. However, Oracle does attempt to minimize all changes to public APIs, even if not published.

Each published API provides an API specification, and definitions as for its parameters, data structures, and status messages. Sample scripts and documented process flow diagrams are included where applicable.

> **Note:** The words *procedure* and *API* are used interchangeably in this document.

# C.1 Parameter Specifications

The specifications for the public APIs provided by Oracle Customer Interaction History define four categories of parameters:

- Standard IN
- Standard OUT
- Procedure specific IN
- Procedure specific OUT

Standard IN and OUT parameters are specified by the Oracle Applications business object API Coding Standards, and are discussed in the following sections.

Procedure specific IN and OUT parameter are related to the API being specified, and are discussed with that individual API.

## C.1.1 Standard IN Parameters

The following table describes standard IN parameters, which are common to all public APIs provided by Oracle Customer Interaction History.

***Table C–1 Standard IN Parameters***

| Parameter | Data Type | Required | Description |
|---|---|---|---|
| p_api_version | NUMBER | Yes | This must match the version number of the API. An unexpected error is returned if the calling program version number is incompatible with the current API version number (provided in the documentation). |

***Table C–1 Standard IN Parameters***

| Parameter | Data Type | Required | Description |
|---|---|---|---|
| p_init_msg_list | VARCHAR2 | Yes | The valid values for this parameter are:<br>■ True = FND_API.G_TRUE<br>■ False = FND_API.G_FALSE<br>■ Default = FND_API.G_FALSE<br>If set to true, then the API makes a call to *fnd_msg_pub.initialize* to initialize the message stack. To set to true, use the value, "T".<br>If set to false then the calling program must initialize the message stack. This action is required to be performed only once, even in the case where more than one API is called. To set to false, use the value, "F". |
| p_commit | VARCHAR2(1) | No | The valid values for this parameter are:<br>■ True = FND_API.G_TRUE<br>■ False = FND_API.G_FALSE<br>■ Default = FND_API.G_FALSE<br>If set to true, then the API commits before returning to the calling program. To set to true, use the value, "T".<br>If set to false, then it is the calling program's responsibility to commit the transaction. To set to false, use the value, "F". |

## C.1.2 Standard OUT Parameters

The following table describes standard OUT parameters, which are common to all public APIs provided by Oracle Customer Interaction History.

**Note:** All standard OUT parameters are required.

*Table C–2 Standard OUT Parameters*

| Parameter | Data Type | Description |
|---|---|---|
| x_return_status | VARCHAR2(1) | Indicates the return status of the API. The values returned are one of the following:<br>■ FND_API.G_RET_STS_SUCCESS<br>Success: Indicates the API call was successful<br>■ FND_API.G_RET_STS_ERROR<br>Expected Error: There is a validation error, or missing data error.<br>■ FND_API.G_RET_STS_UNEXP_ERROR<br>Unexpected Error: The calling program can not correct the error. |
| x_msg_count | NUMBER | Holds the number of messages in the message list. |
| x_msg_data | VARCHAR2(2000) | Holds the encoded message if *x_msg_count* is equal to one. |

## C.1.3 Parameter Size

Verify the size of the column from the base table for that column when passing a parameter of a specific length. For example, if you pass a NUMBER value, first query to find the exact value to pass. An incorrect value can cause the API call to fail.

## C.1.4 Missing Parameter Attributes

The following table describes optional IN parameters which are initialized to pre-defined values representing missing constants. These constants are defined for the common PL/SQL data types and should be used in the initialization of the API formal parameters.

*Table C–3 Initialized IN Parameters*

| Parameter | Type | Initialized Value |
|---|---|---|
| g_miss_num | CONSTANT | NUMBER:= 9.99E125 |
| g_miss_char | CONSTANT | VARCHAR2(1):= chr(0) |
| g_miss_date | CONSTANT | DATE:= TO_DATE('1','j'); |

These constants are defined in the package FND_API in the file *fndpapis.pls*. All columns in a record definition are set to the G_MISS_X constant as defined for the data type.

### C.1.5 Parameter Validations

The following types of parameters are always validated during the API call:

- Standard IN
- Standard OUT
- Mandatory procedure specific IN
- Procedure specific OUT

### C.1.6 Invalid Parameters

If the API encounters any invalid parameters during the API call, then one of the following actions will occur:

- An exception is raised.
- An error message identifying the invalid parameter is generated.
- All API actions are cancelled.

## C.2 Version Information

It is mandatory that every API call pass a version number for that API as its first parameter (*p_api_version*).

This version number must match the internal version number of that API. An unexpected error is returned if the calling program version number is incompatible with the current API version number.

**Warning: The currently supported version at this time is 1.0. Use only this for the API version number.**

In addition, the object version number **must** be input for all update and delete APIs.

- If the *object_version_number* passed by the API matches that of the object in the database, then the update is completed.

- If the *object_version_number* passed by the API does not match that of the object in the database, then an error condition is generated.

## C.3 Status Messages

**Note:** It is not required that all status notifications provide a number identifier along with the message, although, in many cases, it is provided.

Every API must return one of the following states as parameter *x_return_status* after the API is called:

- S (Success)
- E (Error)
- U (Unexpected error)

Each state can be associated with a status message. The following table describes each state.

***Table C–4 Status Message and Description***

| Status | Description |
|---|---|
| S | Indicates that the API performed all the operations requested by its caller.<br>■ A success return status may or may not be accompanied by messages in the API message list.<br>■ Currently, the CRM Foundation APIs do not provide a message for a return status of success. |
| E | Indicates that the API failed to perform one or more of the operations requested by its caller.<br>An error return status is accompanied by one or more messages describing the error. |
| U | Indicates that the API encountered an error condition it did not expect, or could not handle, and that it is unable to continue with its regular processing.<br>■ For example, certain programming errors such as attempting to divide by zero causes this error.<br>■ These types of errors usually cannot be corrected by the user and requires a system administrator or application developer to correct. |

### Warning and Information Messages

In addition to these three types of possible status messages, you can also code the following additional message types:

- Warnings
- Information

To create a warning message, perform the following steps:

1. Create a global variable to be used to signal a warning condition. For example, this could be similar to the following:

   ```
   G_RET_STS_WARNING := 'W'
   ```

   This global variable is not part of the FND_API package.

2. Return this value if the warning condition is encountered. For example, using the same example as in step one, set up the following code in the API to process the warning condition:

   ```
   x_return_status := G_RET_STS_WARNING
   ```

   This code replaces the more usual:

   ```
   x_return_status := fnd_api.g_ret_sts_unexp_error for "U"
   ```

3. If desired, perform a similar procedure to create Information messages.

## C.4 APIS

Oracle Customer Interaction History provides other modules with a common framework for capturing and accessing all customer interaction data associated with customer contacts. Oracle Customer Interaction History provides a central repository for this data and includes APIs for tracking all automated or agent-based customer interactions.

The following topics are discussed in this chapter:

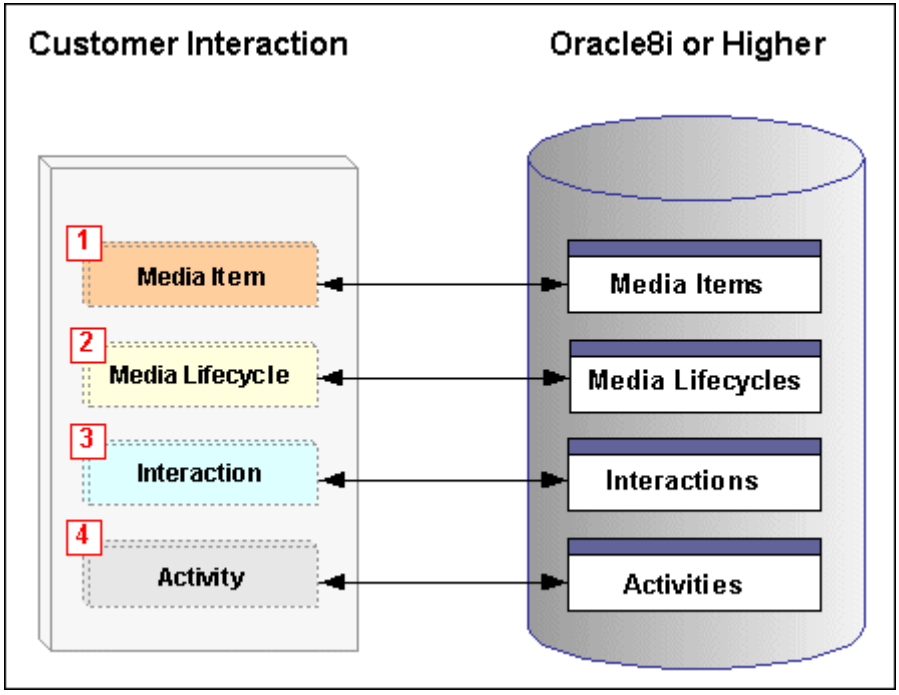
- Counting APIs
- Messages and Notifications
- Sample Code

## C.5 Customer Interaction

A customer interaction contains up to four unique units of customer information, a media item, a media life cycle, an activity, and an interaction. The following figure illustrates the different units of information that comprise a customer interaction and how they are stored in the Oracle database.

1. The customer interaction obtains its media item information from the Media Items table.
2. The customer interaction obtains its media life cycle information from the Media Life cycle table.
3. The customer interaction obtains its interaction information from the Interactions table.
4. The customer interaction obtains its activity information from the Media Items table.

*Figure C–1 Customer Interaction*




## C.5.1 Media Item

Media items are inbound and outbound communications that take place between a customer and a human or automated agent, a system or an application. One or more media items can be associated with a single activity. Telephone conversations and email correspondence between customers and agents are examples of media items.

Media are the individual communication channels through which media items are delivered. Telephones, fax machines, automatic teller machines (ATMs), and email messages are examples of media.

## C.5.2 Media Life Cycle

A Media Lifecycle is a unit of time associated with the handling of a media item. For example, if a customer call passes through four different phone queues for different periods of time contains four segments in its lifecycle.

## C.5.3 Activity

An activity is a business action performed by an agent as part of a customer interaction using one or more methods of communication called media items. Activities are recorded in Interaction History and can be viewed by using the Interaction History windows accessed from calling applications. Some examples of activities include an agent transferring a call, an agent emailing a marketing brochure, or a customer placing an order.

## C.5.4 Interaction

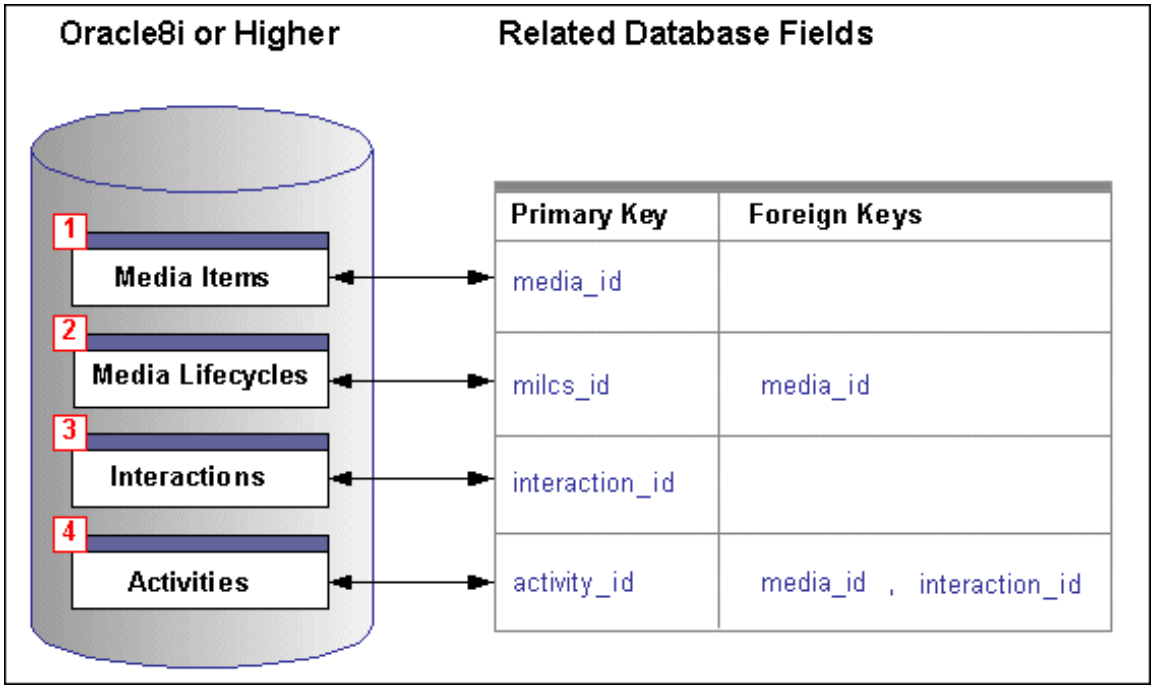
An interaction is a point of contact or touch point between a human or automated agent and a party such as a customer, a customer system, or a potential customer. An interaction is a timed entity with an outcome and a result that can be tracked. When an interaction is closed, it becomes an historical record that subsequently cannot be altered or modified. Multiple forms of communication or media items between the party and the agent can be included in a single interaction.

## C.5.5 Relating Customer Interaction Information

Interaction History applications must identify customer interaction information stored in different Oracle database tables as part of a single customer interaction. The following figure illustrates how the Interaction History database tables relate to each other using primary and foreign key values.

1. The Media Items table contains one primary key, **media_id**.
2. The Media Lifecycle table contains one primary key, **milcs_id** and one foreign key, **media_id**.
3. The Interactions table contains one primary key, **interaction_id**.
4. The Activities table contains one primary key, **activity_id** and two foreign keys, **media_id** and **interaction_id**.

*Figure C–2 Relating Customer Interaction Tables*




## C.6 Package JTF_IH_PUB

All public procedures (APIs) relating to media items, media lifecycles, interactions, and activities are stored in the JTF_IH_PUB package. This package contains three types of Oracle Customer Interaction History APIs:

- Non-cached Creation APIs
- Cached Creation APIs
- Counting APIs

### C.6.1 Data Structure Specifications

The Oracle Customer Interaction History APIs use the following data structures:

- Interaction Record Type
- Activity Record Type

- Media Item Record Type
- Media Item Lifecycle Record Type

### Nested Record Types

PL/SQL record types are used in all open, add, and close Interaction History APIs. In certain cases, nested record types are used as well.

For example, in the Create_Interaction API:

- Input parameter *p_activities* is a record of type *activity_tbl_type.*
- In turn, *activity_tbl_type* contains a record of type *activity_rec_type* as one of its elements.

Using nested data structures in this fashion enables the calling API to pass one or more activities to an Interaction History creation API.

### C.6.1.1 Interaction Record Type

This composite record type enumerates all the elements that represent an interaction record. This business entity represents a contact point between a customer, customer system, or potential customer and a single human or automated agent.

```
TYPE interaction_rec_type IS RECORD
(
  interaction_id              NUMBER          :=fnd_api.g_miss_num,
  reference_form              VARCHAR2(1000)  :=fnd_api.g_miss_char,
  follow_up_action            VARCHAR2(80)    :=fnd_api.g_miss_char,
  duration                    NUMBER          := fnd_api.g_miss_num,
  end_date_time               DATE            :=fnd_api.g_miss_date,
  inter_interaction_duration  NUMBER          :=fnd_api.g_miss_num,
  non_productive_time_amount  NUMBER          :=fnd_api.g_miss_num,
  preview_time_amount         NUMBER          :=fnd_api.g_miss_num,
  productive_time_amount      NUMBER          :=fnd_api.g_miss_num,
  start_date_time             DATE            :=fnd_api.g_miss_date,
  wrapUp_time_amount          NUMBER          :=fnd_api.g_miss_num,
  handler_id                  NUMBER          :=fnd_api.g_miss_num,
  script_id                   NUMBER          :=fnd_api.g_miss_num,
  outcome_id                  NUMBER          :=fnd_api.g_miss_num,
  result_id                   NUMBER          :=fnd_api.g_miss_num,
  reason_id                   NUMBER          :=fnd_api.g_miss_num,
  resource_id                 NUMBER          :=fnd_api.g_miss_num,
  party_id                    NUMBER          :=fnd_api.g_miss_num,
  parent_id                   NUMBER          :=fnd_api.g_miss_num,
```

```
  object_id                   NUMBER          :=fnd_api.g_miss_num,
  object_type                 VARCHAR2(30)    :=fnd_api.g_miss_char,
  source_code_id              NUMBER          :=fnd_api.g_miss_num,
  source_code                 VARCHAR2(100)   :=fnd_api.g_miss_char,
  attribute1                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute2                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute3                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute4                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute5                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute6                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute7                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute8                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute9                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute10                 VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute11                 VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute12                 VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute13                 VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute14                 VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute15                 VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute_category          VARCHAR2(30)    :=fnd_api.g_miss_char,
  touchpoint1_type            VARCHAR2(30)    := 'PARTY',
  touchpoint2_type            VARCHAR2(30)    := 'RS_EMPLOYEE',
  method_code                 VARCHAR2(30)    :=fnd_api.g_miss_char,
  bulk_writer_code            VARCHAR2(240)   := fnd_api.g_miss_char,
  bulk_batch_type             VARCHAR2(240)   := fnd_api.g_miss_char,
  bulk_batch_id               NUMBER          := fnd_api.g_miss_num,
  bulk_interaction_id         NUMBER          := fnd_api.g_miss_num,
  primary_party_id            NUMBER          := fnd_api.g_miss_num,
  contact_rel_party_id        NUMBER          := fnd_api.g_miss_num,
  contact_party_id            NUMBER          := fnd_api.g_miss_num
);
```

For validations performed on these record values, see Appendix D, "Data Validations".

### C.6.1.2 Activity Record Type

This composite record type enumerates all elements that represent an activity record. This business entity can be associated with the business functions performed during an interaction.

```
TYPE activity_rec_type IS RECORD
(
  activity_id                 NUMBER          := fnd_api.g_miss_num,
```

```
  duration                    NUMBER          :=fnd_api.g_miss_num,
  cust_account_id             NUMBER          := fnd_api.g_miss_num,
  cust_org_id                 NUMBER          := fnd_api.g_miss_num,
  role                        VARCHAR2(240)   := fnd_api.g_miss_char,
  end_date_time               DATE            :=fnd_api.g_miss_date,
  start_date_time             DATE            :=fnd_api.g_miss_date,
  task_id                     NUMBER          :=fnd_api.g_miss_num,
  doc_id                      NUMBER          :=fnd_api.g_miss_num,
  doc_ref                     VARCHAR2(30)    :=fnd_api.g_miss_char,
  doc_source_object_name      VARCHAR2(80)    :=fnd_api.g_miss_char,
  media_id                    NUMBER          :=fnd_api.g_miss_num,
  action_item_id              NUMBER          :=fnd_api.g_miss_num,
  interaction_id              NUMBER          :=fnd_api.g_miss_num,
  outcome_id                  NUMBER          :=fnd_api.g_miss_num,
  result_id                   NUMBER          :=fnd_api.g_miss_num,
  reason_id                   NUMBER          :=fnd_api.g_miss_num,
  description                 VARCHAR2(1000)  :=fnd_api.g_miss_char,
  action_id                   NUMBER          :=fnd_api.g_miss_num,
  interaction_action_type     VARCHAR2(240)   :=fnd_api.g_miss_char,
  object_id                   NUMBER          :=fnd_api.g_miss_num,
  object_type                 VARCHAR2(30)    :=fnd_api.g_miss_char,
  source_code_id              NUMBER          :=fnd_api.g_miss_num,
  source_code                 VARCHAR2(100)   :=fnd_api.g_miss_char,
    script_trans_id           NUMBER          :=fnd_api.g_miss_num,
  attribute1                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute2                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute3                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute4                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute5                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute6                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute7                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute8                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute9                  VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute10                 VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute11                 VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute12                 VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute13                 VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute14                 VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute15                 VARCHAR2(150)   :=fnd_api.g_miss_char,
  attribute_category          VARCHAR2(30)    :=fnd_api.g_miss_char,
  bulk_writer_code            VARCHAR2(240)   := fnd_api.g_miss_char,
  bulk_batch_type             VARCHAR2(240)   := fnd_api.g_miss_char,
  bulk_batch_id               NUMBER          := fnd_api.g_miss_num,
  bulk_interaction_id         NUMBER          := fnd_api.g_miss_num
);
```

For validations performed on these record values, see Appendix D, "Data Validations".

### C.6.1.3 Media Item Record Type

This composite record type enumerates all elements that represent a media record. This business entity can be generated by a customer, by the system, or an application.

```
TYPE media_rec_type IS RECORD
(
  media_id                      NUMBER          :=fnd_api.g_miss_num,
  source_id                     NUMBER          :=fnd_api.g_miss_num,
  direction                     VARCHAR2(240)   :=fnd_api.g_miss_char,
  duration                      NUMBER          :=fnd_api.g_miss_num,
  end_date_time                 DATE            :=fnd_api.g_miss_date,
  interaction_performed         VARCHAR2(240)   :=fnd_api.g_miss_char,
  start_date_time               DATE            :=fnd_api.g_miss_date,
  media_data                    VARCHAR2(80)    :=fnd_api.g_miss_char,
  source_item_create_date_time DATE             :=fnd_api.g_miss_date,
  source_item_id                NUMBER          :=fnd_api.g_miss_num,
  media_item_type               VARCHAR2(80)    :=fnd_api.g_miss_char,
  media_item_ref                VARCHAR2(240)   :=fnd_api.g_miss_char,
  media_abandon_flag            VARCHAR2(1)     :=fnd_api.g_miss_char,
  media_transferred_flag        VARCHAR2(1)     :=fnd_api.g_miss_char,
  server_group_id               NUMBER           :=fnd_api.g_miss_num,
  dnis                          VARCHAR2(30)    :=fnd_api.g_miss_char,
  ani                           VARCHAR2(30)    :=fnd_api.g_miss_char,
  classification                VARCHAR2(64)    :=fnd_api.g_miss_char,
  bulk_writer_code              VARCHAR2(240)   := fnd_api.g_miss_char,
  bulk_batch_type               VARCHAR2(240)   := fnd_api.g_miss_char,
  bulk_batch_id                 NUMBER          := fnd_api.g_miss_num,
  bulk_interaction_id           NUMBER          := fnd_api.g_miss_num,
  address                       VARCHAR2(2000)  := fnd_api.g_miss_char
);
```

For validations performed on these record values, see Appendix D, "Data Validations".

### C.6.1.4 Media Item Lifecycle Record Type

This composite record type enumerates all elements that represent a media lifecycle record. This business entity unit represents a unit of time associated with the handling of a media item.

```
TYPE media_lc_rec_type IS RECORD
(
  start_date_time             DATE            :=fnd_api.g_miss_date,
  type_type                   VARCHAR2(80)    :=fnd_api.g_miss_char,
  type_id                     NUMBER          :=fnd_api.g_miss_num,
  duration                    NUMBER          :=fnd_api.g_miss_num,
  end_date_time               DATE            :=fnd_api.g_miss_date,
  milcs_id                    NUMBER          :=fnd_api.g_miss_num,
  milcs_type_id               NUMBER          :=fnd_api.g_miss_num,
  media_id                    NUMBER          :=fnd_api.g_miss_num,
  handler_id                  NUMBER          :=fnd_api.g_miss_num,
  resource_id                 NUMBER          :=fnd_api.g_miss_num,
  milcs_code                  VARCHAR2(80)    := fnd_api.g_miss_char,
  bulk_writer_code            VARCHAR2(240)   := fnd_api.g_miss_char,
  bulk_batch_type             VARCHAR2(240)   := fnd_api.g_miss_char,
  bulk_batch_id               NUMBER          := fnd_api.g_miss_num,
  bulk_interaction_id         NUMBER          := fnd_api.g_miss_num
);
```

For validations performed on these record values, see Appendix D, "Data Validations".

## C.7 Non-cached Creation APIs

Applications with rapid transaction requirements, such as predictive dialing products, can take advantage of the following non-cached creation APIs:

- Create_Interaction
- Create_MediaItem
- Create_MediaLifecycle

### C.7.1 Overview

Non-cached creation APIs enable a client application to write and close an interaction in a single API call. This action is more efficient than that used for the cached APIs as the record is written and created in one transaction cycle. However, the client application must persist the interaction data during the creation of the interaction, or the entire interaction record can be lost if the data flow is interrupted.

If you use non-cached creation APIs for transactions and communication between the client application and the server is disrupted during creation of the record, then the entire transaction is lost.

*Table C–5 Interaction History Non-cached Creation APIs*

| Procedure | Description |
|---|---|
| Create_MediaItem | Creates a media item in the Media Items table. This procedure is optional.. |
| Create_MediaLifecycle | Creates a media lifecycle record in the Media Lifecycle table that is associated with a media item. This procedure is optional. |
| Create_Interaction | Creates an interaction record in the Interactions table and associates it with one or more activities in the Activities table. |

## C.7.2 Process Flow

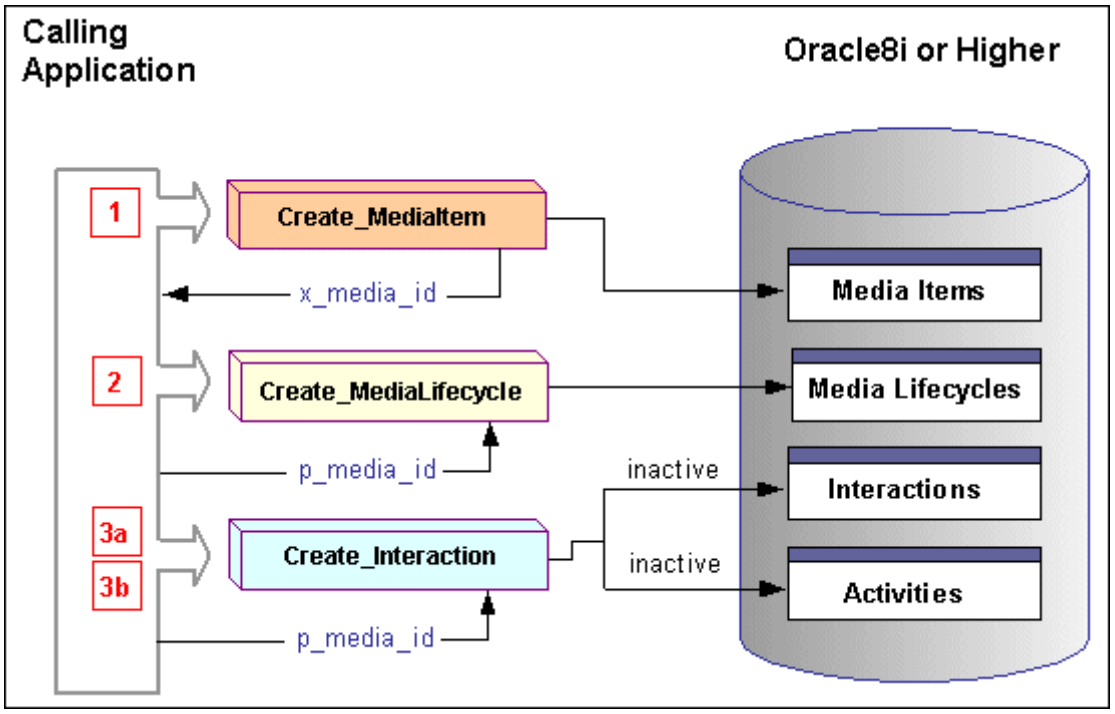
The following figure describes a common sequence of operations performed when a calling application invokes non-cached creation APIs. The process flows described in this figure are common but not required since some API calls are optional.

1. The Create_MediaItem API creates a media item in the Media Items table and passes a unique sequence generated identifier back to the calling application as **x_media_id**.
2. When the calling application invokes the Create_MediaLifecycle API, it passes the media item's unique identifier as **p_media_id**. The Create_MediaLifecycle API then creates a media lifecycle record in the Media Lifecycles table.

   **Note:** Media Item and Media Lifecycle calls are optional since interactions can exist without a media item or its associated media lifecycle record.

3. When the calling application invokes the Create_Interaction API, the following events occur:
   a. The calling application passes the media item's unique identifier as **p_media_id**. The Create_Interaction API then creates an interaction in the Interactions table and sets the interaction to inactive so that it can no longer be modified.
   b. The Create_Interaction API then creates one or more activities in the Activities table, associates the activity with the interaction, and sets the activity to inactive so that it can no longer be modified

*Figure C–3 Non-cached creation API Process Flow*




## C.7.3 Create_MediaItem

The Create_MediaItem API creates a media item in the Media Items table. This procedure is optional since an interaction can be created without any media items and their associated media lifecycles. Currently some applications create a media item and pass its media_id value to the calling application which subsequently passes it to the Create_Interaction API.

For example, media items are currently created by Advanced Outbound (Predictive Calls), Advanced Inbound (OTM), 1-to-1 Fulfillment, and E-mail Center. In the first two cases the media_id value is passed to the desktop application (Telesales, Customer Care, etc.) which enables these applications to associate the media item with the activities when they create an interaction and it's associated activities

### Procedure Specification

```
PROCEDURE create_mediaitem
(
```

```
   p_api_version   in      number,
   p_init_msg_list in      varchar2        default fnd_api.g_false,
   p_commit        in      varchar2        default fnd_api.g_false,
   p_resp_appl_id  in      number          default null,
   p_resp_id       in      number          default null,
   p_user_id       in      number,
   p_login_id      in      number          default null,
   x_return_status out     varchar2,
   x_msg_count     out     number,
   x_msg_data      out     varchar2,
   p_media         in      media_rec_type,
   p_mlcs          in      mlcs_tbl_type
);
```

### Current Version

```
1.0
```

### Parameter Descriptions

The Create_MediaItem API sets the value of the input parameter to NULL if the parameter corresponds to the value of the *G_MISS_X* constant. If the parameter does not correspond to this constant, then the API retains the passed-in value.

The following table describes the IN parameters associated with this API.

***Table C–6 Create Media Item IN Parameters***

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_api_version | NUMBER | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_init_msg_list | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_commit | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_resp_appl_id | NUMBER | No[1] | Application identifier |
| p_resp_id | NUMBER | No* | Responsibility identifier |
| p_user_id | NUMBER | No* | Corresponds to the column USER_ID in the table FND_USER, and identifies the Oracle Applications user. |

*Table C–6 Create Media Item IN Parameters*

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_login_id | NUMBER | No | Corresponds to the column LOGIN_ID in the table FND_LOGINS, and identifies the login session. |
| p_media | media_rec_type | Yes | See "Media Item Record Type" on page C-15. |
| p_mlcs | mlcs_tbl_type | Yes | See "Media Item Lifecycle Record Type" on page C-15. |

[1] The Application ID, Responsibility ID, and User ID determine which profile values are used as defaults. Those items marked with an asterisk also follow these guidelines.

The following table describes the OUT parameters associated with this API.

*Table C–7 Create Media Item OUT Parameters*

| Parameter | Data Type | Description |
|---|---|---|
| x_return_status | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_count | NUMBER | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_data | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |

## C.7.4 Create_MediaLifecycle

The Create_MediaLifecycle API creates a media lifecycle record in the Media Lifecycle table. This procedure is optional since an interaction can be created without any media items and their associated media lifecycles.

### Procedure Specification

```
PROCEDURE create_medialifecycle
(
   p_api_version   in      number,
   p_init_msg_list in      varchar2        default fnd_api.g_false,
   p_commit        in      varchar2        default fnd_api.g_false,
   p_resp_appl_id  in      number          default null,
   p_resp_id       in      number          default null,
   p_user_id       in      number,
   p_login_id      in      number          default null,
```

```
   x_return_status out     varchar2,
   x_msg_count     out     number,
   x_msg_data      out     varchar2,
   p_media_lc_rec  in      media_lc_rec_type
);
```

### Current Version

```
1.0
```

### Parameter Descriptions

The Create_MediaLifecycle API sets the value of the input parameter to NULL if the parameter corresponds to the value of the *G_MISS_X* constant. If the parameter does not correspond to this constant, then the API retains the passed-in value.

The following table describes the IN parameters associated with this API.

***Table C–8 Create Media Lifecycle IN Parameters***

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_api_version | NUMBER | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_init_msg_list | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_commit | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_resp_appl_id | NUMBER | No[1] | Application identifier |
| p_resp_id | NUMBER | No[*] | Responsibility identifier |
| p_user_id | NUMBER | No[*] | Corresponds to the column USER_ID in the table FND_USER, and identifies the Oracle Applications user. |
| p_login_id | NUMBER | No | Corresponds to the column LOGIN_ID in the table FND_LOGINS, and identifies the login session. |

*Table C–8 Create Media Lifecycle IN Parameters*

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_media_lc_rec | media_lc_rec_type | Yes | This record captures the media lifecycle.<br>The following record parameters are always validated:<br>■ start_date_time<br>If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place.<br>■ end_date_time<br>See "Media Item Lifecycle Record Type" on page C-15 for the record specification. |

[1] The Application ID, Responsibility ID, and User ID determine which profile values are used as defaults. Those items marked with an asterisk also follow these guidelines.

The following table describes the OUT parameters associated with this API.

*Table C–9 Create Media Lifecycle OUT Parameters*

| Parameter | Data Type | Description |
|---|---|---|
| x_return_status | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_count | NUMBER | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_data | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |

## C.7.5 Create_Interaction

The Create_Interaction API creates an interaction record in the Interactions table, associates it with one or more activities in the Activities table and sets the status of the interaction to inactive. You can pass multiple Activity Records in the Create_Interaction API using the p_activities parameter. The data type for this parameter is activity_tlb_type which is a table of activity_rec_type values.

### Procedure Specification

```
PROCEDURE create_interaction
```

```
(
   p_api_version           in     number,
   p_init_msg_list         in     varchar2 default fnd_api.g_false,
   p_commit                in     varchar2 default fnd_api.g_false,
   p_resp_appl_id          in     number   default null,
   p_resp_id               in     number   default null,
   p_user_id               in     number,
   p_login_id              in     number   default null,
   x_return_status         out    varchar2,
   x_msg_count             out    number,
   x_msg_data              out    varchar2,
   p_interaction_rec       in     interaction_rec_type,
   p_activities            in     activity_tbl_type
);
```

### Current Version

1.0

### Parameter Descriptions

The Create_Interaction API sets the value of the input parameter to NULL if the parameter corresponds to the value of the *G_MISS_X* constant. If the parameter does not correspond to this constant, then the API retains the passed-in value.

The following table describes the IN parameters associated with this API.

Create Interaction IN Parameters

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_api_version | NUMBER | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_init_msg_list | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_commit | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_resp_appl_id | NUMBER | No[1] | Application identifier |
| p_resp_id | NUMBER | No[*] | Responsibility identifier |
| p_user_id | NUMBER | No[*] | Corresponds to the column USER_ID in the table FND_USER, and identifies the Oracle Applications user. |

Create Interaction IN Parameters

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_login_id | NUMBER | No | Corresponds to the column LOGIN_ID in table FND_LOGINS, and identifies the login session. |
| p_interaction_rec | INTERACTION_REC_TYPE | Yes | Contains the elements that comprise the interaction record.<br>The following record parameters are always validated:<br>■ start_date_time<br>If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place.<br>■ end_date_time<br>■ handler_id<br>■ outcome_id<br>■ result_id<br>■ reason_id<br>■ resource_id<br>■ party_id<br>■ source_code<br>■ source_code_id<br>See "Interaction Record Type" on page C-12 for the record specification. |

Create Interaction IN Parameters

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_activities | activity_tbl_type | Yes | The following record parameters are always validated:<br>■ start_date_time<br>If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place.<br>■ end_date_time<br>■ action_item_id<br>■ outcome_id<br>■ result_id<br>■ reason_id<br>■ action_id<br>■ source_code<br>■ source_code_id<br>See "Activity Record Type" on page C-13 for the record specification. |

[1] The Application ID, Responsibility ID, and the User ID determine which profile values are used as defaults. Those items marked with an asterisk also follow these guidelines.

The following table describes the OUT parameters associated with this API.

***Table C–10 Create Interaction OUT Parameters***

| Parameter | Data Type | Description |
|---|---|---|
| x_return_status | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_count | NUMBER | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_data | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |

# C.8 Cached Creation APIs

Desktop applications, such as a service request system, that must retain interaction information even when transactions are broken as well as the ability to associate

multiple media items and activities with a single interaction can take advantage of the following cached creation APIs:

- Open_MediaItem
- Update_MediaItem
- Add_MediaLifecycle
- Update_MediaLifecycle
- Close_MediaItem
- Open_Interaction
- Update_Interaction
- Add_Activity
- Update_Activity
- Update_ActivityDuration
- Close_Interaction

## C.8.1 Overview

Cached creation APIs enable Interaction History clients to create and update interactions on the server prior to closing the interaction, before the interaction becomes an historical record which cannot be updated or deleted. In this case, a partial interaction record is stored on the server and updated as required until the final Close_Interaction API call makes it a historical record. This mechanism provides some level of fault-tolerance and recovery to client applications creating the interactions but also provides slower performance than non-cached creation APIs. Unlike the non-cached creation APIs, the cached creation APIs require several procedures to create a single media item, media lifecycle, activity, or interaction. Cached creation APIs can also have one set of APIs nested within another.

If you use cached creation APIs for transactions and communication between the client application and the server is disrupted, all information captured up to the point of disruption can be recovered.

***Table C–11 Interaction History Cached Creation APIs***

| Procedure | Description |
|---|---|
| Open_MediaItem | Creates a media item in table Media Items table. |

*Table C–11 Interaction History Cached Creation APIs*

| Procedure | Description |
|---|---|
| Update_MediaItem | Updates the current media item with values supplied by the calling application. |
| Add_MediaLifecycle | Creates a media lifecycle record in the Media Lifecycle table and associates it with the media item passed by the calling application. |
| Update_MediaLifecycle | Updates the current media lifecycle with values supplied by the calling application. |
| Close_MediaItem | Sets the status of the media item and its associated media lifecycle to inactive so that they can no longer be updated. |
| Open_Interaction | Creates an interaction in the Interactions table. |
| Update_Interaction | Updates the current interaction with values supplied by the calling application. |
| Add_Activity | Creates an activity in the Activities table that is associated with the interaction passed by the calling application. |
| Update_Activity | Updates the current activity with values supplied by the calling application. |
| Update_ActivityDuration | Updates the current activity's *end_date_time* and *duration* fields with values supplied by the calling application. |
| Close_Interaction | Sets the status of the interaction and its associated activities to inactive so that they can no longer be modified. |

## C.8.2 Process Flows

Since cached creation APIs perform their operations in a specific sequence, APIs that perform an update function cannot be invoked unless a corresponding API that performs an open or add function has first been invoked. For example, the Update_Interaction API cannot be invoked unless the corresponding Open_Interaction API has first been invoked. The Update_Activity API cannot be invoked unless the corresponding Add_Activity API has first been invoked. Similarly an update API cannot be invoked for a record that has already been closed. For example, after you invoke the Close_Interaction API you cannot invoke the Update_Interaction API for the same record.

The following figure provides an overview of a common process flow for cached creation APIs. The process flows described in this figure are common but not required since some API calls are optional.

1. The calling application executes the first of three steps required to create a media item, by invoking the Open_MediaItem API. This API inserts generic values in the Media Items table of the Oracle database.
2. The calling application executes the second of three step required to create a media item, by invoking the Update_MediaItem API. This API updates the Media Items table with values supplied by the calling application.

> **Note:** If a media lifecycle record is added to the media item, this must occur before closing the media item record. Once the media item record is closed it cannot be modified.

3. Before the media item becomes an historical record in the database, it can optionally contain an associated media lifecycle record. The calling application executes the first of two steps required to create a media lifecycle record by invoking the Add_MediaLifecycle API. This API inserts generic values in the Media Lifecycles table.
4. The calling application executes the second of two steps to create a media lifecycle record by invoking the Update_MediaLifecycle API. This API updates the media lifecycle record with values supplied by the calling application.
5. The calling application executes the third of three steps required to create a media item by invoking the Close_MediaItem API. This API performs all required validations to make the media item and its associated media lifecycle record, historical records.

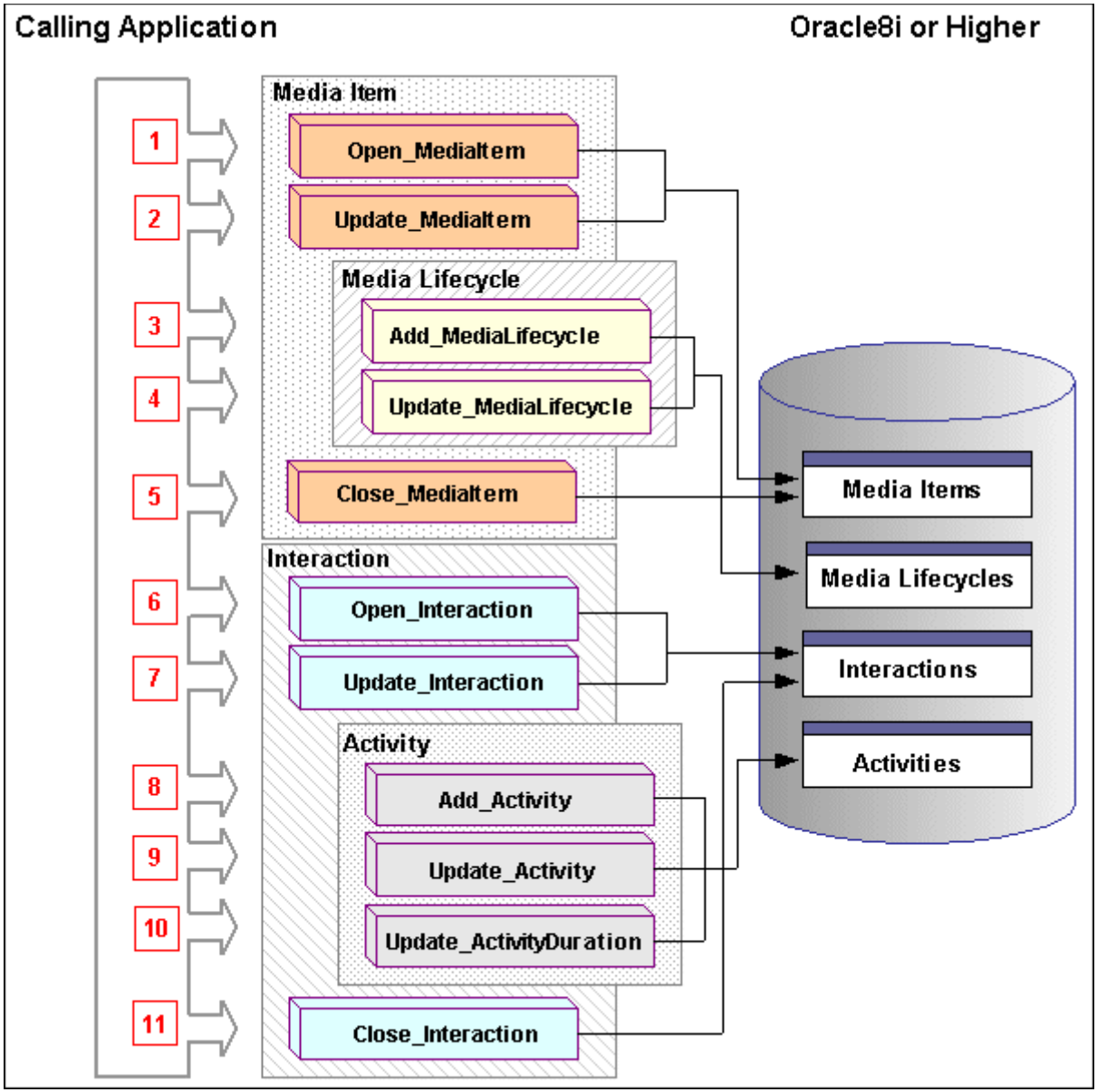
> **Note:** Media Item and Media Lifecycle calls are optional since interactions can exist without a media item or its associated media lifecycle record.

6. The calling application executes the first of three steps required to create an interaction record by invoking the Open_Interaction API. This API inserts generic values in the Interactions table.
7. The calling application executes the second of three steps required to create an interaction by invoking the Update_Interaction API. This API updates the Interactions table with values supplied by the calling application.

**Note:** One or more activities must be created and associated with the interaction before closing the interaction record. Once the interaction record is closed it cannot be modified.

8. The calling application executes the first of three steps required to create an activity that is associated with the interaction by invoking the Add_Activity API. This API inserts generic values in the Activities table.
9. The calling application executes the second of three steps required to create an activity that is associated with an interaction by invoking the Update_Activity API. This API updates the Activities table with values supplied by the calling application.
10. The calling application executes the third of three steps required to create an activity that is associated with an interaction by invoking the Update_ ActivityDuration API. This API updates the current activity's duration with values supplied by the calling application.
11. The calling application performs the third of three steps required to create an interaction by invoking the Close_Interaction API. This API performs all required validations to make the interaction and its associated activity historical records in the Oracle database.

*Figure C–4 Cached Creation API Process Flow Overview*



The following figure provides a detailed explanation of the first five steps required to create a customer Interaction using the cached creation APIs. The process flows described in this figure are common but not required since some API calls are optional.

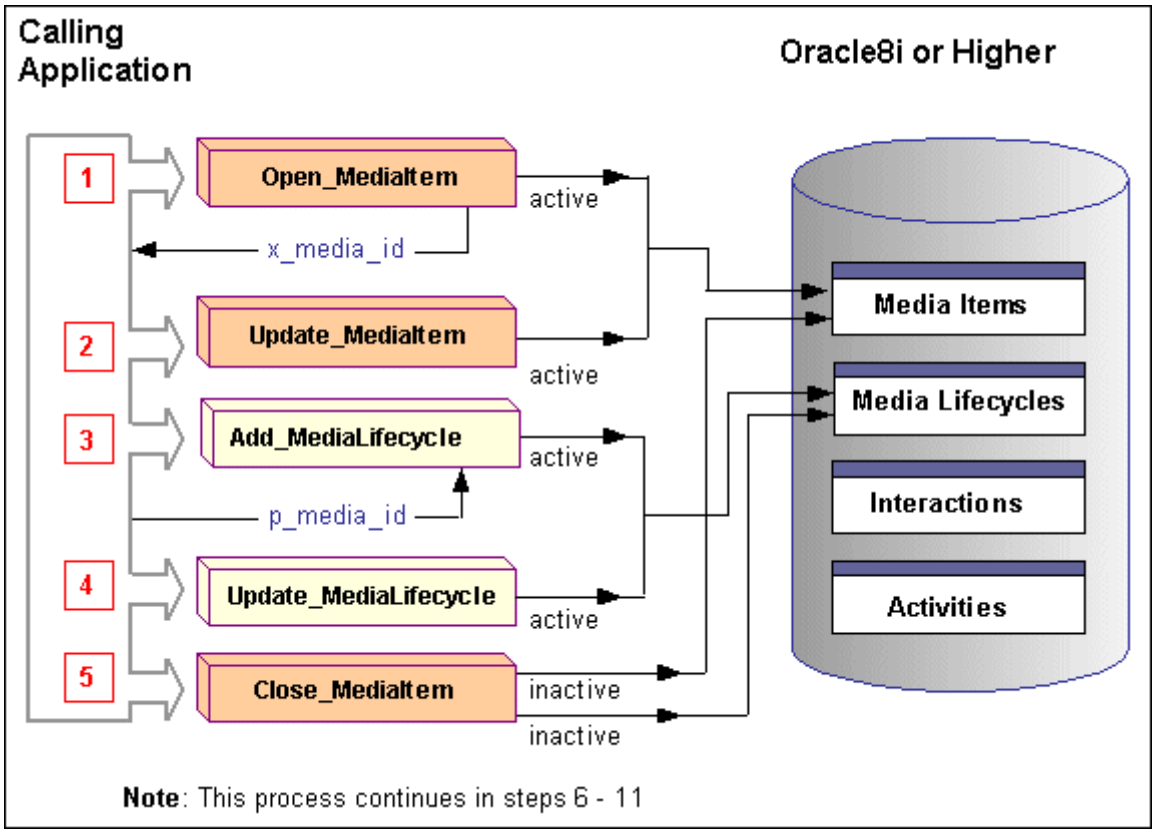
1. The Open_MediaItem API inserts generic records in the Media Items table of the Oracle database, sets the media item's status to active so that it can be updated, and returns a sequence generated identifier to the calling application as **x_media_id**.
2. The Update_MediaItem API updates the Media Items table with values supplied by the calling application and retains the media item's active status.
3. The calling application optionally associates a media lifecycle record with the media item by invoking the Add_MediaLifecycle API. The calling application inputs the media items's sequence generated identifier as **p_media_id**, inserts generic values in the Media Lifecycles table and sets the status of the media lifecycle to active so that it can be updated.
4. The Update_MediaLifecycle API updates the media lifecycle record with values supplied by the calling application and retains its active status.
5. The Close_MediaItem API sets the status of the media item and its associated media lifecycle record to inactive so that they can no longer be updated, and performs validations to verify that the media item and media lifecycle record are associated with each other.

---

**Note:** The process of creating a customer interaction record using the cached creation APIs is not yet complete. This process is continued in steps 6 -11 as illustrated in the following figure.

---

*Figure C–5 Cached Creation API Process Flow: Steps 1 - 5*



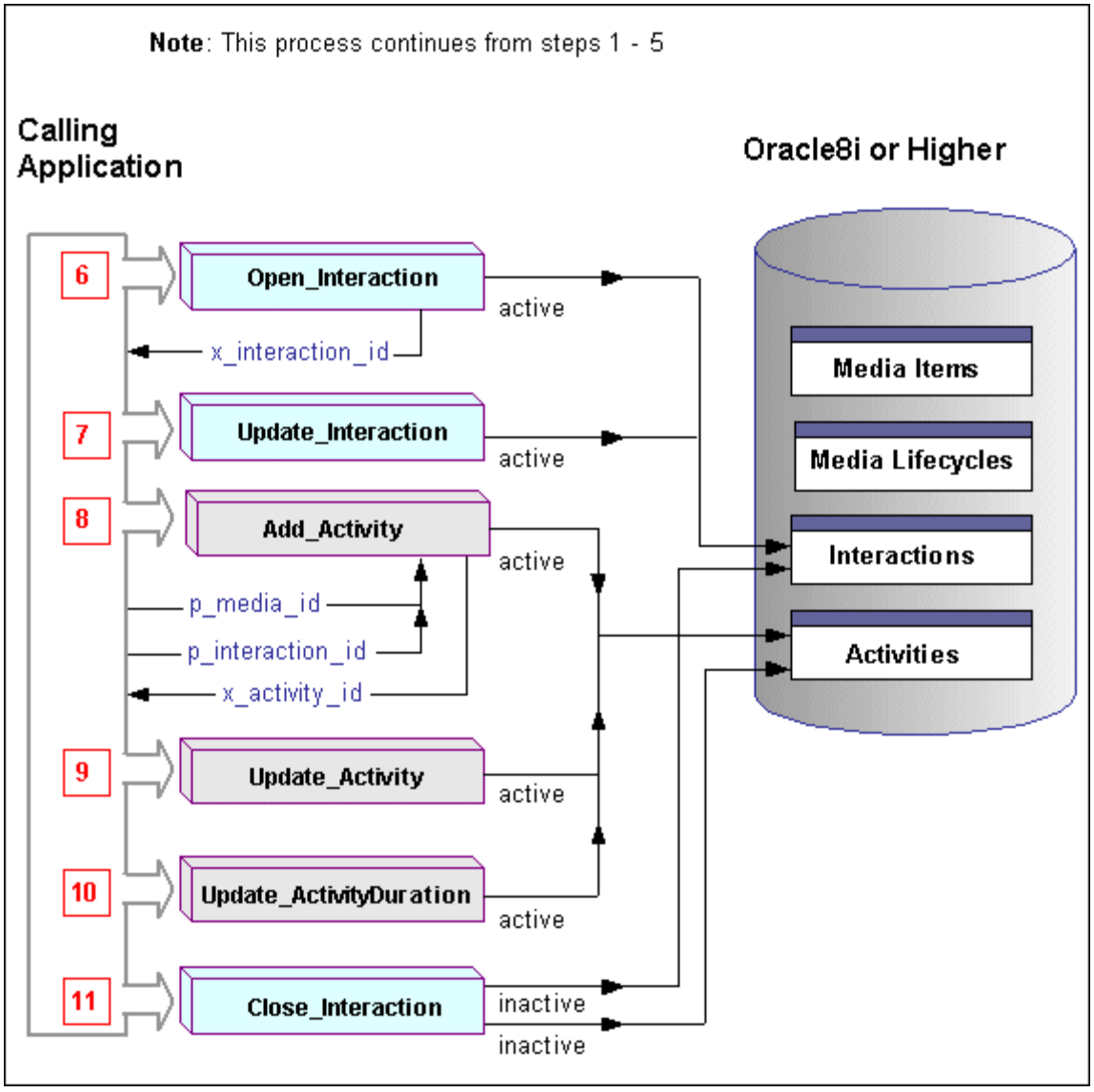

The preceding figure provides a detailed explanation of the remaining six steps required to create a customer Interaction using the cached creation APIs.

6. The Open_Interaction API inserts generic records in the Interactions table, sets the interaction's status to active so that it can be updated, and returns a sequence generated identifier to the calling application as **x_interaction_id**.
7. The Update_Interaction API updates the Interactions table with values supplied by the calling application and retains the interaction's active status.

> **Note:** One or more activities must be created and associated with the interaction before closing the interaction record. Once the interaction record is closed it cannot be modified.

8. The calling application invokes the Add_Activity API, inputs the media item's unique identifier as **p_media_id**, and inputs the interaction's unique identifier as **p_interaction_id**. The API sends a sequence generated identifier to the calling application as **x_activity_id**, inserts generic records in the Activities table, and sets the activity status to active so that it can be updated.
9. The Update_Activity API updates the Activities table with values supplied by the calling application and retains the activity's active status.
10. The Update_ActivityDuration API updates the Activities table 's *end_date_time* and *duration* fields with values supplied by the calling application, and retains the activity's active status so that it can still be modified.
11. The Create_Interaction API sets the status of the interaction and its associated activity to inactive so that they can no longer be updated, and performs validations to verify that the interaction and activity are associated with each other.

***Figure C–6 Cached Creation API Process Flow: Steps 6 -1 1***



## C.8.3 Open_MediaItem

The Open_MediaItem API Creates a media item in the Media Items table, sets the status of the media item to active, and returns a sequence generated media identification number as *p_media_id*.

### Procedure Specification

```
PROCEDURE Open_MediaItem
(
        p_api_version   in      number,
        p_init_msg_list in      varchar2        default fnd_api.g_false,
        p_commit        in      varchar2        default fnd_api.g_false,
        p_resp_appl_id  in      number          default null,
        p_resp_id       in      number          default null,
        p_user_id       in      number,
        p_login_id      in      number          default null,
        x_return_status out     varchar2,
        x_msg_count     out     number,
        x_msg_data      out     varchar2,
        p_media_rec     in      media_rec_type,
        x_media_id      out     number
);
```

### Current Version

```
1.0
```

### Parameter Descriptions

The following table describes the IN parameters associated with this API.

***Table C–12 Open Media Item IN Parameters***

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_api_version | NUMBER | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_init_msg_list | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_commit | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_resp_appl_id | NUMBER | No[1] | Application identifier |
| p_resp_id | NUMBER | No[*] | Responsibility identifier |

*Table C–12 Open Media Item IN Parameters*

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_user_id | NUMBER | No* | Corresponds to the column USER_ID in table FND_USER, and identifies the Oracle Applications user. |
| p_login_id | NUMBER | No | Corresponds to the column LOGIN_ID in table FND_LOGINS, and identifies the login session. |
| p_media | media_rec_type | | Enumerates the elements that comprise a media record.<br>The following record parameters are always validated:<br>■ start_date_time<br>If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place.<br>■ end_date_time<br>See "Media Item Record Type" on page C-15 for the record specification. |

[1] The Application ID, Responsibility ID, and User ID determine which profile values are used as defaults. Those items marked with an asterisk also follow these guidelines.

*Table C–13 Open Media Item OUT Parameter*

| Parameter | Data Type | Description |
|---|---|---|
| x_return_status | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_count | NUMBER | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_data | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_media_id | NUMBER | The record number for the created media item. It is automatically generated by sequence JTF_IH_MEDIA_ITEMS_S1. |

## C.8.4 Update_MediaItem

The Update_MediaItem API updates the current Media Item with values supplied by the calling application and sets the status of the media item to active.

### Procedure Specification

```
PROCEDURE Update_MediaItem
(
   p_api_version   in      number,
   p_init_msg_list in      varchar2        default fnd_api.g_false,
   p_commit        in      varchar2        default fnd_api.g_false,
   p_resp_appl_id  in      number          default null,
   p_resp_id       in      number          default null,
   p_user_id       in      number,
   p_login_id      in      number          default null,
   x_return_status out     varchar2,
   x_msg_count     out     number,
   x_msg_data      out     varchar2,
   p_media_rec     in      media_rec_type
);
```

### Current Version

```
1.0
```

### Parameter Descriptions

The Update_MediaItem API does **not** update columns which have passed-in values corresponding to the *G_MISS_X* constants.

The following table describes the IN parameters associated with this API.

***Table C–14 Update Media Item IN Parameters***

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_api_version | NUMBER | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_init_msg_list | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_commit | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_resp_appl_id | NUMBER | No[1] | Application identifier |
| p_resp_id | NUMBER | No[*] | Responsibility identifier |
| p_user_id | NUMBER | No | Corresponds to the column USER_ID in the table FND_USER, and identifies the Oracle Applications user. |

*Table C–14 Update Media Item IN Parameters*

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_login_id | NUMBER | No* | Corresponds to the column LOGIN_ID in table FND_LOGINS, and identifies the login session. |
| p_media_rec | media_rec_type | | Enumerates the elements that comprise a media record.<br>The following record parameters are always validated:<br>■ start_date_time<br>If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place.<br>■ end_date_time<br>See "Media Item Record Type" on page C-15 for the record specification. |

[1] The Application ID, Responsibility ID, and User ID determine which profile values are used as defaults. Those items marked with an asterisk also follow these guidelines.

The following table describes the OUT parameters associated with this API.

*Table C–15 Update Media Item OUT Parameters*

| Parameter | Data Type | Description |
|---|---|---|
| x_return_status | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_count | NUMBER | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_data | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |

## C.8.5 Add_MediaLifecycle

The Add_MediaLifecycle API creates a media lifecycle record in the Media Lifecycle table, associates it with a Media Item passed by the calling application, and returns a sequence generated milcs_id number. The status of the media item and its associated media lifecycle remain active.

### Procedure Specification

```
PROCEDURE Add_MediaLifecycle
(
   p_api_version   in      number,
   p_init_msg_list in      varchar2        default fnd_api.g_false,
   p_commit        in      varchar2        default fnd_api.g_false,
   p_resp_appl_id  in      number          default null,
   p_resp_id       in      number          default null,
   p_user_id       in      number,
   p_login_id      in      number          default null,
   x_return_status out     varchar2,
   x_msg_count     out     number,
   x_msg_data      out     varchar2,
   p_media_lc_rec  in      media_lc_rec_type,
   x_milcs_id      out     number
);
```

### Current Version

```
1.0
```

### Parameter Descriptions

The following table describes the IN parameters associated with this API.

***Table C–16 Add Media Lifecycle IN Parameters***

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_api_version | NUMBER | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_init_msg_list | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_commit | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_resp_appl_id | NUMBER | No[1] | Application identifier |
| p_resp_id | NUMBER | No[*] | Responsibility identifier |
| p_user_id | NUMBER | No[*] | Corresponds to the column USER_ID in table FND_USER, and identifies the Oracle Applications user. |
| p_login_id | NUMBER | No | Corresponds to the column LOGIN_ID in table FND_LOGINS, and identifies the login session. |

*Table C–16 Add Media Lifecycle IN Parameters*

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_media_lc_rec | media_lc_rec_ type | | Composite record that enumerates the elements that comprise a media lifecycle.<br>The following record parameters are always validated:<br>■ start_date_time<br>If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place.<br>■ end_date_time<br>See "Media Item Lifecycle Record Type" on page C-15 for the record specification. |

[1] The Application ID, Responsibility ID, and User ID determine which profile values are used as defaults. Those items marked with an asterisk also follow these guidelines.

*Table C–17 Add Media Lifecycle OUT Parameter*

| Parameter | Data Type | Description |
|---|---|---|
| x_return_status | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_count | NUMBER | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_data | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_milcs_id | NUMBER | Corresponds to the sequence generated media lifecycle identifier for the record created. |

## C.8.6 Update_MediaLifecycle

The Update_MediaLifecycle API updates the current media lifecycle record with values supplied by the calling application. The status of the media lifecycle remains active.

### Procedure Specification

```
PROCEDURE Update_MediaLifecycle
(
```

```
   p_api_version   in      number,
   p_init_msg_list in      varchar2        default fnd_api.g_false,
   p_commit        in      varchar2        default fnd_api.g_false,
   p_resp_appl_id  in      number          default null,
   p_resp_id       in      number          default null,
   p_user_id       in      number,
   p_login_id      in      number          default null,
   x_return_status out     varchar2,
   x_msg_count     out     number,
   x_msg_data      out     varchar2,
   p_media_lc_rec  in      media_lc_rec_type
);
```

### Current Version

```
1.0
```

### Parameter Descriptions

The Update_MediaLifecycle API does **not** update columns with pass-in values that correspond to the *G_MISS_X* constants.

The following table describes the IN parameters associated with this API.

***Table C–18 Update Media Lifecycle IN Parameters***

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_api_version | NUMBER | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_init_msg_list | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_commit | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_resp_appl_id | NUMBER | No[1] | Application identifier |
| p_resp_id | NUMBER | No* | Responsibility identifier |
| p_user_id | NUMBER | No* | Corresponds to the column USER_ID in table FND_USER, and identifies the Oracle Applications user. |
| p_login_id | NUMBER | No | Corresponds to the column LOGIN_ID in table FND_LOGINS, and identifies the login session. |

*Table C–18 Update Media Lifecycle IN Parameters*

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_media_lc_rec | media_lc_rec_type | Yes | Composite record that enumerates the elements that comprise a media lifecycle.<br>The following record parameters are always validated:<br>■ start_date_time<br>If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place.<br>■ end_date_time<br>See "Media Item Lifecycle Record Type" on page C-15 for the record specification. |

[1] The Application ID, Responsibility ID, and User ID determine which profile values are used as defaults. Those items marked with an asterisk also follow these guidelines.

*Table C–19 Update Media Lifecycle OUT Parameters*

| Parameter | Data Type | Description |
|---|---|---|
| x_return_status | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_count | NUMBER | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_data | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |

## C.8.7 Close_MediaItem

The Close_MediaItem API sets the status of the media item and its associated media lifecycle to inactive so that it can no longer be updated.

### Procedure Specification

```
PROCEDURE Close_MediaItem
(
   p_api_version   in      number,
   p_init_msg_list in      varchar2        default fnd_api.g_false,
   p_commit        in      varchar2        default fnd_api.g_false,
```

```
   p_resp_appl_id  in      number          default null,
   p_resp_id       in      number          default null,
   p_user_id       in      number,
   p_login_id      in      number          default null,
   x_return_status out     varchar2,
   x_msg_count     out     number,
   x_msg_data      out     varchar2,
   p_media_rec     in      media_rec_type
);
```

### Current Version

```
1.0
```

### Parameter Descriptions

The following table describes the IN parameters associated with this API.

***Table C–20 Close Media Item IN Parameters***

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_api_version | NUMBER | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_init_msg_list | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_commit | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_resp_appl_id | NUMBER | No[1] | Application identifier |
| p_resp_id | NUMBER | No* | Responsibility identifier |
| p_user_id | NUMBER | No* | Corresponds to the column USER_ID in table FND_USER, and identifies the Oracle Applications user. |
| p_login_id | NUMBER | No | Corresponds to the column LOGIN_ID in table FND_LOGINS, and identifies the login session. |

*Table C–20 Close Media Item IN Parameters*

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_media_rec | media_rec_type | Yes | Enumerates the elements that comprise a media record.<br>The following record parameters are always validated:<br>■ start_date_time<br>If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place.<br>■ end_date_time<br>If the *end_date_time* parameter is not supplied at the time that the *Close_MediaItem* API is invoked, then *sysdate* is inserted in its place.<br>See "Media Item Record Type" on page C-15 for the record specification. |

[1] The Application ID, Responsibility ID, and User ID determine which profile values are used as defaults. Those items marked with an asterisk also follow these guidelines.

*Table C–21 Close Media Item OUT Parameters*

| Parameter | Data Type | Description |
|---|---|---|
| x_return_status | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_count | NUMBER | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_data | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |

## C.8.8 Open_Interaction

The Open_Interaction API Creates an interaction in the Interactions table, sets the status of the interaction to active, and returns a sequence generated interaction_id number.

### Procedure Specification

```
PROCEDURE Open_Interaction
(
```

```
   p_api_version           in      number,
   p_init_msg_list         in      varchar2 default fnd_api.g_false,
   p_commit                in      varchar2 default fnd_api.g_false,
   p_resp_appl_id          in      number   default null,
   p_resp_id               in      number   default null,
   p_user_id               in      number,
   p_login_id              in      number   default null,
   x_return_status         out     varchar2,
   x_msg_count             out     number,
   x_msg_data              out     varchar2,
   p_interaction_rec       in      interaction_rec_type,
   x_interaction_id        out     number
);
```

### Current Version

```
1.0
```

### Parameter Descriptions

The following table describes the IN parameters associated with this API.

***Table C–22 Open Interaction IN Parameters***

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_api_version | NUMBER | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_init_msg_list | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_commit | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_resp_appl_id | NUMBER | No[1] | Application identifier |
| p_resp_id | NUMBER | No* | Responsibility identifier |
| p_user_id | NUMBER | No* | Corresponds to the column USER_ID in table FND_USER, and identifies the Oracle Applications user. |
| p_login_id | NUMBER | No | Corresponds to the column LOGIN_ID in table FND_LOGINS, and identifies the login session. |

*Table C–22 Open Interaction IN Parameters*

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_interaction_rec | interaction_rec_type | Yes | Contains the elements that comprise the interaction record.<br>The following record parameters are always validated:<br>■ start_date_time<br>If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place.<br>■ end_date_time<br>■ handler_id<br>■ outcome_id<br>■ result_id<br>■ reason_id<br>■ resource_id<br>■ party_id<br>■ source_code<br>■ source_code_id<br>See "Interaction Record Type" on page C-12 for the record specification. |

[1] The Application ID, Responsibility ID, and User ID determine which profile values are used as defaults. Those items marked with an asterisk also follow these guidelines.

The following table describes the OUT parameters associated with this API.

*Table C–23 Open Interaction OUT Parameter*

| Parameter | Data Type | Description |
|---|---|---|
| x_return_status | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_count | NUMBER | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_data | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_interaction_id | NUMBER | Corresponds to a sequence generated reference for the interaction record. |

## C.8.9 Update_Interaction

The Update_Interaction API updates the current interaction with values supplied by the calling application. The state of the interaction remains open.

### Procedure Specification

```
PROCEDURE update_interaction
(
   p_api_version           in      number,
   p_init_msg_list         in      varchar2 default fnd_api.g_false,
   p_commit                in      varchar2 default fnd_api.g_false,
   p_resp_appl_id          in      number   default null,
   p_resp_id               in      number   default null,
   p_user_id               in      number,
   p_login_id              in      number   default null,
   x_return_status         out     varchar2,
   x_msg_count             out     number,
   x_msg_data              out     varchar2,
   p_interaction_rec       in      interaction_rec_type
);
```

### Current Version

```
1.1
```

> **Note:** Calling with p_api_version of 1.0 performs single-party validation. Calling with p_api_version of 1.1 performs multi-party validation. See Appendix D, "Data Validations".

### Parameter Descriptions

The Update_Interaction API does **not** update columns that have passed-in values corresponding to the *G_MISS_X* constants

The following table describes the IN parameters associated with this API.

***Table C–24 Update Interaction IN Parameters***

| Parameter | Date Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_api_version | NUMBER | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_init_msg_list | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |

*Table C–24 Update Interaction IN Parameters*

| Parameter | Date Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_commit | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_resp_appl_id | NUMBER | No[1] | Application identifier |
| p_resp_id | NUMBER | No* | Responsibility identifier |
| p_user_id | NUMBER | No | Corresponds to the column USER_ID in table FND_USER, and identifies the Oracle Applications user. |
| p_login_id | NUMBER | No | Corresponds to the column LOGIN_ID in table FND_LOGINS, and identifies the login session. |
| p_interaction_rec | interaction_rec_type | Yes | Used in updating the interaction record.<br>The following record parameters are always validated:<br>■ start_date_time<br>If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place.<br>■ end_date_time<br>■ handler_id<br>■ outcome_id<br>■ result_id<br>■ reason_id<br>■ resource_id<br>■ party_id<br>■ source_code<br>■ source_code_id<br>See "Interaction Record Type" on page C-12 for the record specification. |

[1] The Application ID, Responsibility ID, and User ID determine which profile values are used as defaults. Those items marked with an asterisk also follow these guidelines.

The following table describes the OUT parameters associated with this API.

*Table C–25 Update Interaction OUT Parameters*

| Parameter | Data Type | Description |
|---|---|---|
| x_return_status | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_count | NUMBER | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_data | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |

## C.8.10 Add_Activity

The Add_Activity API creates an activity in the Activities table, associates it with the interaction passed by the calling application, and returns a sequence generated activity_id number. The status of the interaction and associated activity remain active.

### Procedure Specification

```
PROCEDURE Add_Activity
(
   p_api_version           in      number,
   p_init_msg_list         in      varchar2   default fnd_api.g_false,
   p_commit                in      varchar2   default fnd_api.g_false,
   p_resp_appl_id          in      number     default null,
   p_resp_id               in      number     default null,
   p_user_id               in      number,
   p_login_id              in      number     default null,
   x_return_status         out     varchar2,
   x_msg_count             out     number,
   x_msg_data              out     varchar2,
   p_activity_rec          in      activity_rec_type,
   x_activity_id           out     number
);
```

### Current Version

```
1.0
```

### Parameter Descriptions

The following table describes the IN parameters associated with this API.

***Table C–26 Add Activity IN Parameters***

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_api_version | NUMBER | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_init_msg_list | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_commit | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_resp_appl_id | NUMBER | No[1] | Application identifier |
| p_resp_id | NUMBER | No* | Responsibility identifier |
| p_user_id | NUMBER | No* | Corresponds to the column USER_ID in table FND_USER, and identifies the Oracle Applications user. |
| p_login_id | NUMBER | No | Corresponds to the column LOGIN_ID in table FND_LOGINS, and identifies the login session. |
| p_activity_rec | activity_rec_type | Yes | Used in updating the interaction record.<br>The following record parameters are always validated:<br>■ start_date_time<br>If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place.<br>■ end_date_time<br>■ action_item_id<br>■ outcome_id<br>■ result_id<br>■ reason_id<br>■ action_id<br>■ source_code<br>■ source_code_id<br>See "Activity Record Type" on page C-13 for the record specification. |

[1] The Application ID, Responsibility ID, and User ID determine which profile values are used as defaults. Those items marked with an asterisk also follow these guidelines.

The following table describes the OUT parameters associated with this API.

***Table C–27 Add Activity OUT Parameters***

| Parameter | Data Type | Description |
|---|---|---|
| x_return_status | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_count | NUMBER | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_data | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_activity_id | NUMBER | Corresponds to the sequence generated activity identifier for the record created. |

## C.8.11 Update_Activity

The Update_Activity API updates the current activity with values supplied by the calling application. The status of the activity remains active.

### Procedure Specification

```
PROCEDURE Update_Activity
(
   p_api_version           in      number,
   p_init_msg_list         in      varchar2    default fnd_api.g_false,
   p_commit                in      varchar2    default fnd_api.g_false,
   p_resp_appl_id          in      number      default null,
   p_resp_id               in      number      default null,
   p_user_id               in      number,
   p_login_id              in      number      default null,
   x_return_status         out     varchar2,
   x_msg_count             out     number,
   x_msg_data              out     varchar2,
   p_activity_rec          in      activity_rec_type
);
```

### Current Version

```
1.0
```

### Parameter Descriptions

The Update_Activity API does **not** update columns which have passed-in values corresponding to the *G_MISS_X* constants.

The following table describes the IN parameters associated with this API.

***Table C–28 Update Activity IN Parameters***

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_api_version | NUMBER | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_init_msg_list | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_commit | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_resp_appl_id | NUMBER | No[1] | Application identifier |
| p_resp_id | NUMBER | No* | Responsibility identifier |
| p_user_id | NUMBER | No* | Corresponds to the column USER_ID in table FND_USER, and identifies the Oracle Applications user. |
| p_login_id | NUMBER | No | Corresponds to the column LOGIN_ID in table FND_LOGINS, and identifies the login session. |
| p_activity_rec | activity_rec_type | Yes | Used in updating the interaction record.<br>The following record parameters are always validated:<br>■ start_date_time<br>If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place.<br>■ end_date_time<br>■ action_item_id<br>■ outcome_id<br>■ result_id<br>■ reason_id<br>■ action_id<br>■ source_code<br>■ source_code_id<br>See "Activity Record Type" on page C-13 for the record specification. |

[1] The Application ID, Responsibility ID, and User ID determine which profile values are used as defaults. Those items marked with an asterisk also follow these guidelines.

The following table describes the OUT parameters associated with this API.

***Table C–29 Update Activity OUT Parameters***

| Parameter | Data Type | Description |
|---|---|---|
| x_return_status | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_count | NUMBER | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_data | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |

## C.8.12 Update_ActivityDuration

The Update_ActivityDuration API updates the current activity's *end_date_time* and *duration* fields with values supplied by the calling application.

### Procedure Specification

```
PROCEDURE Update_ActivityDuration
(
    p_api_version           in      number,
    p_init_msg_list         in      varchar2   default fnd_api.g_false,
    p_commit                in      varchar2   default fnd_api.g_false,
    p_resp_appl_id          in      number     default null,
    p_resp_id               in      number     default null,
    p_user_id               in      number,
    p_login_id              in      number     default null,
    x_return_status         out     varchar2,
    x_msg_count             out     number,
    x_msg_data              out     varchar2,
    p_activity_id           in      number,
    p_end_date_time         in      date,
    p_duration              in      number
);
```

### Current Version

```
1.0
```

### Parameter Descriptions

The Update_ActivityDuration API does **not** update columns which have passed-in values corresponding to the *G_MISS_X* constants.

The following table describes the IN parameters associated with this API.

***Table C–30 Update Activity Duration IN Parameters***

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_api_version | NUMBER | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_init_msg_list | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_commit | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_resp_appl_id | NUMBER | No[1] | Application identifier |
| p_resp_id | NUMBER | Optional* | Responsibility identifier |
| p_user_id | NUMBER | Optional* | Corresponds to the column USER_ID in table FND_USER, and identifies the Oracle Applications user. |
| p_login_id | NUMBER | No | Corresponds to the column LOGIN_ID in table FND_LOGINS, and identifies the login session. |
| p_activity_id | NUMBER | | Activity identifier. This number corresponds to a certain activity. |
| p_end_date_time | DATE | | End date time. Time in date format at the end of the transaction. |
| p_duration | NUMBER | | Duration. Time difference between the end_date_time and the start_date_time converted to seconds. |

[1] The application ID, responsibility ID, and user ID determine which profile values are used as defaults. Those items marked with an asterisk also follow these guidelines.

The following table describes the OUT parameters associated with this API.

*Table C–31 Update Activity Duration OUT Parameters*

| Parameter | Data Type | Description |
|---|---|---|
| x_return_status | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_count | NUMBER | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_data | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |

## C.8.13 Close_Interaction

The Close_Interaction API Sets the status of the interaction and its associated activities to inactive so that they can no longer be updated.

### Procedure Specification

```
PROCEDURE Close_Interaction
(
   p_api_version           in      number,
   p_init_msg_list         in      varchar2   default fnd_api.g_false,
   p_commit                in      varchar2   default fnd_api.g_false,
   p_resp_appl_id          in      number     default null,
   p_resp_id               in      number     default null,
   p_user_id               in      number,
   p_login_id              in      number     default null,
   x_return_status         out     varchar2,
   x_msg_count             out     number,
   x_msg_data              out     varchar2,
   p_interaction_rec       in      interaction_rec_type
);
```

### Current Version

```
1.1
```

**Note:** Calling with p_api_version of 1.0 performs single-party validation. Calling with p_api_version of 1.1 performs multi-party validation. See Appendix D, "Data Validations".

### Parameter Descriptions

The following table describes the IN parameters associated with this API.

*Table C–32 Close Interaction IN Parameters*

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_api_version | NUMBER | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_init_msg_list | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_commit | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_resp_appl_id | NUMBER | No[1] | Application identifier |
| p_resp_id | NUMBER | No* | Responsibility identifier |
| p_user_id | NUMBER | No* | Corresponds to the column USER_ID in table FND_USER, and identifies the Oracle Applications user. |
| p_login_id | NUMBER | No | Corresponds to the column LOGIN_ID in table FND_LOGINS, and identifies the login session. |

*Table C–32 Close Interaction IN Parameters*

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_interaction_rec | interaction_rec_type | Yes | Contains the elements that comprise the interaction record.<br>The following record parameters are always validated:<br>■ start_date_time<br>If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place.<br>■ end_date_time<br>If the *end_date_time* parameter is not supplied at the time that the *Close_Interaction* API is invoked, then *sysdate* is inserted in its place.<br>■ handler_id<br>■ outcome_id<br>■ result_id<br>■ reason_id<br>■ resource_id<br>■ party_id<br>■ source_code<br>■ source_code_id<br>See "Interaction Record Type" on page C-12 for the record specification. |

[1] The Application ID, Responsibility ID, and User ID determine which profile values are used as defaults. Those items marked with an asterisk also follow these guidelines.

The following table describes the OUT parameters associated with this API.

*Table C–33 Close Interaction OUT Parameters*

| Parameter | Data Type | Description |
|---|---|---|
| x_return_status | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_count | NUMBER | See Section C.1.2, "Standard OUT Parameters" for more information. |

*Table C–33 Close Interaction OUT Parameters*

| Parameter | Data Type | Description |
|---|---|---|
| x_msg_data | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |

# C.9 Counting APIs

The following are counting APIs:

- Get_InteractionActivityCount
- Get_InteractionCount

## C.9.1 Overview

The counting APIs are classified as selector methods. They return the count of an interaction or an activity based on filtering parameter values that are passed by the calling application.

*Table C–34 Counting APIs*

| Procedure | Description |
|---|---|
| Get_InteractionActivityCount | Retrieves the activity count from Activities table. |
| Get_InteractionCount | Retrieves the interaction count from Interaction table. |

## C.9.2 Get_InteractionActivityCount

This Get_InteractionActivityCount API retrieves the interaction and activity count from table JTF_IH_ACTIVITIES based on the input parameters.

### Procedure Specification

```
PROCEDURE Get_InteractionActivityCount
(
   p_api_version        in      number,
   p_init_msg_list      in      varchar2        default fnd_api.g_false,
   p_resp_appl_id       in      number          default null,
   p_resp_id            in      number          default null,
   p_user_id            in      number,
   p_login_id           in      number          default null,
   x_return_status      out     varchar2,
   x_msg_count          out     number,
```

```
   x_msg_data           out     varchar2,
   p_outcome_id         in      number,
   p_result_id          in      number,
   p_reason_id          in      number,
   p_script_id          in      number,
   p_media_id           in      number,
   x_activity_count     out     number
);
```

### Current Version

1.0

### Parameter Descriptions

The following table describes the IN parameters associated with this API.

***Table C–35 Get Interaction Activity Count IN Parameters***

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_api_version | NUMBER | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_init_msg_list | VARCHAR2 | No | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_resp_appl_id | NUMBER | No[1] | Application identifier |
| p_resp_id | NUMBER | No* | Responsibility identifier |
| p_user_id | NUMBER | Yes | Corresponds to the column USER_ID in table FND_USER, and identifies the Oracle Applications user. |
| p_login_id | NUMBER | No | Corresponds to the column LOGIN_ID in table FND_LOGINS, and identifies the login session. |
| p_outcome_id | NUMBER | Yes | Outcome identifier. The number corresponds to a certain outcome. |
| p_result_id | NUMBER | Yes | Result identifier. The number corresponds to a certain result. |
| p_reason_id | NUMBER | Yes | Reason identifier. The number corresponds to certain reasons. |
| p_script_id | NUMBER | No | User hook |

*Table C–35 Get Interaction Activity Count IN Parameters*

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_media_id | NUMBER | No | See "Media Item Record Type" on page C-15 for the record specification. |

[1] The Application ID, Responsibility ID, and User ID determine which profile values are used as defaults. Those items marked with an asterisk also follow these guidelines.

The following table describes the OUT parameters associated with this API.

*Table C–36 Get Interaction Activity Count OUT Parameters*

| Parameter | Data Type | Description |
|---|---|---|
| x_return_status | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_count | NUMBER | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_data | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_activity_count | NUMBER | Corresponds to the number of interactions and activities found that match the search criteria. |

## C.9.3 Get_InteractionCount

The Get_InteractionCount API retrieves the interaction count from table JTF_IH_INTERACTIONS based on the input parameters.

### Procedure Specification

```
PROCEDURE Get_InteractionCount
(
   p_api_version        in      number,
   p_init_msg_list      in      varchar2        default fnd_api.g_false,
   p_resp_appl_id       in      number          default null,
   p_resp_id            in      number          default null,
   p_user_id            in      number,
   p_login_id           in      number          default null,
   x_return_status      out     varchar2,
   x_msg_count          out     number,
   x_msg_data           out     varchar2,
   p_outcome_id         in      number,
   p_result_id          in      number,
```

```
   p_reason_id          in      number,
   p_attribute1         in      varchar2        default null,
   p_attribute2         in      varchar2        default null,
   p_attribute3         in      varchar2        default null,
   p_attribute4         in      varchar2        default null,
   p_attribute5         in      varchar2        default null,
   p_attribute6         in      varchar2        default null,
   p_attribute7         in      varchar2        default null,
   p_attribute8         in      varchar2        default null,
   p_attribute9         in      varchar2        default null,
   p_attribute10        in      varchar2        default null,
   p_attribute11        in      varchar2        default null,
   p_attribute12        in      varchar2        default null,
   p_attribute13        in      varchar2        default null,
   p_attribute14        in      varchar2        default null,
   p_attribute15        in      varchar2        default null,
   p_attribute_category in      varchar2        default null,
   x_interaction_count  out     number
);
```

### Current Version

```
1.0
```

### Parameter Descriptions

The following table describes the IN parameters associated with this API.

***Table C–37 Get Interaction Count IN Parameters***

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_api_version | NUMBER | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_init_msg_list | VARCHAR2 | Yes | See Section C.1.1, "Standard IN Parameters" for more information. |
| p_resp_appl_id | NUMBER | No[1] | Application identifier |
| p_resp_id | NUMBER | No* | Responsibility identifier |
| p_user_id | NUMBER | No* | Corresponds to the column USER_ID in table FND_USER, and identifies the Oracle Applications user. |
| p_login_id | NUMBER | No | Corresponds to the column LOGIN_ID in table FND_LOGINS, and identifies the login session. |

***Table C–37 Get Interaction Count IN Parameters***

| Parameter | Data Type | Required | Descriptions and Validations |
|---|---|---|---|
| p_outcome_id | NUMBER | No | Outcome identifier. The number corresponds to a certain outcome. |
| p_result_id | NUMBER | No | Result identifier. The number corresponds to a certain result. |
| p_reason_id | NUMBER | No | Reason identifier. The number corresponds to certain reasons. |
| p_attribute1 | VARCHAR2(150) | No[2] | Customer flex field segment. |
| p_attribute2 | VARCHAR2(150) | No** | Customer flex field segment. |
| p_attribute3 | VARCHAR2(150) | No** | Customer flex field segment. |
| p_attribute4 | VARCHAR2(150) | No** | Customer flex field segment. |
| p_attribute5 | VARCHAR2(150) | No** | Customer flex field segment. |
| p_attribute6 | VARCHAR2(150) | No** | Customer flex field segment. |
| p_attribute7 | VARCHAR2(150) | No** | Customer flex field segment. |
| p_attribute8 | VARCHAR2(150) | No** | Customer flex field segment. |
| p_attribute9 | VARCHAR2(150) | No** | Customer flex field segment. |
| p_attribute10 | VARCHAR2(150) | No** | Customer flex field segment. |
| p_attribute11 | VARCHAR2(150) | No** | Customer flex field segment. |
| p_attribute12 | VARCHAR2(150) | No** | Customer flex field segment. |
| p_attribute13 | VARCHAR2(150) | No** | Customer flex field segment. |
| p_attribute14 | VARCHAR2(150) | No** | Customer flex field segment. |
| p_attribute15 | VARCHAR2(150) | No** | Customer flex field segment. |
| p_attribute_ category | VARCHAR2(30) | No | |

[1] The application ID, responsibility ID, and user ID determine which profile values are used as defaults. Those items marked with an asterisk also follow these guidelines.

[2] You must pass in segment IDs for none or all descriptive flexfield columns that might be used in the descriptive flexfield. Those items marked with two asterisks also follow these guidelines.

The following table describes the OUT parameters associated with this API.

*Table C–38 Get Interaction Count OUT Parameter*

| Parameter | Data Type | Description |
|---|---|---|
| x_return_status | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_count | NUMBER | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_msg_data | VARCHAR2 | See Section C.1.2, "Standard OUT Parameters" for more information. |
| x_interaction_ count | NUMBER | Corresponds to the number of interactions found. |

## C.10 Messages and Notifications

The following APIs contained in package JTF_IH_PUB generate messages and notifications as required:

- Create_Interaction
- Open_MediaItem
- Add_MediaLifecycle
- Close_MediaItem
- Open_Interaction
- Update_Interaction
- Add_Activity
- Update_Activity
- Close_Interaction

**Note:** It is not required that all status notifications provide a number identifier along with the message, although, in many cases, it is provided.

## C.10.1 JTF_IH_PUB

### C.10.1.1 Create_Interaction

The following table lists the messages and notifications generated by the Create_Interaction API.

***Table C–39 Create Interaction Messages***

| Type | Name | Text |
|---|---|---|
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for party_id touchpoint1_ type is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for resource_id touchpoint1_ type is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for party_id touchpoint2_ type is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for resource_id touchpoint2_ type is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for handler_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for outcome_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for result_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for reason_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for action_item_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for action_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for script_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for source_code_id set and source_code not set is invalid. |

*Table C–39 Create Interaction Messages*

| Type | Name | Text |
|---|---|---|
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for source_code set and source_code_id not set is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for end_date_time is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for interaction_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for non_production_time_ amount. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for interaction is not active. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for activity_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for action_item_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for action_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for activity is not active. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Create_Interaction): The value <parameter value> for cust_account_id is invalid. |

### C.10.1.2 Open_MediaItem

The following table lists the messages and notifications generated by the Open_MediaItem API.

*Table C–40 Open Media Item Messages*

| Type | Name | Text |
|---|---|---|
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_MediaItem): The value <parameter value> for end_date_time is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB. Open_MediaItem): The value <parameter value> for media_item_type is invalid. |

### C.10.1.3 Update_MediaItem

The following table lists the messages and notifications generated by the Update_MediaItem API.

***Table C–41 Update Media Item Messages***

| Type | Name | Text |
|---|---|---|
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_MediaItem): The value <parameter value> for end_date_time is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB. Update_ MediaItem): The value <parameter value> for media_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB. Update_ MediaItem): The value <parameter value> for media_item_ type is invalid. |

### C.10.1.4 Add_MediaLifecycle

The following table lists the messages and notifications generated by the Add_MediaLifecycle API.

***Table C–42 Add Media Lifecycle Messages***

| Type | Name | Text |
|---|---|---|
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Add_ Medialifecycle): The value <parameter value> for end_ date_time is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB. Add_ Medialifecycle): The value <parameter value> for milcs_ code is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB. Add_ Medialifecycle): The value <parameter value> for milcs_ type_id is invalid. |

### C.10.1.5 Close_MediaItem

The following table lists the messages and notifications generated by the Close_MediaItem API.

*Table C–43 Close Media Item Messages*

| Type | Name | Text |
|---|---|---|
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_MediaItem): The value <parameter value> for end_date_time is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB. Close_MediaItem): The value <parameter value> for media_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB. Close_MediaItem): The value <parameter value> for media_item_type is invalid. |

### C.10.1.6 Open_Interaction

The following table lists the messages and notifications generated by the Open_Interaction API.

*Table C–44 Open Interaction Messages*

| Type | Name | Text |
|---|---|---|
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for party_id touchpoint1_ type is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for resource_id touchpoint1_ type is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for party_id touchpoint2_ type is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for resource_id touchpoint2_ type is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for handler_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for outcome_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for result_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for reason_id is invalid. |

*Table C–44 Open Interaction Messages*

| Type | Name | Text |
|---|---|---|
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for action_item_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for action_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for script_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for source_code_id set and source_code not set is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for source_code set and source_code_id not set is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for end_date_time is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for interaction_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for non_production_time_ amount. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for interaction is not active. |

### C.10.1.7 Update_Interaction

The following table lists the messages and notifications generated by the Update_Interaction API.

***Table C–45 Update Interaction Messages***

| Type | Name | Text |
|---|---|---|
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for party_id touchpoint1_ type is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for resource_id touchpoint1_ type is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for party_id touchpoint2_ type is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for resource_id touchpoint2_ type is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for handler_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for outcome_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for result_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for reason_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for action_item_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for action_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for script_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for source_code_id set and source_code not set is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for source_code set and source_code_id not set is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for end_date_time is invalid. |

*Table C–45 Update Interaction Messages*

| Type | Name | Text |
|---|---|---|
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for interaction_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for non_production_time_ amount. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for interaction is not active. |

### C.10.1.8 Add_Activity

The following table lists the messages and notifications generated by the Add_Activity API.

*Table C–46 Add Activity Messages*

| Type | Name | Text |
|---|---|---|
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Add_Activity): The value <parameter value> for outcome_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Add_Activity): The value <parameter value> for result_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Add_Activity): The value <parameter value> for reason_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Add_Activity): The value <parameter value> for action_item_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Add_Activity): The value <parameter value> for action_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Add_Activity): The value <parameter value> for source_code_id set and source_code not set is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Add_Activity): The value <parameter value> for source_code set and source_code_id not set is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Add_Activity): The value <parameter value> for end_date_time is invalid. |

*Table C–46 Add Activity Messages*

| Type | Name | Text |
|---|---|---|
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Add_Activity): The value <parameter value> for activity_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Add_Activity): The value <parameter value> for cust_account_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Add_Activity): The value <parameter value> for active is not active. |

### C.10.1.9 Update_Activity

The following table lists the messages and notifications generated by the Update_Activity API.

*Table C–47 Update Activity Messages*

| Type | Name | Text |
|---|---|---|
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Activity): The value <parameter value> for outcome_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Activity): The value <parameter value> for result_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Activity): The value <parameter value> for reason_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Activity): The value <parameter value> for action_item_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Activity): The value <parameter value> for action_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Activity): The value <parameter value> for source_code_id set and source_code not set is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Activity): The value <parameter value> for source_code set and source_code_id not set is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Activity): The value <parameter value> for end_date_time is invalid. |

*Table C–47 Update Activity Messages*

| Type | Name | Text |
|---|---|---|
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Activity): The value <parameter value> for activity_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Activity): The value <parameter value> for cust_account_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Update_Activity): The value <parameter value> for active is not active. |

### C.10.1.10 Close_Interaction

The following table lists the messages and notifications generated by the Close_Interaction API.

*Table C–48 Close Interaction Messages*

| Type | Name | Text |
|---|---|---|
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for party_id touchpoint1_ type is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for resource_id touchpoint1_ type is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for party_id touchpoint2_ type is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for resource_id touchpoint2_ type is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for handler_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for outcome_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for result_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for reason_id is invalid. |

*Table C–48 Close Interaction Messages*

| Type | Name | Text |
|---|---|---|
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for action_item_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for action_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for script_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for source_code_id set and source_code not set is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for source_code set and source_code_id not set is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for end_date_time is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for interaction_id is invalid. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for non_production_time_ amount. |
| E | JTF_API_ALL_INVALID_ARGUMENT | API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for interaction is not active. |

# C.11 Sample Code

This section contains SQL scripts that call the following types of Interaction History public APIs contained in the JTF_IH_PUB package and insert values as required:

- Non-cached Creation APIs
- Cached Creation APIs
- Counting APIs

## C.11.1 Non-cached Creation APIs

The SQL scripts in this section build a customer interaction record by calling the non-cached creation APIs in succession and by providing them with the required values.

### C.11.1.1 Create_MediaItem

This script calls the Create_MediaItem API and provides the following values using the Create_MediaItem IN parameters:

- p_api_version: **1.0**
- p_init_msg_list: **T** indicates that this parameter is set to true
- p_commit: **T** indicates that this parameter is set to true
- p_resp_appl_id: the application identifier is **690**
- p_resp_id: the responsibility identifier is **-1**
- p_user_id: the user identifier is **2877**
- p_login_id: the login identifier is **-1**
- p_media: the record type, media_rec_type contained in the JTF_IH_PUB and identified here as **l_media**
- p_mlcs: the record type, mlcs_tbl_type contained in the JTF_IH_PUB and identified here as **l_mlcs**

```
set serveroutput on;

declare
 l_return_status VARCHAR2(30);
 l_msg_count NUMBER;
 l_msg_data VARCHAR2(200);

 l_interaction_rec APPS.JTF_IH_PUB.interaction_rec_type;
 l_interaction_id  NUMBER;
 l_jtf_note_id     NUMBER;
 m_count NUMBER := 0;
 m_active VARCHAR2(1);
 p_interaction_id NUMBER;

 l_activity_rec APPS.JTF_IH_PUB.activity_rec_type;
 l_activity_id_1  NUMBER;
 l_activity_id_2  NUMBER;
 m_activitycount NUMBER := 0;
 m_activityactive VARCHAR2(1);
 l_startdatecheck VARCHAR2(30);
 l_enddatecheck VARCHAR2(30);
 l_data                VARCHAR2(8000);
```

```
 l_msg_index_out        NUMBER;
 xInteraction_Count     NUMBER := 2;
 cInteraction_Count     VARCHAR2(80);
 xparty_id              NUMBER := 1000;
 cparty_id              VARCHAR2(80);
 xresource_id           NUMBER := 10039;
 cresource_id           VARCHAR2(80);
 status                 NUMBER;
 end_time_runDATE;
 begin_time_runDATE;
 total_time_runNUMBER;
 nDbl               NUMBER := 1;
 l_activity_tbl     JTF_IH_PUB.activity_tbl_type;


 l_media APPS.JTF_IH_PUB.media_rec_type;
 l_media_lc_rec APPS.JTF_IH_PUB.media_lc_rec_type;
 l_media_id         NUMBER;
 l_mlcs APPS.JTF_IH_PUB.mlcs_tbl_type;
 l_milcs_id         NUMBER;

 l_activity_count    NUMBER;
 l_interaction_count    NUMBER;
 l_interaction_id_2      NUMBER;
begin
    DBMS_SESSION.SET_SQL_TRACE(TRUE);
    DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.TRACE_ALL_CALLS+
        DBMS_TRACE.TRACE_ALL_EXCEPTIONS+
        DBMS_TRACE.TRACE_RESUME);

  -- obtain loop parameter values
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Interaction History (IH) Test Script');
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Author: Author's Name Here.');
  DBMS_OUTPUT.PUT_LINE('Version 1.0 - Initial Version - 06.27.2001');
  DBMS_OUTPUT.PUT_LINE(' ');
  xInteraction_Count := &xInteraction_Count;
  xparty_id := &xparty_id;
  xresource_id := &xresource_id;
  -- begin major loop
  begin_time_run := sysdate;
  DBMS_OUTPUT.PUT_LINE('Start Time := ' || TO_CHAR(begin_time_run, 'DD-MON-YYYY:HH:MI:SS'));

  -- Test: Create_MediaItem,
```

```
l_media.media_id := NULL;
l_media.source_id := NULL;
l_media.direction := NULL;
--l_media.duration := null;
--l_media.end_date_time := null;
l_media.interaction_performed := NULL;
l_media.start_date_time := SYSDATE;
l_media.media_data  := 'N';
l_media.source_item_id := NULL;
l_media.media_item_type := 'VM';
l_media.media_item_ref := NULL;
l_media.media_abandon_flag := NULL;
l_media.media_transferred_flag := NULL;

JTF_IH_PUB.Create_MediaItem
                (
                1.0,
                'T',
                'T',
                690,
                -1,
                2877,
                -1,
                l_return_status,
                l_msg_count,
                l_msg_data,
                l_media,
                l_mlcs
                );
                IF l_return_status != 'S' THEN
                --Display all the error messages
                   FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                           dbms_output.put_line(j);
                       l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                           p_encoded=>'F');
                       DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_msg_data);
                   END LOOP;
                END IF;

DBMS_OUTPUT.PUT_LINE('PAST Create_MediaItem Method - II');
DBMS_OUTPUT.PUT_LINE('Create_MediaItem Method - II - l_return_status: '||l_return_status);
```

### C.11.1.2 Create_MediaLifecycle

This script calls the Create_MediaLifecycle API and provides the following values using the Create_MediaLifecycle IN parameters:

- p_api_version: **1.0**
- p_init_msg_list: **T** indicates that this parameter is set to true
- p_commit: **T** indicates that this parameter is set to true
- p_resp_appl_id: the application identifier is **690**
- p_resp_id: the responsibility identifier is **-1**
- p_user_id: the user identifier is **2877**
- p_login_id: the login identifier is **-1**
- p_media_lc_rec: the record type, media_lc_rec_type contained in the JTF_IH_PUB and identified here as **l_media_lc_rec**

```
set serveroutput on;

declare
 l_return_status VARCHAR2(30);
 l_msg_count NUMBER;
 l_msg_data VARCHAR2(200);

 l_interaction_rec APPS.JTF_IH_PUB.interaction_rec_type;
 l_interaction_id  NUMBER;
 l_jtf_note_id     NUMBER;
 m_count NUMBER := 0;
 m_active VARCHAR2(1);
 p_interaction_id NUMBER;

 l_activity_rec APPS.JTF_IH_PUB.activity_rec_type;
 l_activity_id_1  NUMBER;
 l_activity_id_2  NUMBER;
 m_activitycount NUMBER := 0;
 m_activityactive VARCHAR2(1);
 l_startdatecheck VARCHAR2(30);
 l_enddatecheck VARCHAR2(30);
 l_data                 VARCHAR2(8000);
 l_msg_index_out        NUMBER;
 xInteraction_Count     NUMBER := 2;
 cInteraction_Count     VARCHAR2(80);
```

```
 xparty_id              NUMBER := 1000;
 cparty_id              VARCHAR2(80);
 xresource_id           NUMBER := 10039;
 cresource_id           VARCHAR2(80);
 status                 NUMBER;
 end_time_runDATE;
 begin_time_runDATE;
 total_time_runNUMBER;
 nDbl               NUMBER := 1;
 l_activity_tbl     JTF_IH_PUB.activity_tbl_type;


 l_media APPS.JTF_IH_PUB.media_rec_type;
 l_media_lc_rec APPS.JTF_IH_PUB.media_lc_rec_type;
 l_media_id         NUMBER;
 l_mlcs APPS.JTF_IH_PUB.mlcs_tbl_type;
 l_milcs_id         NUMBER;

 l_activity_count    NUMBER;
 l_interaction_count    NUMBER;
 l_interaction_id_2      NUMBER;
begin
    DBMS_SESSION.SET_SQL_TRACE(TRUE);
    DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.TRACE_ALL_CALLS+
        DBMS_TRACE.TRACE_ALL_EXCEPTIONS+
        DBMS_TRACE.TRACE_RESUME);

  -- obtain loop parameter values
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Interaction History (IH) Test Script');
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Author: Author's Name Here.');
  DBMS_OUTPUT.PUT_LINE('Version 1.0 - Initial Version - 06.27.2001');
  DBMS_OUTPUT.PUT_LINE(' ');
  xInteraction_Count := &xInteraction_Count;
  xparty_id := &xparty_id;
  xresource_id := &xresource_id;
  -- begin major loop
  begin_time_run := sysdate;
  DBMS_OUTPUT.PUT_LINE('Start Time := ' || TO_CHAR(begin_time_run, 'DD-MON-YYYY:HH:MI:SS'));


l_media_lc_rec.media_id := l_media_id;
l_media_lc_rec.start_date_time := sysdate;
l_media_lc_rec.handler_id := 690;
```

```
l_media_lc_rec.resource_id := xresource_id;
l_media_lc_rec.milcs_type_id := 9;
    jtf_ih_pub.Create_MediaLifecycle(
   1.0,
   'T',
   'T',
   690,
   -1,
   2877,
   -1,
   l_return_status,
   l_msg_count,
   l_msg_data,
   l_media_lc_rec);
        IF l_return_status != 'S' THEN
                --Display all the error messages
                   FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                           dbms_output.put_line(j);
                       l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                           p_encoded=>'F');
                       DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_msg_data);
                   END LOOP;
        END IF;


DBMS_OUTPUT.PUT_LINE('PAST Create_MediaLifecycle ');
DBMS_OUTPUT.PUT_LINE('Create_MediaLifecycle - l_return_status: '||l_return_status);
```

### C.11.1.3 Create_Interaction

This script calls the Create_Interaction API and provides the following values using the Create_Interaction IN parameters:

- p_api_version: **1.0**
- p_init_msg_list: **T** indicates that this parameter is set to true
- p_commit: **T** indicates that this parameter is set to true
- p_resp_appl_id: the application identifier is **690**
- p_resp_id: the responsibility identifier is **-1**
- p_user_id: the user identifier is **2877**
- p_login_id: the login identifier is **-1**

- p_interaction_rec: the record type, interaction_rec_type contained in the JTF_IH_PUB and identified here as **l_interaction_rec**
- p_activity: the record type, activity_tbl_type contained in the JTF_IH_PUB and identified here as **l_activity_tbl**

```
set serveroutput on;

declare
 l_return_status VARCHAR2(30);
 l_msg_count NUMBER;
 l_msg_data VARCHAR2(200);

 l_interaction_rec APPS.JTF_IH_PUB.interaction_rec_type;
 l_interaction_id  NUMBER;
 l_jtf_note_id     NUMBER;
 m_count NUMBER := 0;
 m_active VARCHAR2(1);
 p_interaction_id NUMBER;

 l_activity_rec APPS.JTF_IH_PUB.activity_rec_type;
 l_activity_id_1  NUMBER;
 l_activity_id_2  NUMBER;
 m_activitycount NUMBER := 0;
 m_activityactive VARCHAR2(1);
 l_startdatecheck VARCHAR2(30);
 l_enddatecheck VARCHAR2(30);
 l_data                 VARCHAR2(8000);
 l_msg_index_out        NUMBER;
 xInteraction_Count     NUMBER := 2;
 cInteraction_Count     VARCHAR2(80);
 xparty_id              NUMBER := 1000;
 cparty_id              VARCHAR2(80);
 xresource_id           NUMBER := 10039;
 cresource_id           VARCHAR2(80);
 status                 NUMBER;
 end_time_runDATE;
 begin_time_runDATE;
 total_time_runNUMBER;
 nDbl               NUMBER := 1;
 l_activity_tbl     JTF_IH_PUB.activity_tbl_type;


 l_media APPS.JTF_IH_PUB.media_rec_type;
```

```
 l_media_lc_rec APPS.JTF_IH_PUB.media_lc_rec_type;
 l_media_id         NUMBER;
 l_mlcs APPS.JTF_IH_PUB.mlcs_tbl_type;
 l_milcs_id         NUMBER;

 l_activity_count    NUMBER;
 l_interaction_count    NUMBER;
 l_interaction_id_2      NUMBER;
begin
    DBMS_SESSION.SET_SQL_TRACE(TRUE);
    DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.TRACE_ALL_CALLS+
        DBMS_TRACE.TRACE_ALL_EXCEPTIONS+
        DBMS_TRACE.TRACE_RESUME);

  -- obtain loop parameter values
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Interaction History (IH) Test Script');
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Author: Author's Name Here.');
  DBMS_OUTPUT.PUT_LINE('Version 1.0 - Initial Version - 06.27.2001');
  DBMS_OUTPUT.PUT_LINE(' ');
  xInteraction_Count := &xInteraction_Count;
  xparty_id := &xparty_id;
  xresource_id := &xresource_id;
  -- begin major loop
  begin_time_run := sysdate;
  DBMS_OUTPUT.PUT_LINE('Start Time := ' || TO_CHAR(begin_time_run, 'DD-MON-YYYY:HH:MI:SS'));
FOR K in 1..xInteraction_Count LOOP

        --Create interaction

        l_interaction_rec.interaction_id := NULL;
        l_interaction_rec.reference_form := 'Test for Create Interaction';
        l_interaction_rec.follow_up_action := 'No FollowUp';
        --  l_interaction_rec.duration := 15;
        --  l_interaction_rec.start_date_time := to_date('23-OCT-2000', 'DD-MON-YYYY');
        l_interaction_rec.start_date_time := sysdate;
        --  l_interaction_rec.end_date_time := to_date('31-OCT-2000', 'DD-MON-YYYY');
        --    l_interaction_rec.end_date_time := NULL;
        --    l_interaction_rec.inter_interaction_duration := 12;
        l_interaction_rec.non_productive_time_amount := NULL;
        l_interaction_rec.preview_time_amount := 2;
        l_interaction_rec.productive_time_amount := 12;
        l_interaction_rec.wrapup_time_amount := 1;
        l_interaction_rec.handler_id := 690;  -- Customers: please validate for your environment
```

```
l_interaction_rec.script_id := NULL;
l_interaction_rec.outcome_id := 4;
l_interaction_rec.result_id := 2;
l_interaction_rec.reason_id := 2;
l_interaction_rec.resource_id := xresource_id; -- A environment
l_interaction_rec.party_id := xparty_id; -- B environment
l_interaction_rec.parent_id := NULL;
--
--add an activities
--
for idx in 1..xparty_id loop
    l_activity_rec.activity_id := NULL;
    l_activity_rec.duration := NULL;
    l_activity_rec.cust_account_id := NULL; --checked
    l_activity_rec.cust_org_id := null;
    l_activity_rec.role := 1;
    l_activity_rec.script_trans_id := NULL;

    l_activity_rec.start_date_time := sysdate;
    l_activity_rec.end_date_time := NULL;

    l_activity_rec.media_id := NULL;
    l_activity_rec.action_item_id := 17;
    l_activity_rec.interaction_id := NULL;
    l_activity_rec.outcome_id := 7;
    l_activity_rec.result_id := 7;
    l_activity_rec.reason_id := 8;
    l_activity_rec.description := 'test Activity 1';
    l_activity_rec.interaction_action_type := 'unknown';

    if nDbl = 1 then
        l_activity_rec.action_id := 14;
        nDbl := 2;
    else
        l_activity_rec.action_id := 13;
        nDbl := 1;
    end if;

    l_activity_tbl(idx) := l_activity_rec;
end loop;
APPS.JTF_IH_PUB.Create_Interaction
    (
    1.0,
    'T',
    'T',
```

```
            690,
            -1,
            2877,
            -1,
            l_return_status,
            l_msg_count,
            l_msg_data,
            l_interaction_rec,
            l_activity_tbl
            );
            IF l_return_status != 'S' THEN
                --Display all the error messages
                   FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                           dbms_output.put_line(j);
                       l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                           p_encoded=>'F');
                       DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_msg_data);
                   END LOOP;
            END IF;
END LOOP;
        DBMS_OUTPUT.PUT_LINE('PAST Create_Interaction ');
```

## C.11.2 Cached Creation APIs

The SQL scripts in this section build a customer interaction record by calling the cached creation APIs in succession and by providing them with the required values.

### C.11.2.1 Open_MediaItem

This script calls the Open_MediaItem API and provides the following values using the Open_MediaItem IN parameters:

- p_api_version: **1.0**
- p_init_msg_list: **T** indicates that this parameter is set to true
- p_commit: **T** indicates that this parameter is set to true
- p_resp_appl_id: the application identifier is **690**
- p_resp_id: the responsibility identifier is **-1**
- p_user_id: the user identifier is **2877**
- p_login_id: the login identifier is **-1**

- p_media: the record type, media_rec_type contained in the JTF_IH_PUB and identified here as **l_media**
- p_media_rec: the record type, mlcs_tbl_type contained in the JTF_IH_PUB and identified here as **l_mlcs**

```
set serveroutput on;

declare
 l_return_status VARCHAR2(30);
 l_msg_count NUMBER;
 l_msg_data VARCHAR2(200);

 l_interaction_rec APPS.JTF_IH_PUB.interaction_rec_type;
 l_interaction_id  NUMBER;
 l_jtf_note_id     NUMBER;
 m_count NUMBER := 0;
 m_active VARCHAR2(1);
 p_interaction_id NUMBER;

 l_activity_rec APPS.JTF_IH_PUB.activity_rec_type;
 l_activity_id_1  NUMBER;
 l_activity_id_2  NUMBER;
 m_activitycount NUMBER := 0;
 m_activityactive VARCHAR2(1);
 l_startdatecheck VARCHAR2(30);
 l_enddatecheck VARCHAR2(30);
 l_data                 VARCHAR2(8000);
 l_msg_index_out        NUMBER;
 xInteraction_Count     NUMBER := 2;
 cInteraction_Count     VARCHAR2(80);
 xparty_id              NUMBER := 1000;
 cparty_id              VARCHAR2(80);
 xresource_id           NUMBER := 10039;
 cresource_id           VARCHAR2(80);
 status                 NUMBER;
 end_time_runDATE;
 begin_time_runDATE;
 total_time_runNUMBER;
 nDbl               NUMBER := 1;
 l_activity_tbl     JTF_IH_PUB.activity_tbl_type;


 l_media APPS.JTF_IH_PUB.media_rec_type;
```

```
 l_media_lc_rec APPS.JTF_IH_PUB.media_lc_rec_type;
 l_media_id         NUMBER;
 l_mlcs APPS.JTF_IH_PUB.mlcs_tbl_type;
 l_milcs_id         NUMBER;

 l_activity_count    NUMBER;
 l_interaction_count    NUMBER;
 l_interaction_id_2      NUMBER;
begin
    DBMS_SESSION.SET_SQL_TRACE(TRUE);
    DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.TRACE_ALL_CALLS+
        DBMS_TRACE.TRACE_ALL_EXCEPTIONS+
        DBMS_TRACE.TRACE_RESUME);

  -- obtain loop parameter values
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Interaction History (IH) Test Script');
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Author: Author's Name Here.');
  DBMS_OUTPUT.PUT_LINE('Version 1.0 - Initial Version - 06.27.2001');
  DBMS_OUTPUT.PUT_LINE(' ');
  xInteraction_Count := &xInteraction_Count;
  xparty_id := &xparty_id;
  xresource_id := &xresource_id;
  -- begin major loop
  begin_time_run := sysdate;
  DBMS_OUTPUT.PUT_LINE('Start Time := ' || TO_CHAR(begin_time_run, 'DD-MON-YYYY:HH:MI:SS'));


    JTF_IH_PUB.Open_MediaItem
                (
                1.0,
                'T',
                'T',
                690,
                -1,
                2877,
                -1,
                l_return_status,
                l_msg_count,
                l_msg_data,
                l_media,
                l_media_id
                );
    if l_return_status != 'S' then
```

```
/*   JTF_IH_PUB.Add_MediaLifecycle
            (
               1.0,
               'T',
               'T',
               690,
               -1,
               2877,
               -1,
               l_return_status,
               l_msg_count,
               l_msg_data,
               l_mlcs,
               l_milcs_id);

            if l_return_status != 'S' then
                   FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                        dbms_output.put_line(j);
                    l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                           p_encoded=>'F');
                       DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_msg_data);
                   END LOOP;
            end if;
    else*/
                   FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                        dbms_output.put_line(j);
                    l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                           p_encoded=>'F');
                       DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_msg_data);
                   END LOOP;
    end if;
    dbms_output.put_line(l_return_status);
    dbms_output.put_line(l_msg_count);
    dbms_output.put_line(l_media_id);
end;
```

### C.11.2.2 Update_MediaItem

This script calls the Update_MediaItem API and provides the following values using the Update_MediaItem IN parameters:

- p_api_version: **1.0**
- p_init_msg_list: **T** indicates that this parameter is set to true

- p_commit: **T** indicates that this parameter is set to true
- p_resp_appl_id: the application identifier is **690**
- p_resp_id: the responsibility identifier is **-1**
- p_user_id: the user identifier is **2877**
- p_login_id: the login identifier is **-1**
- p_media_rec: the record type, media_rec_type contained in the JTF_IH_PUB and identified here as **l_media**. The the media_id field of this record type has been modified to **l_media_id**, and the direction field has been modified as **Updated media_id**.

```
set serveroutput on;

declare
 l_return_status VARCHAR2(30);
 l_msg_count NUMBER;
 l_msg_data VARCHAR2(200);

 l_interaction_rec APPS.JTF_IH_PUB.interaction_rec_type;
 l_interaction_id  NUMBER;
 l_jtf_note_id     NUMBER;
 m_count NUMBER := 0;
 m_active VARCHAR2(1);
 p_interaction_id NUMBER;

 l_activity_rec APPS.JTF_IH_PUB.activity_rec_type;
 l_activity_id_1  NUMBER;
 l_activity_id_2  NUMBER;
 m_activitycount NUMBER := 0;
 m_activityactive VARCHAR2(1);
 l_startdatecheck VARCHAR2(30);
 l_enddatecheck VARCHAR2(30);
 l_data                 VARCHAR2(8000);
 l_msg_index_out        NUMBER;
 xInteraction_Count     NUMBER := 2;
 cInteraction_Count     VARCHAR2(80);
 xparty_id              NUMBER := 1000;
 cparty_id              VARCHAR2(80);
 xresource_id           NUMBER := 10039;
 cresource_id           VARCHAR2(80);
 status                 NUMBER;
```

```
 end_time_runDATE;
 begin_time_runDATE;
 total_time_runNUMBER;
 nDbl               NUMBER := 1;
 l_activity_tbl     JTF_IH_PUB.activity_tbl_type;


 l_media APPS.JTF_IH_PUB.media_rec_type;
 l_media_lc_rec APPS.JTF_IH_PUB.media_lc_rec_type;
 l_media_id         NUMBER;
 l_mlcs APPS.JTF_IH_PUB.mlcs_tbl_type;
 l_milcs_id         NUMBER;

 l_activity_count    NUMBER;
 l_interaction_count    NUMBER;
 l_interaction_id_2      NUMBER;
begin
    DBMS_SESSION.SET_SQL_TRACE(TRUE);
    DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.TRACE_ALL_CALLS+
        DBMS_TRACE.TRACE_ALL_EXCEPTIONS+
        DBMS_TRACE.TRACE_RESUME);

  -- obtain loop parameter values
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Interaction History (IH) Test Script');
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Author: Author's Name Here.');
  DBMS_OUTPUT.PUT_LINE('Version 1.0 - Initial Version - 06.27.2001');
  DBMS_OUTPUT.PUT_LINE(' ');
  xInteraction_Count := &xInteraction_Count;
  xparty_id := &xparty_id;
  xresource_id := &xresource_id;
  -- begin major loop
  begin_time_run := sysdate;
  DBMS_OUTPUT.PUT_LINE('Start Time := ' || TO_CHAR(begin_time_run, 'DD-MON-YYYY:HH:MI:SS'));


FOR K in 1..xInteraction_Count LOOP

l_media.media_id := l_media_id;
l_media.direction := 'Updated Media_ID';

    JTF_IH_PUB.Update_MediaItem
                (
                1.0,
```

```
                'T',
                'T',
                690,
                -1,
                2877,
                -1,
                l_return_status,
                l_msg_count,
                l_msg_data,
                l_media
                );
                IF l_return_status != 'S' THEN
                --Display all the error messages
                   FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                           dbms_output.put_line(j);
                       l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                           p_encoded=>'F');
                       DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_msg_data);
                   END LOOP;
                END IF;

DBMS_OUTPUT.PUT_LINE('PAST Update_MediaItem ');
DBMS_OUTPUT.PUT_LINE('Update_MediaItem - l_return_status: '||l_return_status);
```

### C.11.2.3 Add_MediaLifecycle

This script calls the Add_MediaLifecycle API and provides the following values using the Add_MediaLifecycle IN parameters:

- p_api_version: **1.0**
- p_init_msg_list: **T** indicates that this parameter is set to true
- p_commit: **T** indicates that this parameter is set to true
- p_resp_appl_id: the application identifier is **690**
- p_resp_id: the responsibility identifier is **-1**
- p_user_id: the user identifier is **2877**
- p_login_id: the login identifier is **-1**
- l_media_lc_rec: the record type, media_lc_rec_type contained in the JTF_IH_PUB and identified here as **l_media_lc_rec**

```
set serveroutput on;

declare
 l_return_status VARCHAR2(30);
 l_msg_count NUMBER;
 l_msg_data VARCHAR2(200);

 l_interaction_rec APPS.JTF_IH_PUB.interaction_rec_type;
 l_interaction_id  NUMBER;
 l_jtf_note_id     NUMBER;
 m_count NUMBER := 0;
 m_active VARCHAR2(1);
 p_interaction_id NUMBER;

 l_activity_rec APPS.JTF_IH_PUB.activity_rec_type;
 l_activity_id_1  NUMBER;
 l_activity_id_2  NUMBER;
 m_activitycount NUMBER := 0;
 m_activityactive VARCHAR2(1);
 l_startdatecheck VARCHAR2(30);
 l_enddatecheck VARCHAR2(30);
 l_data                 VARCHAR2(8000);
 l_msg_index_out        NUMBER;
 xInteraction_Count     NUMBER := 2;
 cInteraction_Count     VARCHAR2(80);
 xparty_id              NUMBER := 1000;
 cparty_id              VARCHAR2(80);
 xresource_id           NUMBER := 10039;
 cresource_id           VARCHAR2(80);
 status                 NUMBER;
 end_time_runDATE;
 begin_time_runDATE;
 total_time_runNUMBER;
 nDbl               NUMBER := 1;
 l_activity_tbl     JTF_IH_PUB.activity_tbl_type;


 l_media APPS.JTF_IH_PUB.media_rec_type;
 l_media_lc_rec APPS.JTF_IH_PUB.media_lc_rec_type;
 l_media_id         NUMBER;
 l_mlcs APPS.JTF_IH_PUB.mlcs_tbl_type;
 l_milcs_id         NUMBER;

 l_activity_count    NUMBER;
 l_interaction_count    NUMBER;
```

```
 l_interaction_id_2      NUMBER;
begin
    DBMS_SESSION.SET_SQL_TRACE(TRUE);
    DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.TRACE_ALL_CALLS+
        DBMS_TRACE.TRACE_ALL_EXCEPTIONS+
        DBMS_TRACE.TRACE_RESUME);

  -- obtain loop parameter values
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Interaction History (IH) Test Script');
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Author: Author's Name Here.');
  DBMS_OUTPUT.PUT_LINE('Version 1.0 - Initial Version - 06.27.2001');
  DBMS_OUTPUT.PUT_LINE(' ');
  xInteraction_Count := &xInteraction_Count;
  xparty_id := &xparty_id;
  xresource_id := &xresource_id;
  -- begin major loop
  begin_time_run := sysdate;
  DBMS_OUTPUT.PUT_LINE('Start Time := ' || TO_CHAR(begin_time_run, 'DD-MON-YYYY:HH:MI:SS'));

JTF_IH_Pub.Add_MediaLifecycle
(
1.0,
'T',
'T',
680,
-1,
2877,
-1,
l_return_status,
l_msg_count,
l_msg_data,
l_media_lc_rec,
l_milcs_id);
    IF l_return_status != 'S' THEN
                --Display all the error messages
                   FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                           dbms_output.put_line(j);
                       l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                           p_encoded=>'F');
                       DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_msg_data);
                   END LOOP;
    END IF;
```

```
DBMS_OUTPUT.PUT_LINE('PAST Add_MediaLifecycle ');
DBMS_OUTPUT.PUT_LINE('Add_MediaLifecycle - l_return_status: '||l_return_status);
```

### C.11.2.4 Update_MediaLifecycle

This script calls the Update_MediaLifecycle API and provides the following values using the Update_MediaLifecycle IN parameters:

- p_api_version: **1.0**
- p_init_msg_list: **T** indicates that this parameter is set to true
- p_commit: **T** indicates that this parameter is set to true
- p_resp_appl_id: the application identifier is **690**
- p_resp_id: the responsibility identifier is **-1**
- p_user_id: the user identifier is **2877**
- p_login_id: the login identifier is **-1**
- l_media_lc_rec: the record type, media_lc_rec_type contained in the JTF_IH_PUB and identified here as **l_media_lc_rec**. The milcs_id field of this record type has been modified as **l_milcs_id**.

```
set serveroutput on;

declare
 l_return_status VARCHAR2(30);
 l_msg_count NUMBER;
 l_msg_data VARCHAR2(200);

 l_interaction_rec APPS.JTF_IH_PUB.interaction_rec_type;
 l_interaction_id  NUMBER;
 l_jtf_note_id     NUMBER;
 m_count NUMBER := 0;
 m_active VARCHAR2(1);
 p_interaction_id NUMBER;

 l_activity_rec APPS.JTF_IH_PUB.activity_rec_type;
 l_activity_id_1  NUMBER;
 l_activity_id_2  NUMBER;
 m_activitycount NUMBER := 0;
 m_activityactive VARCHAR2(1);
 l_startdatecheck VARCHAR2(30);
```

```
 l_enddatecheck VARCHAR2(30);
 l_data                 VARCHAR2(8000);
 l_msg_index_out        NUMBER;
 xInteraction_Count     NUMBER := 2;
 cInteraction_Count     VARCHAR2(80);
 xparty_id              NUMBER := 1000;
 cparty_id              VARCHAR2(80);
 xresource_id           NUMBER := 10039;
 cresource_id           VARCHAR2(80);
 status                 NUMBER;
 end_time_runDATE;
 begin_time_runDATE;
 total_time_runNUMBER;
 nDbl               NUMBER := 1;
 l_activity_tbl     JTF_IH_PUB.activity_tbl_type;


 l_media APPS.JTF_IH_PUB.media_rec_type;
 l_media_lc_rec APPS.JTF_IH_PUB.media_lc_rec_type;
 l_media_id         NUMBER;
 l_mlcs APPS.JTF_IH_PUB.mlcs_tbl_type;
 l_milcs_id         NUMBER;

 l_activity_count    NUMBER;
 l_interaction_count    NUMBER;
 l_interaction_id_2      NUMBER;
begin
    DBMS_SESSION.SET_SQL_TRACE(TRUE);
    DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.TRACE_ALL_CALLS+
        DBMS_TRACE.TRACE_ALL_EXCEPTIONS+
        DBMS_TRACE.TRACE_RESUME);

  -- obtain loop parameter values
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Interaction History (IH) Test Script');
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Author: Author's Name Here.');
  DBMS_OUTPUT.PUT_LINE('Version 1.0 - Initial Version - 06.27.2001');
  DBMS_OUTPUT.PUT_LINE(' ');
  xInteraction_Count := &xInteraction_Count;
  xparty_id := &xparty_id;
  xresource_id := &xresource_id;
  -- begin major loop
  begin_time_run := sysdate;
  DBMS_OUTPUT.PUT_LINE('Start Time := ' || TO_CHAR(begin_time_run, 'DD-MON-YYYY:HH:MI:SS'));
```

```
l_media_lc_rec.milcs_id := l_milcs_id;
JTF_IH_PUB.Update_MediaLifecycle
(
1.0,
'T',
'T',
690,
-1,
2877,
-1,
l_return_status,
l_msg_count,
l_msg_data,
l_media_lc_rec);
    IF l_return_status != 'S' THEN
                --Display all the error messages
                   FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                           dbms_output.put_line(j);
                       l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                           p_encoded=>'F');
                       DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_msg_data);
                   END LOOP;
    END IF;

DBMS_OUTPUT.PUT_LINE('PAST Update_MediaLifecycle ');
DBMS_OUTPUT.PUT_LINE('Update_MediaLifecycle - l_return_status: '||l_return_status);
```

### C.11.2.5 Close_MediaItem

This script calls the Close_MediaItem API and provides the following values using the Close_MediaItem IN parameters:

- p_api_version: **1.0**
- p_init_msg_list: **T** indicates that this parameter is set to true
- p_commit: **T** indicates that this parameter is set to true
- p_resp_appl_id: the application identifier is **690**
- p_resp_id: the responsibility identifier is **-1**
- p_user_id: the user identifier is **2877**
- p_login_id: the login identifier is **-1**

- p_media_rec: the record type, media_rec_type contained in the JTF_IH_PUB and identified here as **l_media**. The media_id field of this record type has been modified as **l_media_id**.

```
set serveroutput on;

declare
 l_return_status VARCHAR2(30);
 l_msg_count NUMBER;
 l_msg_data VARCHAR2(200);

 l_interaction_rec APPS.JTF_IH_PUB.interaction_rec_type;
 l_interaction_id  NUMBER;
 l_jtf_note_id     NUMBER;
 m_count NUMBER := 0;
 m_active VARCHAR2(1);
 p_interaction_id NUMBER;

 l_activity_rec APPS.JTF_IH_PUB.activity_rec_type;
 l_activity_id_1  NUMBER;
 l_activity_id_2  NUMBER;
 m_activitycount NUMBER := 0;
 m_activityactive VARCHAR2(1);
 l_startdatecheck VARCHAR2(30);
 l_enddatecheck VARCHAR2(30);
 l_data                 VARCHAR2(8000);
 l_msg_index_out        NUMBER;
 xInteraction_Count     NUMBER := 2;
 cInteraction_Count     VARCHAR2(80);
 xparty_id              NUMBER := 1000;
 cparty_id              VARCHAR2(80);
 xresource_id           NUMBER := 10039;
 cresource_id           VARCHAR2(80);
 status                 NUMBER;
 end_time_runDATE;
 begin_time_runDATE;
 total_time_runNUMBER;
 nDbl               NUMBER := 1;
 l_activity_tbl     JTF_IH_PUB.activity_tbl_type;


 l_media APPS.JTF_IH_PUB.media_rec_type;
 l_media_lc_rec APPS.JTF_IH_PUB.media_lc_rec_type;
 l_media_id         NUMBER;
 l_mlcs APPS.JTF_IH_PUB.mlcs_tbl_type;
 l_milcs_id         NUMBER;
```

```
 l_activity_count    NUMBER;
 l_interaction_count    NUMBER;
 l_interaction_id_2      NUMBER;
begin
    DBMS_SESSION.SET_SQL_TRACE(TRUE);
    DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.TRACE_ALL_CALLS+
        DBMS_TRACE.TRACE_ALL_EXCEPTIONS+
        DBMS_TRACE.TRACE_RESUME);

  -- obtain loop parameter values
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Interaction History (IH) Test Script');
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Author: Author's Name Here.');
  DBMS_OUTPUT.PUT_LINE('Version 1.0 - Initial Version - 06.27.2001');
  DBMS_OUTPUT.PUT_LINE(' ');
  xInteraction_Count := &xInteraction_Count;
  xparty_id := &xparty_id;
  xresource_id := &xresource_id;
  -- begin major loop
  begin_time_run := sysdate;
  DBMS_OUTPUT.PUT_LINE('Start Time := ' || TO_CHAR(begin_time_run, 'DD-MON-YYYY:HH:MI:SS'));


l_media.media_id := l_media_id;

    JTF_IH_PUB.Close_MediaItem
                (
                1.0,
                'T',
                'T',
                690,
                -1,
                2877,
                -1,
                l_return_status,
                l_msg_count,
                l_msg_data,
                l_media
                );
                IF l_return_status != 'S' THEN
                                  --Display all the error messages
                                FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                                        dbms_output.put_line(j);
                                       l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                                           p_encoded=>'F');
                                       DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_msg_data);
                                END LOOP;
                END IF;
DBMS_OUTPUT.PUT_LINE('PAST Close_MediaItem ');
```

```
DBMS_OUTPUT.PUT_LINE('Close_MediaItemv - l_return_status: '||l_return_status);
DBMS_SESSION.SET_SQL_TRACE(FALSE);
end;
/
```

### C.11.2.6 Open_Interaction

This script calls the Open_Interaction API and provides the following values using the Open_Interaction IN parameters:

- p_api_version: **1.0**
- p_init_msg_list: **T** indicates that this parameter is set to true
- p_commit: **T** indicates that this parameter is set to true
- p_resp_appl_id: the application identifier is **690**
- p_resp_id: the responsibility identifier is **-1**
- p_user_id: the user identifier is **2877**
- p_login_id: the login identifier is **-1**
- p_interaction_rec: the record type, interaction_rec_type contained in the JTF_IH_PUB and identified here as **l_interaction_rec**

```
set serveroutput on;

declare
 l_return_status VARCHAR2(30);
 l_msg_count NUMBER;
 l_msg_data VARCHAR2(200);

 l_interaction_rec APPS.JTF_IH_PUB.interaction_rec_type;
 l_interaction_id  NUMBER;
 l_jtf_note_id     NUMBER;
 m_count NUMBER := 0;
 m_active VARCHAR2(1);
 p_interaction_id NUMBER;

 l_activity_rec APPS.JTF_IH_PUB.activity_rec_type;
 l_activity_id_1  NUMBER;
 l_activity_id_2  NUMBER;
 m_activitycount NUMBER := 0;
 m_activityactive VARCHAR2(1);
 l_startdatecheck VARCHAR2(30);
 l_enddatecheck VARCHAR2(30);
```

```
 l_data                 VARCHAR2(8000);
 l_msg_index_out        NUMBER;
 xInteraction_Count     NUMBER := 2;
 cInteraction_Count     VARCHAR2(80);
 xparty_id              NUMBER := 1000;
 cparty_id              VARCHAR2(80);
 xresource_id           NUMBER := 10039;
 cresource_id           VARCHAR2(80);
 status                 NUMBER;
 end_time_runDATE;
 begin_time_runDATE;
 total_time_runNUMBER;
 nDbl               NUMBER := 1;
 l_activity_tbl     JTF_IH_PUB.activity_tbl_type;


 l_media APPS.JTF_IH_PUB.media_rec_type;
 l_media_lc_rec APPS.JTF_IH_PUB.media_lc_rec_type;
 l_media_id         NUMBER;
 l_mlcs APPS.JTF_IH_PUB.mlcs_tbl_type;
 l_milcs_id         NUMBER;

 l_activity_count    NUMBER;
 l_interaction_count    NUMBER;
 l_interaction_id_2      NUMBER;
begin
    DBMS_SESSION.SET_SQL_TRACE(TRUE);
    DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.TRACE_ALL_CALLS+
        DBMS_TRACE.TRACE_ALL_EXCEPTIONS+
        DBMS_TRACE.TRACE_RESUME);

  -- obtain loop parameter values
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Interaction History (IH) Test Script');
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Author: Author's Name Here.');
  DBMS_OUTPUT.PUT_LINE('Version 1.0 - Initial Version - 06.27.2001');
  DBMS_OUTPUT.PUT_LINE(' ');
  xInteraction_Count := &xInteraction_Count;
  xparty_id := &xparty_id;
  xresource_id := &xresource_id;
  -- begin major loop
  begin_time_run := sysdate;
  DBMS_OUTPUT.PUT_LINE('Start Time := ' || TO_CHAR(begin_time_run, 'DD-MON-YYYY:HH:MI:SS'));


l_interaction_rec.interaction_id := NULL;
        l_interaction_rec.reference_form := 'Test for JTF Open Interaction (Will close by Method 2)';
        l_interaction_rec.start_date_time := sysdate;
        l_interaction_rec.start_date_time := sysdate;
```

```
        l_interaction_rec.handler_id := 690;  -- Bell South: please validate for your environment
        l_interaction_rec.script_id := NULL;
        l_interaction_rec.outcome_id := 4;
        l_interaction_rec.result_id := 2;
        l_interaction_rec.reason_id := 2;
        l_interaction_rec.resource_id := xresource_id; -- jtfdom environment
        l_interaction_rec.party_id := xparty_id; -- JTFTECH environment
        l_interaction_rec.parent_id := NULL;

        JTF_IH_PUB.Open_Interaction(    1.0,
                                    'T',
                                    'T',
                                    690,
                                    -1,
                                    2877,
                                    -1,
                                    l_return_status,
                                    l_msg_count,
                                    l_msg_data,
                                    l_interaction_rec,
                                    l_interaction_id
                                    );
                                        IF l_return_status != 'S' THEN
                                         --Display all the error messages
                                            FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                                                    dbms_output.put_line(j);
                                                l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                                                            p_encoded=>'F');
                                                DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_msg_data);
                                            END LOOP;
                                         END IF;
        DBMS_OUTPUT.PUT_LINE('PAST Open_Interaction ');
        DBMS_OUTPUT.PUT_LINE('Open_Interaction - l_return_status: '||l_return_status);
        --
        --Interaction implicit notes bind
        --

            jtf_notes_pub.create_note(
         p_api_version => 1.0,
         p_source_object_id => l_interaction_id,
         p_source_object_code => 'JTF_INTERACTION',
         p_notes => 'Service Request Interaction - Note 1 - Customer claims that automatic water softener is
non-functional. Request full refund.',
         p_entered_by => 2877,
         p_entered_date => sysdate,
         p_last_update_date => sysdate,
         p_last_updated_by => 2877,
         p_creation_date => sysdate,
         x_jtf_note_id => l_jtf_note_id,
```

```
     x_msg_count => l_msg_count,
                 x_msg_data => l_msg_data,
     x_return_status => l_return_status);
                IF l_return_status != 'S' THEN
                --Display all the error messages
                       FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                           dbms_output.put_line(j);
                       l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                               p_encoded=>'F');
                        DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_msg_data);
                       END LOOP;
                END IF;
        DBMS_OUTPUT.PUT_LINE(' ');
        DBMS_OUTPUT.PUT('return_jtf_note_id: ' || l_jtf_note_id);
        IF (l_msg_count >= 1) THEN
        --Only one error
            FND_MSG_PUB.Get(p_msg_index => FND_MSG_PUB.G_FIRST,
                  p_encoded=>'F',
                  p_data=>l_data,
                 p_msg_index_out=>l_msg_index_out);
            DBMS_OUTPUT.PUT_LINE('Message(' || 1 ||'):= ' || l_data);
            IF (l_msg_count > 1) THEN
            --Display all the error messages
               FOR  j in  2..FND_MSG_PUB.Count_Msg LOOP
                   FND_MSG_PUB.Get(p_msg_index => FND_MSG_PUB.G_NEXT,
                           p_encoded=>'F',
                           p_data=>l_data,
                           p_msg_index_out=>l_msg_index_out);
                   DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' || l_data);
               END LOOP;
            END IF;
        END IF;
DBMS_OUTPUT.PUT_LINE(' ');


    l_interaction_rec.reference_form := 'Test for JTF Open Interaction';


    JTF_IH_PUB.Open_Interaction(    1.0,
                                'T',
                                'T',
                                690,
                                -1,
                                2877,
                                -1,
                                l_return_status,
                                l_msg_count,
                                l_msg_data,
                                l_interaction_rec,
                                l_interaction_id_2
```

```
                                );
                                 IF l_return_status != 'S' THEN
                                     --Display all the error messages
                                    FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                                             dbms_output.put_line(j);
                                            l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                                                p_encoded=>'F');
                                            DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_msg_data);
                                    END LOOP;
                                 END IF;
        DBMS_OUTPUT.PUT_LINE('PAST Open_Interaction ');
        DBMS_OUTPUT.PUT_LINE('PAST Open_Interaction - l_return_status: '||l_return_status);
```

### C.11.2.7 Update_Interaction

This script calls the Update_Interaction API and provides the following values using the Update_Interaction IN parameters:

- p_api_version: **1.0**
- p_init_msg_list: **T** indicates that this parameter is set to true
- p_commit: **T** indicates that this parameter is set to true
- p_resp_appl_id: the application identifier is **690**
- p_resp_id: the responsibility identifier is **-1**
- p_user_id: the user identifier is **2877**
- p_login_id: the login identifier is **-1**
- p_interaction_rec: the record type, interaction_rec_type contained in the JTF_IH_PUB and identified here as **l_interaction_rec**

```
set serveroutput on;

declare
 l_return_status VARCHAR2(30);
 l_msg_count NUMBER;
 l_msg_data VARCHAR2(200);

 l_interaction_rec APPS.JTF_IH_PUB.interaction_rec_type;
 l_interaction_id  NUMBER;
 l_jtf_note_id     NUMBER;
 m_count NUMBER := 0;
 m_active VARCHAR2(1);
 p_interaction_id NUMBER;
```

```
 l_activity_rec APPS.JTF_IH_PUB.activity_rec_type;
 l_activity_id_1  NUMBER;
 l_activity_id_2  NUMBER;
 m_activitycount NUMBER := 0;
 m_activityactive VARCHAR2(1);
 l_startdatecheck VARCHAR2(30);
 l_enddatecheck VARCHAR2(30);
 l_data                 VARCHAR2(8000);
 l_msg_index_out        NUMBER;
 xInteraction_Count     NUMBER := 2;
 cInteraction_Count     VARCHAR2(80);
 xparty_id              NUMBER := 1000;
 cparty_id              VARCHAR2(80);
 xresource_id           NUMBER := 10039;
 cresource_id           VARCHAR2(80);
 status                 NUMBER;
 end_time_runDATE;
 begin_time_runDATE;
 total_time_runNUMBER;
 nDbl               NUMBER := 1;
 l_activity_tbl     JTF_IH_PUB.activity_tbl_type;


 l_media APPS.JTF_IH_PUB.media_rec_type;
 l_media_lc_rec APPS.JTF_IH_PUB.media_lc_rec_type;
 l_media_id         NUMBER;
 l_mlcs APPS.JTF_IH_PUB.mlcs_tbl_type;
 l_milcs_id         NUMBER;

 l_activity_count    NUMBER;
 l_interaction_count    NUMBER;
 l_interaction_id_2      NUMBER;
begin
    DBMS_SESSION.SET_SQL_TRACE(TRUE);
    DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.TRACE_ALL_CALLS+
        DBMS_TRACE.TRACE_ALL_EXCEPTIONS+
        DBMS_TRACE.TRACE_RESUME);

  -- obtain loop parameter values
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Interaction History (IH) Test Script');
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Author: Author's Name Here.');
  DBMS_OUTPUT.PUT_LINE('Version 1.0 - Initial Version - 06.27.2001');
  DBMS_OUTPUT.PUT_LINE(' ');
  xInteraction_Count := &xInteraction_Count;
  xparty_id := &xparty_id;
  xresource_id := &xresource_id;
  -- begin major loop
```

```
begin_time_run := sysdate;
DBMS_OUTPUT.PUT_LINE('Start Time := ' || TO_CHAR(begin_time_run, 'DD-MON-YYYY:HH:MI:SS'));



l_interaction_rec.interaction_id := l_interaction_id;
       l_interaction_rec.reference_form := 'Test for Update Interaction';

       JTF_IH_PUB.Update_Interaction( 1.0,
                                 'T',
                                 'T',
                                 690,
                                 -1,
                                 2877,
                                 -1,
                                 l_return_status,
                                 l_msg_count,
                                 l_msg_data,
                                 l_interaction_rec
                                 );
           IF l_return_status != 'S' THEN
               --Display all the error messages
                  FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                          dbms_output.put_line(j);
                      l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                          p_encoded=>'F');
                      DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_msg_data);
                  END LOOP;
           END IF;
       DBMS_OUTPUT.PUT_LINE('PAST Update_Interaction ');
       DBMS_OUTPUT.PUT_LINE('Update_Interaction - l_return_status: '||l_return_status);

       --
```

### C.11.2.8 Add_Activity

This script calls the Add_Activity API and provides the following values using the Add_Activity IN parameters:

- p_api_version: **1.0**
- p_init_msg_list: **T** indicates that this parameter is set to true
- p_commit: **T** indicates that this parameter is set to true

- p_resp_appl_id: the application identifier is **690**
- p_resp_id: the responsibility identifier is -1
- p_user_id: the user identifier is 2877
- p_login_id: the login identifier is **-1**
- p_activity_rec: the record type, activity_rec_type contained in the JTF_IH_PUB and identified here as **l_activity_rec**

```
set serveroutput on;

declare
 l_return_status VARCHAR2(30);
 l_msg_count NUMBER;
 l_msg_data VARCHAR2(200);

 l_interaction_rec APPS.JTF_IH_PUB.interaction_rec_type;
 l_interaction_id  NUMBER;
 l_jtf_note_id     NUMBER;
 m_count NUMBER := 0;
 m_active VARCHAR2(1);
 p_interaction_id NUMBER;

 l_activity_rec APPS.JTF_IH_PUB.activity_rec_type;
 l_activity_id_1  NUMBER;
 l_activity_id_2  NUMBER;
 m_activitycount NUMBER := 0;
 m_activityactive VARCHAR2(1);
 l_startdatecheck VARCHAR2(30);
 l_enddatecheck VARCHAR2(30);
 l_data                 VARCHAR2(8000);
 l_msg_index_out        NUMBER;
 xInteraction_Count     NUMBER := 2;
 cInteraction_Count     VARCHAR2(80);
 xparty_id              NUMBER := 1000;
 cparty_id              VARCHAR2(80);
 xresource_id           NUMBER := 10039;
 cresource_id           VARCHAR2(80);
 status                 NUMBER;
 end_time_runDATE;
 begin_time_runDATE;
 total_time_runNUMBER;
 nDbl               NUMBER := 1;
 l_activity_tbl     JTF_IH_PUB.activity_tbl_type;


 l_media APPS.JTF_IH_PUB.media_rec_type;
```

```
 l_media_lc_rec APPS.JTF_IH_PUB.media_lc_rec_type;
 l_media_id         NUMBER;
 l_mlcs APPS.JTF_IH_PUB.mlcs_tbl_type;
 l_milcs_id         NUMBER;

 l_activity_count    NUMBER;
 l_interaction_count    NUMBER;
 l_interaction_id_2      NUMBER;
begin
    DBMS_SESSION.SET_SQL_TRACE(TRUE);
    DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.TRACE_ALL_CALLS+
        DBMS_TRACE.TRACE_ALL_EXCEPTIONS+
        DBMS_TRACE.TRACE_RESUME);

  -- obtain loop parameter values
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Interaction History (IH) Test Script');
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Author: Author's Name Here.');
  DBMS_OUTPUT.PUT_LINE('Version 1.0 - Initial Version - 06.27.2001');
  DBMS_OUTPUT.PUT_LINE(' ');
  xInteraction_Count := &xInteraction_Count;
  xparty_id := &xparty_id;
  xresource_id := &xresource_id;
  -- begin major loop
  begin_time_run := sysdate;
  DBMS_OUTPUT.PUT_LINE('Start Time := ' || TO_CHAR(begin_time_run, 'DD-MON-YYYY:HH:MI:SS'));



l_activity_rec.activity_id := NULL;
            l_activity_rec.duration := NULL;
            l_activity_rec.cust_account_id := NULL; --checked
            l_activity_rec.cust_org_id := null;
            l_activity_rec.role := 1;
            l_activity_rec.script_trans_id := fnd_api.g_miss_num;

            l_activity_rec.start_date_time := sysdate;
            --    l_activity_rec.start_date_time := to_date('29-SEP-2000 13:00:00', 'DD-MON-YYYY
HH24:MI:SS');
            l_activity_rec.end_date_time := NULL;
            --    l_activity_rec.end_date_time := to_date('26-JUL-2000 13:11:25', 'DD-MON-YYYY HH24:MI:SS');
            --    l_activity_rec.end_date_time := to_date('29-SEP-2000 13:11:25', 'DD-MON-YYYY HH24:MI:SS');

            --  l_activity_rec.task_id := 30;
            --  l_activity_rec.doc_id := 1;
            --  l_activity_rec.doc_ref := 1;
            l_activity_rec.media_id := NULL;
            l_activity_rec.action_item_id := 17;
```

```
    l_activity_rec.interaction_id := l_interaction_id;
    l_activity_rec.outcome_id := 7;
    l_activity_rec.result_id := 7;
    l_activity_rec.reason_id := 8;
    l_activity_rec.description := 'test Add Activity';
    l_activity_rec.action_id := 13;
    l_activity_rec.interaction_action_type := 'unknown';
    --  l_activity_rec.object_id := 1;
    --  l_activity_rec.object_type := 'JEZHU_Type_1';
    --  l_activity_rec.source_code_id := 10000;
    --  l_activity_rec.source_code := 'EEXHB10000';
    JTF_IH_PUB.Add_Activity(1.0,
                            'T',
                            'T',
                            -1,
                            690,
                            2877,
                            -1,
                            l_return_status,
                            l_msg_count,
                            l_msg_data,
                            l_activity_rec,
                            l_activity_id_1);
                            IF l_return_status != 'S' THEN
                                --Display all the error messages
                               FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                                        dbms_output.put_line(j);
                                       l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                                           p_encoded=>'F');
                                       DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_msg_data);
                               END LOOP;
                            END IF;

DBMS_OUTPUT.PUT_LINE('PAST Add_Activity ');
DBMS_OUTPUT.PUT_LINE('Add_Activity - l_return_status: '||l_return_status);

    IF (l_msg_count >= 1) THEN
    --Only one error
        FND_MSG_PUB.Get(p_msg_index => FND_MSG_PUB.G_FIRST,
                p_encoded=>'F',
                p_data=>l_data,
                p_msg_index_out=>l_msg_index_out);
        DBMS_OUTPUT.PUT_LINE('Message(' || 1 ||'):= ' || l_data);
        IF (l_msg_count > 1) THEN
            --Display all the error messages
           FOR j in  2..FND_MSG_PUB.Count_Msg LOOP
               FND_MSG_PUB.Get(p_msg_index => FND_MSG_PUB.G_NEXT,
                   p_encoded=>'F',
                   p_data=>l_data,
                   p_msg_index_out=>l_msg_index_out);
```

```
                DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' || l_data);
            END LOOP;
        END IF;
    END IF;
    DBMS_OUTPUT.PUT_LINE(' ');
    --
    --Activity implicit notes bind
    --

    jtf_notes_pub.create_note(
                    p_api_version => 1.0,
                    p_source_object_id => l_activity_id_1,
                    p_source_object_code => 'JTF_ACTIVITY',
                    p_notes => 'Service Request Activity 1 - Note 1 - Customer is angry. Rust color
water.',
                    p_entered_by => 2877,
                    p_entered_date => sysdate,
                    p_last_update_date => sysdate,
                    p_last_updated_by => 2877,
                    p_creation_date => sysdate,
                    x_jtf_note_id => l_jtf_note_id,
                    x_msg_count => l_msg_count,
                    x_msg_data => l_msg_data,
                    x_return_status => l_return_status);
        DBMS_OUTPUT.PUT_LINE(' ');
        DBMS_OUTPUT.PUT('return_jtf_note_id: ' || l_jtf_note_id);
        DBMS_OUTPUT.PUT_LINE(' ');
        DBMS_OUTPUT.PUT('Activity 1 Implicit Note return_status: ' || l_return_status);
        DBMS_OUTPUT.PUT_LINE(' ');
        DBMS_OUTPUT.PUT('msg_count: ' || l_msg_count);
        DBMS_OUTPUT.PUT_LINE(' ');
        IF (l_msg_count >= 1) THEN
            --Only one error
            FND_MSG_PUB.Get(p_msg_index => FND_MSG_PUB.G_FIRST,
                            p_encoded=>'F',
                            p_data=>l_data,
                            p_msg_index_out=>l_msg_index_out);
            DBMS_OUTPUT.PUT_LINE('Message(' || 1 ||'):= ' || l_data);
            IF (l_msg_count > 1) THEN
                --Display all the error messages
               FOR j in  2..FND_MSG_PUB.Count_Msg LOOP
                   FND_MSG_PUB.Get(p_msg_index => FND_MSG_PUB.G_NEXT,
                                   p_encoded=>'F',
                                   p_data=>l_data,
                                   p_msg_index_out=>l_msg_index_out);
                   DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' || l_data);
               END LOOP;
            END IF;
        END IF;
```

```
DBMS_OUTPUT.PUT_LINE(' ');
```

### C.11.2.9 Update_Activity

This script calls the Update_Activity API and provides the following values using the Update_Activity IN parameters:

- p_api_version: **1.0**
- p_init_msg_list: **T** indicates that this parameter is set to true
- p_commit: **T** indicates that this parameter is set to true
- p_resp_appl_id: the application identifier is **690**
- p_resp_id: the responsibility identifier is **-1**
- p_user_id: the user identifier is **2877**
- p_login_id: the login identifier is **-1**
- p_activity_rec: the record type, activity_rec_type contained in the JTF_IH_PUB and identified here as **l_activity_rec**. The activity_id field for this record type has been changed to **l_activity_id_1** and the description field has been changed to "**test update activity**".

```
set serveroutput on;

declare
 l_return_status VARCHAR2(30);
 l_msg_count NUMBER;
 l_msg_data VARCHAR2(200);

 l_interaction_rec APPS.JTF_IH_PUB.interaction_rec_type;
 l_interaction_id  NUMBER;
 l_jtf_note_id     NUMBER;
 m_count NUMBER := 0;
 m_active VARCHAR2(1);
 p_interaction_id NUMBER;

 l_activity_rec APPS.JTF_IH_PUB.activity_rec_type;
 l_activity_id_1  NUMBER;
 l_activity_id_2  NUMBER;
 m_activitycount NUMBER := 0;
 m_activityactive VARCHAR2(1);
 l_startdatecheck VARCHAR2(30);
 l_enddatecheck VARCHAR2(30);
 l_data                VARCHAR2(8000);
```

```
 l_msg_index_out        NUMBER;
 xInteraction_Count     NUMBER := 2;
 cInteraction_Count     VARCHAR2(80);
 xparty_id              NUMBER := 1000;
 cparty_id              VARCHAR2(80);
 xresource_id           NUMBER := 10039;
 cresource_id           VARCHAR2(80);
 status                 NUMBER;
 end_time_runDATE;
 begin_time_runDATE;
 total_time_runNUMBER;
 nDbl               NUMBER := 1;
 l_activity_tbl     JTF_IH_PUB.activity_tbl_type;


 l_media APPS.JTF_IH_PUB.media_rec_type;
 l_media_lc_rec APPS.JTF_IH_PUB.media_lc_rec_type;
 l_media_id         NUMBER;
 l_mlcs APPS.JTF_IH_PUB.mlcs_tbl_type;
 l_milcs_id         NUMBER;

 l_activity_count    NUMBER;
 l_interaction_count    NUMBER;
 l_interaction_id_2      NUMBER;
begin
    DBMS_SESSION.SET_SQL_TRACE(TRUE);
    DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.TRACE_ALL_CALLS+
        DBMS_TRACE.TRACE_ALL_EXCEPTIONS+
        DBMS_TRACE.TRACE_RESUME);

  -- obtain loop parameter values
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Interaction History (IH) Test Script');
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Author: Author's Name Here.');
  DBMS_OUTPUT.PUT_LINE('Version 1.0 - Initial Version - 06.27.2001');
  DBMS_OUTPUT.PUT_LINE(' ');
  xInteraction_Count := &xInteraction_Count;
  xparty_id := &xparty_id;
  xresource_id := &xresource_id;
  -- begin major loop
  begin_time_run := sysdate;
  DBMS_OUTPUT.PUT_LINE('Start Time := ' || TO_CHAR(begin_time_run, 'DD-MON-YYYY:HH:MI:SS'));



l_activity_rec.activity_id := l_activity_id_1;
            l_activity_rec.description := 'Test Update Activity';
```

```
            JTF_IH_PUB.Update_Activity(1.0,
                                       'T',
                                       'T',
                                       -1,
                                       690,
                                       2877,
                                       -1,
                                       l_return_status,
                                       l_msg_count,
                                       l_msg_data,
                                       l_activity_rec);
                                       IF l_return_status != 'S' THEN
                                           --Display all the error messages
                                          FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                                                   dbms_output.put_line(j);
                                                  l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                                                      p_encoded=>'F');
                                                  DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_
msg_data);
                                          END LOOP;
                                       END IF;
```

### C.11.2.10 Update_ActivityDuration

This script calls the Update_ActivityDuration API and provides the following values using the Update_ActivityDuration IN parameters:

- p_api_version: **1.0**
- p_init_msg_list: **T** indicates that this parameter is set to true
- p_commit: **T** indicates that this parameter is set to true
- p_resp_appl_id: the application identifier is **690**
- p_resp_id: the responsibility identifier is **-1**
- p_user_id: the user identifier is **2877**
- p_login_id: the login identifier is **-1**
- p_activity_id: the activity identifier is **l_activity_id_1**
- p_end_date_time: the end date and time is determined using the **SYSDATE** SQL command
- p_duration: the activity duration is **1** second

```
set serveroutput on;

declare
 l_return_status VARCHAR2(30);
 l_msg_count NUMBER;
 l_msg_data VARCHAR2(200);

 l_interaction_rec APPS.JTF_IH_PUB.interaction_rec_type;
 l_interaction_id  NUMBER;
 l_jtf_note_id     NUMBER;
 m_count NUMBER := 0;
 m_active VARCHAR2(1);
 p_interaction_id NUMBER;

 l_activity_rec APPS.JTF_IH_PUB.activity_rec_type;
 l_activity_id_1  NUMBER;
 l_activity_id_2  NUMBER;
 m_activitycount NUMBER := 0;
 m_activityactive VARCHAR2(1);
 l_startdatecheck VARCHAR2(30);
 l_enddatecheck VARCHAR2(30);
 l_data                 VARCHAR2(8000);
 l_msg_index_out        NUMBER;
 xInteraction_Count     NUMBER := 2;
 cInteraction_Count     VARCHAR2(80);
 xparty_id              NUMBER := 1000;
 cparty_id              VARCHAR2(80);
 xresource_id           NUMBER := 10039;
 cresource_id           VARCHAR2(80);
 status                 NUMBER;
 end_time_runDATE;
 begin_time_runDATE;
 total_time_runNUMBER;
 nDbl               NUMBER := 1;
 l_activity_tbl     JTF_IH_PUB.activity_tbl_type;


 l_media APPS.JTF_IH_PUB.media_rec_type;
 l_media_lc_rec APPS.JTF_IH_PUB.media_lc_rec_type;
 l_media_id         NUMBER;
 l_mlcs APPS.JTF_IH_PUB.mlcs_tbl_type;
 l_milcs_id         NUMBER;

 l_activity_count    NUMBER;
 l_interaction_count    NUMBER;
 l_interaction_id_2      NUMBER;
begin
    DBMS_SESSION.SET_SQL_TRACE(TRUE);
```

```
   DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.TRACE_ALL_CALLS+
       DBMS_TRACE.TRACE_ALL_EXCEPTIONS+
       DBMS_TRACE.TRACE_RESUME);

  -- obtain loop parameter values
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Interaction History (IH) Test Script');
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Author: Author's Name Here.');
  DBMS_OUTPUT.PUT_LINE('Version 1.0 - Initial Version - 06.27.2001');
  DBMS_OUTPUT.PUT_LINE(' ');
  xInteraction_Count := &xInteraction_Count;
  xparty_id := &xparty_id;
  xresource_id := &xresource_id;
  -- begin major loop
  begin_time_run := sysdate;
  DBMS_OUTPUT.PUT_LINE('Start Time := ' || TO_CHAR(begin_time_run, 'DD-MON-YYYY:HH:MI:SS'));


 JTF_IH_PUB.Update_ActivityDuration
                                    ( 1.0,
                                    'T',
                                    'T',
                                    690,
                                    -1,
                                    2877,
                                    -1,
                                    l_return_status,
                                    l_msg_count,
                                    l_msg_data,
                                    l_activity_id_1,
                                    sysdate,
                                    1
                                    );
                                    IF l_return_status != 'S' THEN
                                        --Display all the error messages
                                       FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                                                dbms_output.put_line(j);
                                               l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                                                   p_encoded=>'F');
                                               DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_
msg_data);
                                       END LOOP;
                                    END IF;

         DBMS_OUTPUT.PUT_LINE('PAST Update_ActivityDuration ');
```

```
DBMS_OUTPUT.PUT_LINE('Update_ActivityDuration - l_return_status: '||l_return_status);
```

### C.11.2.11 Close_Interaction

This script contains two different methods for calling the Close_Interaction API and for providing the following values using the Close_Interaction IN parameters:

- p_api_version: **1.0**
- p_init_msg_list: **T** indicates that this parameter is set to true
- p_commit: **T** indicates that this parameter is set to true
- p_resp_appl_id: the application identifier is **690**
- p_resp_id: the responsibility identifier is **-1**
- p_user_id: the user identifier is **2877**
- p_login_id: the login identifier is **-1**
- p_interaction_rec: the record type, interaction_rec_type contained in the JTF_IH_PUB and identified as **l_interaction_rec** for method 1 and **l_interaction_id_2** for method 2

```
set serveroutput on;

declare
 l_return_status VARCHAR2(30);
 l_msg_count NUMBER;
 l_msg_data VARCHAR2(200);

 l_interaction_rec APPS.JTF_IH_PUB.interaction_rec_type;
 l_interaction_id  NUMBER;
 l_jtf_note_id     NUMBER;
 m_count NUMBER := 0;
 m_active VARCHAR2(1);
 p_interaction_id NUMBER;

 l_activity_rec APPS.JTF_IH_PUB.activity_rec_type;
 l_activity_id_1  NUMBER;
 l_activity_id_2  NUMBER;
 m_activitycount NUMBER := 0;
 m_activityactive VARCHAR2(1);
 l_startdatecheck VARCHAR2(30);
 l_enddatecheck VARCHAR2(30);
```

```
 l_data                 VARCHAR2(8000);
 l_msg_index_out        NUMBER;
 xInteraction_Count     NUMBER := 2;
 cInteraction_Count     VARCHAR2(80);
 xparty_id              NUMBER := 1000;
 cparty_id              VARCHAR2(80);
 xresource_id           NUMBER := 10039;
 cresource_id           VARCHAR2(80);
 status                 NUMBER;
 end_time_runDATE;
 begin_time_runDATE;
 total_time_runNUMBER;
 nDbl               NUMBER := 1;
 l_activity_tbl     JTF_IH_PUB.activity_tbl_type;


 l_media APPS.JTF_IH_PUB.media_rec_type;
 l_media_lc_rec APPS.JTF_IH_PUB.media_lc_rec_type;
 l_media_id         NUMBER;
 l_mlcs APPS.JTF_IH_PUB.mlcs_tbl_type;
 l_milcs_id         NUMBER;

 l_activity_count    NUMBER;
 l_interaction_count    NUMBER;
 l_interaction_id_2      NUMBER;
begin
    DBMS_SESSION.SET_SQL_TRACE(TRUE);
    DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.TRACE_ALL_CALLS+
        DBMS_TRACE.TRACE_ALL_EXCEPTIONS+
        DBMS_TRACE.TRACE_RESUME);

  -- obtain loop parameter values
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Interaction History (IH) Test Script');
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Author: Author's Name Here.');
  DBMS_OUTPUT.PUT_LINE('Version 1.0 - Initial Version - 06.27.2001');
  DBMS_OUTPUT.PUT_LINE(' ');
  xInteraction_Count := &xInteraction_Count;
  xparty_id := &xparty_id;
  xresource_id := &xresource_id;
  -- begin major loop
  begin_time_run := sysdate;
  DBMS_OUTPUT.PUT_LINE('Start Time := ' || TO_CHAR(begin_time_run, 'DD-MON-YYYY:HH:MI:SS'));


--
--      Method - I
--
```

```
    l_interaction_rec.interaction_id := l_interaction_id;
        JTF_IH_PUB.Close_Interaction(   1.0,
                                        'T',
                                        'T',
                                        690,
                                        -1,
                                        2877,
                                        null,
                                        l_return_status,
                                        l_msg_count,
                                        l_msg_data,
                                        l_interaction_rec);
                                    IF l_return_status != 'S' THEN
                                        --Display all the error messages
                                       FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                                                dbms_output.put_line(j);
                                               l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                                                   p_encoded=>'F');
                                               DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_
msg_data);
                                       END LOOP;
                                    END IF;

        DBMS_OUTPUT.PUT_LINE('PAST Close_Interaction  Method 1 ');
        DBMS_OUTPUT.PUT_LINE('Close_Interaction - l_return_status: '||l_return_status);
--
--      Method - II
--

        JTF_IH_PUB.Close_Interaction(   1.0,
                                        'T',
                                        'T',
                                        690,
                                        -1,
                                        2877,
                                        null,
                                        l_return_status,
                                        l_msg_count,
                                        l_msg_data,
                                        l_interaction_id_2
                                        );
                                    IF l_return_status != 'S' THEN
                                        --Display all the error messages
                                       FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                                                dbms_output.put_line(j);
```

```
                                                l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                                                    p_encoded=>'F');
                                                DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_
msg_data);
                                            END LOOP;
                                        END IF;

        DBMS_OUTPUT.PUT_LINE('PAST Close_Interaction  Method 2');
        DBMS_OUTPUT.PUT_LINE('Close_Interaction - l_return_status: '||l_return_status);
--
-- Close_MediaItem
--
l_media.media_id := l_media_id;

    JTF_IH_PUB.Close_MediaItem
                (
                1.0,
                'T',
                'T',
                690,
                -1,
                2877,
                -1,
                l_return_status,
                l_msg_count,
                l_msg_data,
                l_media
                );
                IF l_return_status != 'S' THEN
                                         --Display all the error messages
                                        FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                                                 dbms_output.put_line(j);
                                                l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                                                    p_encoded=>'F');
                                                DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_
msg_data);
                                        END LOOP;
                END IF;
DBMS_OUTPUT.PUT_LINE('PAST Close_MediaItem ');
DBMS_OUTPUT.PUT_LINE('Close_MediaItemv - l_return_status: '||l_return_status);
DBMS_SESSION.SET_SQL_TRACE(FALSE);
end;
/
```

## C.11.3 Counting APIs

The SQL scripts in this section pass filtering parameters to the counting APIs to return the count on an interaction or an activity.

### C.11.3.1 Get_InteractionActivityCount

This script calls the Get_InteractionActivityCount API and provides the following filtering parameters using the Get_InteractionActivityCount IN parameters:

- p_api_version: **1.0**
- p_init_msg_list: **T** indicates that this parameter is set to true
- p_resp_appl_id: the application identifier is **690**
- p_resp_id: the responsibility identifier is **-1**
- p_user_id: the user identifier is **2877**
- p_login_id: the login identifier is **-1**
- p_outcome_id: the activity's outcome identifier is **l_activity_rec.outcome_id**
- p_result_id: the activity's result identifier is **l_activity_rec.result_id**
- p_reason_id: the activity's reason identifier is **l_activity_rec.reason_id**
- p_script_id: the interaction's script identifier is **l_interaction_rec.script_id**
- p_media_id: the activity's media identifier is **l_activity_rec.media_id**

```
set serveroutput on;

declare
 l_return_status VARCHAR2(30);
 l_msg_count NUMBER;
 l_msg_data VARCHAR2(200);

 l_interaction_rec APPS.JTF_IH_PUB.interaction_rec_type;
 l_interaction_id  NUMBER;
 l_jtf_note_id     NUMBER;
 m_count NUMBER := 0;
 m_active VARCHAR2(1);
 p_interaction_id NUMBER;

 l_activity_rec APPS.JTF_IH_PUB.activity_rec_type;
 l_activity_id_1  NUMBER;
 l_activity_id_2  NUMBER;
```

```
 m_activitycount NUMBER := 0;
 m_activityactive VARCHAR2(1);
 l_startdatecheck VARCHAR2(30);
 l_enddatecheck VARCHAR2(30);
 l_data                 VARCHAR2(8000);
 l_msg_index_out        NUMBER;
 xInteraction_Count     NUMBER := 2;
 cInteraction_Count     VARCHAR2(80);
 xparty_id              NUMBER := 1000;
 cparty_id              VARCHAR2(80);
 xresource_id           NUMBER := 10039;
 cresource_id           VARCHAR2(80);
 status                 NUMBER;
 end_time_runDATE;
 begin_time_runDATE;
 total_time_runNUMBER;
 nDbl               NUMBER := 1;
 l_activity_tbl     JTF_IH_PUB.activity_tbl_type;


 l_media APPS.JTF_IH_PUB.media_rec_type;
 l_media_lc_rec APPS.JTF_IH_PUB.media_lc_rec_type;
 l_media_id         NUMBER;
 l_mlcs APPS.JTF_IH_PUB.mlcs_tbl_type;
 l_milcs_id         NUMBER;

 l_activity_count    NUMBER;
 l_interaction_count    NUMBER;
 l_interaction_id_2      NUMBER;
begin
    DBMS_SESSION.SET_SQL_TRACE(TRUE);
    DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.TRACE_ALL_CALLS+
        DBMS_TRACE.TRACE_ALL_EXCEPTIONS+
        DBMS_TRACE.TRACE_RESUME);

  -- obtain loop parameter values
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Interaction History (IH) Test Script');
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Author: Author's Name Here.');
  DBMS_OUTPUT.PUT_LINE('Version 1.0 - Initial Version - 06.27.2001');
  DBMS_OUTPUT.PUT_LINE(' ');
  xInteraction_Count := &xInteraction_Count;
  xparty_id := &xparty_id;
  xresource_id := &xresource_id;
  -- begin major loop
  begin_time_run := sysdate;
  DBMS_OUTPUT.PUT_LINE('Start Time := ' || TO_CHAR(begin_time_run, 'DD-MON-YYYY:HH:MI:SS'));
```

```
JTF_IH_PUB.Get_InteractionActivityCount
                                      (  1.0,
                                     'T',
                                     690,
                                     -1,
                                     2877,
                                     -1,
                                     l_return_status,
                                     l_msg_count,
                                     l_msg_data,
                                     l_activity_rec.outcome_id,
                                     l_activity_rec.result_id,
                                     l_activity_rec.reason_id,
                                     l_interaction_rec.script_id,
                                     l_activity_rec.media_id,
                                     l_activity_count
                                         );
                                  IF l_return_status != 'S' THEN
                                      --Display all the error messages
                                     FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                                              dbms_output.put_line(j);
                                             l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                                                 p_encoded=>'F');
                                             DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_msg_data);
                                     END LOOP;
                                  END IF;

        DBMS_OUTPUT.PUT_LINE('PAST Get_InteractionActivityCount ');
        DBMS_OUTPUT.PUT_LINE('Get_InteractionActivityCount - l_return_status: '||l_return_status);
        DBMS_OUTPUT.PUT_LINE('l_activity_count - '|| to_char(l_activity_count));
```

### C.11.3.2 Get_InteractionCount

This script calls the Get_InteractionCount API and provides the following input parameters using the Get_InteractionCount IN parameters, to derive an interaction count:

- p_api_version: **1.0**
- p_init_msg_list: **T** indicates that this parameter is set to true
- p_resp_appl_id: the application identifier is **690**
- p_resp_id: the responsibility identifier is **-1**
- p_user_id: the user identifier is **2877**
- p_login_id: the login identifier is **-1**

- p_outcome_id: the interaction's outcome identifier is **l_interaction_rec.outcome_id**
- p_result_id: the interaction's result identifier is **l_interaction_rec.result_id**
- p_reason_id: the interaction's reason identifier is **l_interaction_rec.reason_id**

```
set serveroutput on;

declare
 l_return_status VARCHAR2(30);
 l_msg_count NUMBER;
 l_msg_data VARCHAR2(200);

 l_interaction_rec APPS.JTF_IH_PUB.interaction_rec_type;
 l_interaction_id  NUMBER;
 l_jtf_note_id     NUMBER;
 m_count NUMBER := 0;
 m_active VARCHAR2(1);
 p_interaction_id NUMBER;

 l_activity_rec APPS.JTF_IH_PUB.activity_rec_type;
 l_activity_id_1  NUMBER;
 l_activity_id_2  NUMBER;
 m_activitycount NUMBER := 0;
 m_activityactive VARCHAR2(1);
 l_startdatecheck VARCHAR2(30);
 l_enddatecheck VARCHAR2(30);
 l_data                 VARCHAR2(8000);
 l_msg_index_out        NUMBER;
 xInteraction_Count     NUMBER := 2;
 cInteraction_Count     VARCHAR2(80);
 xparty_id              NUMBER := 1000;
 cparty_id              VARCHAR2(80);
 xresource_id           NUMBER := 10039;
 cresource_id           VARCHAR2(80);
 status                 NUMBER;
 end_time_runDATE;
 begin_time_runDATE;
 total_time_runNUMBER;
 nDbl               NUMBER := 1;
 l_activity_tbl     JTF_IH_PUB.activity_tbl_type;


 l_media APPS.JTF_IH_PUB.media_rec_type;
 l_media_lc_rec APPS.JTF_IH_PUB.media_lc_rec_type;
 l_media_id         NUMBER;
 l_mlcs APPS.JTF_IH_PUB.mlcs_tbl_type;
 l_milcs_id         NUMBER;
```

```
 l_activity_count    NUMBER;
 l_interaction_count    NUMBER;
 l_interaction_id_2      NUMBER;
begin
    DBMS_SESSION.SET_SQL_TRACE(TRUE);
    DBMS_TRACE.SET_PLSQL_TRACE(DBMS_TRACE.TRACE_ALL_CALLS+
        DBMS_TRACE.TRACE_ALL_EXCEPTIONS+
        DBMS_TRACE.TRACE_RESUME);

  -- obtain loop parameter values
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Interaction History (IH) Test Script');
  DBMS_OUTPUT.PUT_LINE(' ');
  DBMS_OUTPUT.PUT_LINE('Author: Author's Name Here.');
  DBMS_OUTPUT.PUT_LINE('Version 1.0 - Initial Version - 06.27.2001');
  DBMS_OUTPUT.PUT_LINE(' ');
  xInteraction_Count := &xInteraction_Count;
  xparty_id := &xparty_id;
  xresource_id := &xresource_id;
  -- begin major loop
  begin_time_run := sysdate;
  DBMS_OUTPUT.PUT_LINE('Start Time := ' || TO_CHAR(begin_time_run, 'DD-MON-YYYY:HH:MI:SS'));


  JTF_IH_PUB.Get_InteractionCount(   1.0,
                                         'T',
                                         690,
                                         -1,
                                         2877,
                                         -1,
                                         l_return_status,
                                         l_msg_count,
                                         l_msg_data,
                                         l_interaction_rec.outcome_id,
                                         l_interaction_rec.result_id,
                                         l_interaction_rec.reason_id,
                                         x_interaction_count => l_interaction_count);
                                   IF l_return_status != 'S' THEN
                                       --Display all the error messages
                                      FOR j in  1..FND_MSG_PUB.Count_Msg LOOP
                                               dbms_output.put_line(j);
                                              l_msg_data := FND_MSG_PUB.Get(p_msg_index => j,
                                                  p_encoded=>'F');
                                              DBMS_OUTPUT.PUT_LINE('Message(' || j ||'):= ' ||l_msg_data);
                                      END LOOP;
                                   END IF;

        DBMS_OUTPUT.PUT_LINE('PAST Get_InteractionCount ');
```

```
DBMS_OUTPUT.PUT_LINE('Get_InteractionCount - l_return_status: '||l_return_status);
DBMS_OUTPUT.PUT_LINE('l_interaction_count - '|| to_char(l_interaction_count));
```

# D

# Data Validations

The following sections describe the data validations done when populating Interaction History either via the Public API or via the Import concurrent program.

Topics include:

- Section D.1, "Interaction Validations and Defaults"
- Section D.2, "Activity Validations and Defaults"
- Section D.3, "Media Item Validations and Defaults"
- Section D.4, "Media Item Life-cycle Segment Validations and Defaults"

## D.1 Interaction Validations and Defaults

The following table describes the validations and defaults for all of the Interaction Record Type columns that are performed when calling the IH API to Insert or update an Interaction. This includes the following API Calls:

- Create_Interaction
- Open_Interaction
- Update_Interaction
- Close_Interaction

The JTF_IH_INTERACTIONS columns not exposed via the API are documented at the end of the table for reference purposes.

These validations apply when importing interactions from the JTF_IH_ INTERACTIONS_STG table via the Import.

| Interaction Record Value | Table column mapping in JTF_IH_ INTERACTIONS and JTF_IH_ INTERACTIONS_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| interaction_id | INTERACTION_ID | Unique interaction identifier | Y | | Sequence Generated | If passed, the value passed will be used. Value passed should be generated from sequence: jtf_ih_interactions_s1. If not passed, an ID is generated in Open_ Interaction and Create_ Interaction calls. |
| party_id | PARTY_ID | ID of customer Person, Relationship or Organization with whom the interaction was done. FK to HZ_ PARTIES | Y | | | Valid PARTY_ID in HZ_ PARTIES. |
| Primary_party_ id | PRIMARY_PARTY_ID | ID of customer (Person or Organization) with whom the interaction was done. FK to HZ_ PARTIES | N | | | Valid PARTY_ID in HZ_ PARTIES. This value cannot be a RELATIONSHIP party type. This value is not required using the pre-11.5.10 single party ID calling convention. |
| Contact_rel_ party_id | CONTACT_REL_ PARTY_ID | ID of the relationship party that defines the contacts relationship to the customer. FK to HZ_PARTIES | Y | | | Valid PARTY_ID in HZ_ PARTIES. This value must be a RELATIONSHIP party type. This value is not required using the pre-11.5.10 single party ID calling convention. |
| Contact_ party_ id | CONTACT_PARTY_ID | ID of the person party that is the contacts of the customer. FK to HZ_PARTIES | N | | | Valid PARTY_ID in HZ_ PARTIES. This value must be a PERSON party type. Requires that the Contact_Rel_party_id be passed with a RELATIOSHIP ID that the contact is a part of. This value is not required using the pre-11.5.10 single party ID calling convention. |

| Interaction Record Value | Table column mapping in JTF_IH_ INTERACTIONS and JTF_IH_ INTERACTIONS_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| resource_id | RESOURCE_ID | ID of the agent/user of the person who perform the interaction with the customer. FK to JTF_RS_ RESOURCE_ EXTN. | N | | | Valid RESOURCE_ID in JTF_RS_RESOURCE_ EXTN |
| handler_id | HANDLER_ID | The application id of the application that logged the interaction. FK to FND_ APPLICATION. | Y | | | Valid APPLICATION_ID in FND_APPLICATION |
| outcome_id | OUTCOME_ID | ID of outcome code assigned to the interaction. FK to JTF_IH_ OUTCOMES_B. | Y | | | Valid OUTCOME_ID in JTF_IH_OUTCOMES_B |
| result_id | RESULT_ID | ID of the result code assigned to the interaction. FK to JTF_IH_ RESULTS_B. | | | | Valid RESULT_ID in JTF_ IH_RESULTS_B |
| reason_id | REASON_ID | ID of the reason code assigned to the interaction. FK to JTF_IH_ REASONS_B. | | | | Valid REASON_ID in JTF_IH_REASONS_B |
| source_code_id | SOURCE_CODE_ID | The Marketing Campaign source code identifier. FK to AMS_ SOURCE_ CODES | | | | Valid SOURCE_CODE_ ID in AMS_SOURCE_ CODES |
| source_code | SOURCE_CODE | The Marketing source code. FK to AMS_ SOURCE_ CODES | | | | Valid SOURCE_CODE in AMS_SOURCE_CODES |

| Interaction Record Value | Table column mapping in JTF_IH_ INTERACTIONS and JTF_IH_ INTERACTIONS_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| object_type | OBJECT_TYPE | The type of marketing source code. Campaign, Event, Campaign Schedule, Etc. | | | | Valid ARC_SOURCE_ CODE_FOR in AMS_ SOURCE_CODES. Marketing SOURCE_ TYPE. |
| object_id | OBJECT_ID | The ID of the Campaign, Event, Etc. as qualified by OBJECT_TYPE. | | | | Valid SOURCE_CODE_ FOR_ID in AMS_ SOURCE_CODES. Marketing OBJECT_ID |
| parent_id | JTF_IH_ INTERACTION_ INTERS.INTERACT_ INTERACTION_ IDRELATES | INTERACTION_ ID of the related parent interaction. | | | | Valid INTERACTION_ID in JTF_IH_ INTERACTIONS. |
| start_date_time | START_DATE_TIME | The date and time the interaction started, down to the second. | | | SYSDATE | Set to SYSDATE via Open_Interaction or Create_Interaction calls. If a value is passed it is used in place of SYSDATE. Must be less then or equal to END_ DATE_TIME. |
| end_date_time | END_DATE_TIME | The date and time the interaction ended, down to the second. | | | SYSDATE | Set to SYSDATE via Close_Interaction or Create_Interaction calls. If a value is passed it is used in place of SYSDATE. Must be greater then or equal to START_DATE_TIME. |
| Duration | DURATION | Number of seconds that the interaction was active. | | | Calculated | If passed, the value passed will be used, if not it is calculated as: END_ DATE_TIME – START_ DATE_TIME. |
| inter_ interaction_ duration | INTER_INTERACTION_ DURATION | The amount of time, in seconds, that the agent spent between interactions. | | | | |

| Interaction Record Value | Table column mapping in JTF_IH_ INTERACTIONS and JTF_IH_ INTERACTIONS_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| non_ productive_ time_amount | NON_PRODUCTIVE_ TIME_AMOUNT | Currently Not used. The amount of time the agent spent in seconds on non-customer related activities. | | | | |
| Preview_time_ amount | PREVIEW_TIME_ AMOUNT | The amount of time, in seconds, that the agent spent reviewing the customer information before interacting with the customer. | | | | |
| productive_ time_amount | PRODUCTIVE_TIME_ AMOUNT | The amount of time in seconds that the agent spent doing productive work. | | | | |
| wrapUp_time_ amount | WRAP_UP_TIME_ AMOUNT | The amount of time the agent spent in seconds after the phone call was terminated to status and close the interaction. | | | | |
| script_id | SCRIPT_ID | Not used. | | Y | | A script/survey is related via the JTF_IH_ ACTIVITIES.SCRIPT_ TRANS_ID. |
| attribute_ category | ATTRIBUTE_ CATEGORY | Descriptive Flexfield Structured definition column | | | | |
| attribute1 | ATTRIBUTE1 | Descriptive Flexfield Segment | | | | |
| attribute2 | ATTRIBUTE2 | Descriptive Flexfield Segment | | | | |

| Interaction Record Value | Table column mapping in JTF_IH_ INTERACTIONS and JTF_IH_ INTERACTIONS_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| attribute3 | ATTRIBUTE3 | Descriptive Flexfield Segment | | | | |
| attribute4 | ATTRIBUTE4 | Descriptive Flexfield Segment | | | | |
| attribute5 | ATTRIBUTE5 | Descriptive Flexfield Segment | | | | |
| attribute6 | ATTRIBUTE6 | Descriptive Flexfield Segment | | | | |
| attribute7 | ATTRIBUTE7 | Descriptive Flexfield Segment | | | | |
| attribute8 | ATTRIBUTE8 | Descriptive Flexfield Segment | | | | |
| attribute9 | ATTRIBUTE9 | Descriptive Flexfield Segment | | | | |
| attribute10 | ATTRIBUTE10 | Descriptive Flexfield Segment | | | | |
| attribute11 | ATTRIBUTE11 | Descriptive Flexfield Segment | | | | |
| attribute12 | ATTRIBUTE12 | Descriptive Flexfield Segment | | | | |
| attribute13 | ATTRIBUTE13 | Descriptive Flexfield Segment | | | | |
| attribute14 | ATTRIBUTE14 | Descriptive Flexfield Segment | | | | |
| attribute15 | ATTRIBUTE15 | Descriptive Flexfield Segment | | | | |

| Interaction Record Value | Table column mapping in JTF_IH_ INTERACTIONS and JTF_IH_ INTERACTIONS_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| touchpoint1_ type | TOUCHPOINT1_TYPE | Type indicator to specify the type of ID stored in the RESOURCE_ ID field. If it is "PARTY" (default), then the ID is considered to be a PARTY_ID, else it is treated as a RESOURCE_ID | Y | Y | 'PARTY' | Should not be changed. Use default values only. |
| touchpoint2_ type | TOUCHPOINT2_TYPE | Type indicator to specify the type of ID stored in the RESOURCE_ ID field. If it is "RS_ EMPLOYEE" (default), then the ID is considered to be a RESOURCE_ ID, else it is treated as a PARTY_ID. | Y | Y | 'RS_ EMPLOYE E' | Should not be changed. Use default values only. |
| method_code | METHOD_CODE | Legacy from CS 3i interactions manager schema. Describes the type of media that was used to perform the interaction. Call, E-mail, etc. | | Y | | Added for Service Migration from 3i schema, however it is not used in the migration scripts. |
| reference_form | REFERENCE_FORM | Legacy from CS 3i interactions manager schema. | | Y | | Added for Service Migration from 3i schema from CS_ INTERACTIONS.REFERE NCE_FORM |
| follow_up_ action | FOLLOW_UP_ACTION | Legacy from CS 3i interactions manager schema. A free-form text note. | | Y | | Added for Service Migration from 3i schema from CS_ INTERACTIONS. FOLLOW_UP_ACTION |

| Interaction Record Value | Table column mapping in JTF_IH_ INTERACTIONS and JTF_IH_ INTERACTIONS_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| | ACTIVE | Y/N indicator. Y if the interaction is open, N if it is closed. | Y | | Y/N | Set to 'Y' in Open_ Interaction and to 'N' in Close_Interaction calls. If Create_Interaction is used, the interaction is created as non-active, N. |
| | OBJECT_VERSION_ NUMBER | Standard Object Version Field | Y | | | Set to 1.0 |
| | INTERACTION_ INTERS_ID | Replaced by JTF_ IH_ INTERACTION_ INTERS table INTERACT_ INTERACTION_ IDRELATES populated by parent_id. (See above) | | Y | | |
| P_user_id | CREATED_BY | Standard who column - user who created this row (foreign key to FND_ USER.USER_ID) | | | | Populated using valid user id from standard API parameters |
| | CREATION_DATE | Standard who column - date when this row was created. | | | SYSDATE | |
| P_user_id | LAST_UPDATED_BY | Standard who column - user who last updated this row (foreign key to FND_ USER.USER_ID). | | | | Populated using valid user id from standard API parameters |
| | LAST_UPDATE_DATE | Standard Who column - date when a user last updated this row. | | | SYSDATE | |

| Interaction Record Value | Table column mapping in JTF_IH_ INTERACTIONS and JTF_IH_ INTERACTIONS_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| P_login_id | LAST_UPDATE_LOGIN | Standard who column - operating system login of user who last updated this row (foreign key to FND_ LOGINS.LOGIN _ID). | | | | Populated using valid login id from standard API parameters |
| bulk_writer_ code | bulk_writer_code | Internal Use Only | | | | |
| bulk_batch_type | bulk_batch_type | Internal Use Only | | | | |
| bulk_batch_id | bulk_batch_id | Internal Use Only | | | | |
| bulk_ interaction_id | bulk_interaction_id | Internal Use Only | | | | |
| | PROGRAM_ APPLICATION_ID | Concurrent Manager Info. | | | | Not set via API calls. |
| | PROGRAM_ID | Concurrent Manager Info. | | | | Not set via API calls. |
| | PROGRAM_UPDATE_ DATE | Concurrent Manager Info. | | | | Not set via API calls. |
| | REQUEST_ID | Concurrent Manager Info. | | | | Not set via API calls. |
| | PUBLIC_FLAG | Legacy from CS 3i interactions manager schema. | | | | Added for Service Migration from 3i schema from CS_ INTERACTIONS.PUBLIC _FLAG |
| | ORG_ID | Legacy from CS 3i interactions manager schema. | | | | Added for Service Migration from 3i schema from CS_ INTERACTIONS.ORG_ ID |
| | ORIG_SYSTEM_ REFERENCE | Pre-11i upgrade info. | | | | Not set via API calls. |
| | ORIG_SYSTEM_ REFERENCE_ID | Pre-11i upgrade info. | | | | Not set via API calls. |

| Interaction Record Value | Table column mapping in JTF_IH_ INTERACTIONS and JTF_IH_ INTERACTIONS_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| | UPG_ORIG_SYSTEM_ REF | Pre-11i upgrade info. | | | | Not set via API calls. |
| | UPG_ORIG_SYSTEM_ REF_ID | Pre-11i upgrade info. | | | | Not set via API calls. |
| | UPGRADED_STATUS_ FLAG | Pre-11i upgrade info. | | | | Not set via API calls. |
| | SECURITY_GROUP_ID | Hosting SGID | | | | Not set via API calls. |

## D.2 Activity Validations and Defaults

The following table describes the validations and defaults for all of the Activity Record Type columns that are performed when calling the IH API to Insert or update an Activity.

This includes the following API Calls:

- Create_Interaction
- Add_Activity
- Update_Activity

The JTF_IH_ACTIVITIES columns not exposed via the API are documented at the end of the table for reference purposes.

These validations apply when importing interactions from the JTF_IH_ ACTIVITIES_STG table via the Import.

| Activity Record Value | Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ ACTIVITIES_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| activity_id | ACTIVITY_ID | Unique Activity Identifier | Y | | Sequence Generated | If passed, the value passed will be used. Value passed should be generated from sequence: jtf_ih_activities_s1. If not passed, an ID is generated in Add_Activity and Create_Interaction calls. |

| Activity Record Value | Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ ACTIVITIES_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| interaction_id | INTERACTION_ID | Unique Interaction Identifier. FK to JTF_IH_ INTERACTIONS .INTERACTION_ ID | Y | | | Valid INTERACTION_ID in JTF_IH_ INTERACTIONS. |
| action_id | ACTION_ID | Identifier of the activity action code that defines the type of work done. Example: "Added", "Updated", etc. FK to JTF_IH_ ACTIONS_B. | Y | | | Valid ACTION_ID in JTF_IH_ACTIONS_B. |
| action_item_id | ACTION_ITEM_ID | Identifier of the activity action item code that defines the type of object on to which the work was done. Example: "Order", "Service Request", etc. FK to JTF_IH_ ACTION_ ITEMS_B. | Y | | | Valid ACTION_ITEM_ID in JTF_IH_ACTION_ ITEMS_B. |
| outcome_id | OUTCOME_ID | ID of outcome code assigned to the activity. FK to JTF_IH_ OUTCOMES_B. | Y | | | Valid OUTCOME_ID in JTF_IH_OUTCOMES_B |
| result_id | RESULT_ID | ID of the result code assigned to the activity. FK to JTF_IH_ RESULTS_B. | | | | Valid RESULT_ID in JTF_ IH_RESULTS_B |
| reason_id | REASON_ID | ID of the reason code assigned to the activity. FK to JTF_IH_ REASONS_B. | | | | Valid REASON_ID in JTF_IH_REASONS_B |

| Activity Record Value | Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ ACTIVITIES_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| cust_account_id | CUST_ACCOUNT_ID | The account number ID of the customer for which the activity was performed. FK to HZ_CUST_ ACCOUNTS | | | | Valid CUST_ACCOUNT_ ID in HZ_CUST_ ACCOUNTS |
| cust_org_id | CUST_ORG_ID | The ORGANIZATIO N Party_ID of the customer's business. | | | | Not validated in API. Should be a valid Party_ id in HZ_PARTIES where PARTY_TYPE = 'ORGANIZATION' |
| media_id | MEDIA_ID | The identifier of the media used to perform the interaction. FK to JTF_IH_MEDIA_ ITEMS. | | | | Valid MEDIA_ID in JTF_ IH_MEDIA_ITEMS. |
| task_id | TASK_ID | The ID of a task related to the activity. | | | | Not validated in API. Should be valid TASK_ID in JTF_TASKS_B. |
| source_code_id | SOURCE_CODE_ID | The Marketing Campaign source code identifier. FK to AMS_ SOURCE_ CODES | | | | Valid SOURCE_CODE_ ID in AMS_SOURCE_ CODES |
| source_code | SOURCE_CODE | The Marketing source code. FK to AMS_ SOURCE_ CODES | | | | Valid SOURCE_CODE in AMS_SOURCE_CODES |
| object_type | OBJECT_TYPE | The type of marketing source code. Campaign, Event, Campaign Schedule, Etc. | | | | Not Validated in the API. Should be a valid ARC_ SOURCE_CODE_FOR in AMS_SOURCE_CODES. Marketing SOURCE_ TYPE. |
| object_id | OBJECT_ID | The ID of the Campaign, Event, Etc. as qualified by OBJECT_TYPE. | | | | Not Validated in the API. Should be a valid SOURCE_CODE_FOR_ID in AMS_SOURCE_ CODES. Marketing OBJECT_ID |

| Activity Record Value | Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ACTIVITIES_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| doc_ref | DOC_REF | The object code of the type of object related to the activity. Example objects are "Order", Service Request", "Customer". FK to JTF_OBJECTS_B.OBJECT_CODE. | | | | Not Validated in the API. Should be a valid OBJECT_CODE in JTF_OBJECTS_B. |
| doc_id | DOC_ID | The unique identifier of the object designated in the doc_ref column. | | | | Not Validated in the API. Should be a valid ID in the table indicated in the JTF_OBJECTS_B.FROM_TABLE column. |
| start_date_time | START_DATE_TIME | The date and time the activity started. | | | SYSDATE | Set to SYSDATE via Add_Activity or Create_Interaction calls. If a value is passed it is used in place of SYSDATE. Must be less then or equal to END_DATE_TIME. |
| end_date_time | END_DATE_TIME | The date and time that the activity was completed. | | | SYSDATE | Set to SYSDATE via Close_Interaction or Create_Interaction calls. If a value is passed it is used in place of SYSDATE. Must be greater then or equal to START_DATE_TIME. |
| duration | DURATION | Time in seconds between the START_DATE_TIME and END_DATE_TIME | | | Calculated | If passed, the value passed will be used, if not it is calculated as: END_DATE_TIME – START_DATE_TIME. |
| description | DESCRIPTION | Free form text description of the activity. | | | | |
| doc_source_object_name | DOC_SOURCE_OBJECT_NAME | TBD | | | | Need to determine current use/need for this column. |
| interaction_action_type | INTERACTION_ACTION_TYPE | TBD | | | | Need to determine current use/need for this column. |

| Activity Record Value | Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ ACTIVITIES_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| script_trans_id | SCRIPT_TRANS_ID | ID of the Script transaction performed when performing the Interaction. | | | | Not Validated in API. Should be a valid TRANSACTION_ID in IES_TRANSACTIONS. |
| role | ROLE | TBD | | | | Need to determine current use/need for this column. |
| attribute_ category | ATTRIBUTE_ CATEGORY | Descriptive Flexfield Structured definition column | | | | |
| attribute1 | ATTRIBUTE1 | Descriptive Flexfield Segment | | | | |
| attribute2 | ATTRIBUTE2 | Descriptive Flexfield Segment | | | | |
| attribute3 | ATTRIBUTE3 | Descriptive Flexfield Segment | | | | |
| attribute4 | ATTRIBUTE4 | Descriptive Flexfield Segment | | | | |
| attribute5 | ATTRIBUTE5 | Descriptive Flexfield Segment | | | | |
| attribute6 | ATTRIBUTE6 | Descriptive Flexfield Segment | | | | |
| attribute7 | ATTRIBUTE7 | Descriptive Flexfield Segment | | | | |
| attribute8 | ATTRIBUTE8 | Descriptive Flexfield Segment | | | | |
| attribute9 | ATTRIBUTE9 | Descriptive Flexfield Segment | | | | |

| Activity Record Value | Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ ACTIVITIES_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| attribute10 | ATTRIBUTE10 | Descriptive Flexfield Segment | | | | |
| attribute11 | ATTRIBUTE11 | Descriptive Flexfield Segment | | | | |
| attribute12 | ATTRIBUTE12 | Descriptive Flexfield Segment | | | | |
| attribute13 | ATTRIBUTE13 | Descriptive Flexfield Segment | | | | |
| attribute14 | ATTRIBUTE14 | Descriptive Flexfield Segment | | | | |
| attribute15 | ATTRIBUTE15 | Descriptive Flexfield Segment | | | | |
| | ACTIVE | Y/N indicator. Y if the interaction is open, N if it is closed. | Y | | Y/N | Set to 'Y' in Add_Activity and to 'N' in Close_ Interaction calls. If Create_Interaction is used, the activity is created as non-active, N. |
| | CUST_ACCOUNT_ PARTY_ID | | | Y | | |
| | AVT_SCRIPT_ID | | | Y | | |
| | OBJECT_VERSION_ NUMBER | Standard Object Version Field | Y | | | Set to 1.0 |
| P_user_id | CREATED_BY | Standard who column - user who created this row (foreign key to FND_ USER.USER_ID) | | | | Populated using valid user id from standard API parameters |
| | CREATION_DATE | Standard who column - date when this row was created. | | | SYSDATE | |

| Activity Record Value | Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ ACTIVITIES_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| P_user_id | LAST_UPDATED_BY | Standard who column - user who last updated this row (foreign key to FND_ USER.USER_ID). | | | | Populated using valid user id from standard API parameters |
| | LAST_UPDATE_DATE | Standard Who column - date when a user last updated this row. | | | SYSDATE | |
| P_login_id | LAST_UPDATE_LOGIN | Standard who column - operating system login of user who last updated this row (foreign key to FND_ LOGINS.LOGIN _ID). | | | | Populated using valid login id from standard API parameters |
| bulk_writer_ code | bulk_writer_code | Internal Use Only | | | | |
| bulk_batch_type | bulk_batch_type | Internal Use Only | | | | |
| bulk_batch_id | bulk_batch_id | Internal Use Only | | | | |
| bulk_ interaction_id | bulk_interaction_id | Internal Use Only | | | | |
| | PROGRAM_ APPLICATION_ID | Concurrent Manager Info. | | | | Not set via API calls. |
| | PROGRAM_ID | Concurrent Manager Info. | | | | Not set via API calls. |
| | PROGRAM_UPDATE_ DATE | Concurrent Manager Info. | | | | Not set via API calls. |
| | REQUEST_ID | Concurrent Manager Info. | | | | Not set via API calls. |
| | PUBLIC_FLAG | Legacy from CS 3i interactions manager schema. | | | | Need to determine current use/need for this column. |
| | ORG_ID | Legacy from CS 3i interactions manager schema. | | | | Need to determine current use/need for this column. |

| Activity Record Value | Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ ACTIVITIES_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| | ORIG_SYSTEM_ REFERENCE | Pre-11i upgrade info. | | | | Not set via API calls. |
| | ORIG_SYSTEM_ REFERENCE_ID | Pre-11i upgrade info. | | | | Not set via API calls. |
| | UPG_ORIG_SYSTEM_ REF | Pre-11i upgrade info. | | | | Not set via API calls. |
| | UPG_ORIG_SYSTEM_ REF_ID | Pre-11i upgrade info. | | | | Not set via API calls. |
| | UPGRADED_STATUS_ FLAG | Pre-11i upgrade info. | | | | Not set via API calls. |
| | SECURITY_GROUP_ID | Hosting SGID | | | | Not set via API calls. |

## D.3 Media Item Validations and Defaults

The following table describes the validations and defaults for all of the Media Item Record Type columns that are performed when calling the IH API to Insert or update an Activity. This includes the following API Calls:

- Create_MediaItem
- Open_MediaItem
- Update_MediaItem
- Close_MediaItem

The JTF_IH_MEDIA_ITEMS columns not exposed via the API are documented at the end of the table for reference purposes.

These validations apply when importing interactions from the JTF_IH_MEDIA_ ITEMS_STG table via the Import.

| Media Item Record Value | Table column mapping in JTF_IH_MEDIA_ ITEMS and JTF_IH_ MEDIA_ITEMS_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| media_id | MEDIA_ID | Unique Media Item identifier | Y | | Sequence Generated | If passed, the value passed will be used. Value passed should be generated from sequence: jtf_ih_media_items_s1. If not passed, an ID is generated in Open_ MediaIItem and Create_ Media Item calls. |
| media_item_ type | MEDIA_ITEM_TYPE | The class of media item (TELEPHONE, EMAIL, etc.) | Y | | | Must be a non-null value. Should be a valid LOOKUP_CODE in the FND_ LOOKUPS.LOOKUP_ TYPE = 'JTF_MEDIA_ TYPE' |
| direction | DIRECTION | The direction of the MI relative to the system. INBOUND or OUTBOUND | | | | 'INBOUND' or 'OUTBOUND' or Null |
| server_group_id | SERVER_GROUP_ID | The Identifier of the Telephony server that processed the telephone type media items. | | | | Not validated in the API. ID of the SERVER_ GROUP in CCT. |
| start_date_time | START_DATE_TIME | The date and time the Media Item started. | | | SYSDATE | Set to SYSDATE via Open_MediaItem or Create_MediaItem calls. If a value is passed it is used in place of SYSDATE. Must be less then or equal to END_ DATE_TIME. |
| end_date_time | END_DATE_TIME | The date and time that the Media Item was completed. | | | SYSDATE | Set to SYSDATE via Close_ Media Item or Create_ Media Item calls. If a value is passed it is used in place of SYSDATE. Must be greater then or equal to START_DATE_TIME. |

| Media Item Record Value | Table column mapping in JTF_IH_MEDIA_ITEMS and JTF_IH_MEDIA_ITEMS_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| duration | DURATION | Time in seconds between the START_DATE_TIME and END_DATE_TIME | | | Calculated | If passed, the value passed will be used, if not it is calculated as: END_DATE_TIME – START_DATE_TIME. |
| interaction_performed | INTERACTION_PERFORMED | Flag to indicate if the MI was part of an interaction. | | | | |
| media_data | MEDIA_DATA | Media specific data based on MEDIA_ITEM_TYPE. Example: EMAIL: First 80 chars. Of the subject. | | | | Email Center places the first 80 characters of the subject on EMAIL here. |
| media_item_ref | MEDIA_ITEM_REF | Media specific data based on MEDIA_ITEM_TYPE. Example: EMAIL: Email RFC formatted ID. | | | | |
| source_id | SOURCE_ID | Media specific source ID. Example: EMAIL = Email Account ID. | | | | |
| source_item_id | SOURCE_ITEM_ID | ID of a related Media Source Item. I.e. the reference to a related object to the media_item. | | | | |
| source_item_create_date_time | SOURCE_ITEM_CREATE_DATE_TIME | The date and time the object identified by source_item_id was created. | | | | |

| Media Item Record Value | Table column mapping in JTF_IH_MEDIA_ ITEMS and JTF_IH_ MEDIA_ITEMS_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| media_ abandon_flag | MEDIA_ABANDON_ FLAG | For Telephone calls only. Indicates if the call was disconnected by the caller before and agent answered. | | | | |
| media_ transferred_flag | MEDIA_ TRANSFERRED_FLAG | For Telephone calls only. Indicates that the call was transferred. | | | | |
| dnis | DNIS | For Inbound Phone Calls only. For Inbound Calls only. The Dialed Number Identification Service (DNIS) Number. The numbered dialed by the caller. | | | | |
| ani | ANI | For Inbound Phone Calls only. The Automatic Number Identification (ANI) number. The number the caller called from. Aka. Caller ID. (for the most part) | | | | |
| classification | CLASSIFICATION | Caller or Emailer's customer classification. Example: Gold, Silver, etc. | | | | Not validated in the API. Should be a valid CLASSIFICATION_ VALUE in CCT_ CLASSIFICATION_ VALUES for Telephone Calls or a valid NAME value in IEM_ROUTE_ CLASSIFICATIONS for e-mails. |

| Media Item Record Value | Table column mapping in JTF_IH_MEDIA_ITEMS and JTF_IH_MEDIA_ITEMS_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| Address | ADDRESS | The origin address for inbound media and the target address for outbound media.<br>For Phone Calls<br>Inbound = Number dialed from (ANI)<br>Outbound = Number dialed (DNIS)<br>For Email<br>Inbound = email address sent from<br>Outbound = email address sent to<br>For Fax<br>Outbound = dialed fax number<br>For Printer<br>Outbound = printer address | | | | |
| | ACTIVE | Y/N indicator. Y if the media item is open, N if it is closed. | Y | | Y/N | Set to 'Y' in Open_MediaItem and to 'N' in Close_MediaItem calls. If Create_MediaItem is used, the Media Item is created as non-active, N. |
| | OBJECT_VERSION_NUMBER | Standard Object Version Field | Y | | | Set to 1.0 |
| P_user_id | CREATED_BY | Standard who column - user who created this row (foreign key to FND_USER.USER_ID) | | | | Populated using valid user id from standard API parameters |
| | CREATION_DATE | Standard who column - date when this row was created. | | | SYSDATE | |

| Media Item Record Value | Table column mapping in JTF_IH_MEDIA_ ITEMS and JTF_IH_ MEDIA_ITEMS_STG | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| P_user_id | LAST_UPDATED_BY | Standard who column - user who last updated this row (foreign key to FND_ USER.USER_ID). | | | | Populated using valid user id from standard API parameters |
| | LAST_UPDATE_DATE | Standard Who column - date when a user last updated this row. | | | SYSDATE | |
| P_login_id | LAST_UPDATE_LOGIN | Standard who column - operating system login of user who last updated this row (foreign key to FND_ LOGINS.LOGIN _ID). | | | | Populated using valid login id from standard API parameters |
| bulk_writer_ code | bulk_writer_code | Internal Use Only | | | | |
| bulk_batch_type | bulk_batch_type | Internal Use Only | | | | |
| bulk_batch_id | bulk_batch_id | Internal Use Only | | | | |
| bulk_ interaction_id | bulk_interaction_id | Internal Use Only | | | | |
| | SECURITY_GROUP_ID | Hosting SGID | | | | Not set via API calls. |

## D.4 Media Item Life-cycle Segment Validations and Defaults

The following table describes the validations and defaults for all of the Media Item Record Type columns that are performed when calling the IH API to Insert or update an Activity. This includes the following API Calls:

- Create_MediaLifeCycle
- Add_MediaLifecycle
- Update_MediaLifecyle
- Close_MediaItem

The JTF_IH_MEDIA_ITEM_LC_SEGS columns not exposed via the API are documented at the end of the table for reference purposes.

| Media Item LC Seg Record Value | Table column mapping in JTF_IH_MEDIA_ITEM_LC_SEGS | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| milcs_id | MILCS_ID | Unique Media Item lifecycle segment identifier | Y | | Sequence Generated | If passed, the value passed will be used. Value passed should be generated from sequence: jtf_ih_media_items_s1. If not passed, an ID is generated in Add_MediaLifecycle and Create_MediaLifecycle calls. |
| media_id | MEDIA_ID | The media item id of the media item to which the media lifecycle segment is related FK to JTF_IH_MEDIA_ITEMS | Y | | | Valid MEDIA_ID in JTF_IH_MEDIA_ITEMS |
| milcs_type_id | MILCS_TYPE_ID | The type of lifecycle segment. Example: "WITH_AGENT", "IVR" FK to JTF_IH_MEDIA_ITEM_LC_SEG_TYS | Y | | | Valid MILCS_TYPE_ID in JTF_IH_MEDIA_ITM_LC_SEG_TYS. This value is required directly or via the milcs_code value. See below. |
| start_date_time | START_DATE_TIME | The date and time the Media Item Life Cycle Segment started. | | | SYSDATE | Set to SYSDATE via Add_MediaLifecycle or Create_MediaLifecycle calls. If a value is passed it is used in place of SYSDATE. Must be less then or equal to END_DATE_TIME. |
| end_date_time | END_DATE_TIME | The date and time that the Media Item Life Cycle Segment was completed. | | | SYSDATE | Set to SYSDATE via Add_MediaLifecycle and Create_MediaLifecycle calls. If a value is passed it is used in place of SYSDATE. Must be greater then or equal to START_DATE_TIME. |

| Media Item LC Seg Record Value | Table column mapping in JTF_IH_MEDIA_ITEM_LC_SEGS | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| duration | DURATION | Time in seconds between the START_DATE_TIME and END_DATE_TIME | | | Calculated | If passed, the value passed will be used, if not it is calculated as: END_DATE_TIME – START_DATE_TIME. |
| type_type | TYPE_TYPE | Identifies the system or application responsible for handling the Lifecycle segment. | | | | |
| type_id | TYPE_ID | Identifies the object in the system or application responsible for handling the Lifecycle segment. | | | | |
| handler_id | HANDLER_ID | The application id of the application that logged the media lifecycle segment FK to FND_APPLICATION. | | | | |
| resource_id | RESOURCE_ID | ID of the JTF Resource ( agent/user) related to the media lifecycle. Used with the WITH_AGENT. FK to JTF_RS_RESOURCE_EXTN. | | | | Valid RESOURCE_ID in JTF_RS_RESOURCE_EXTN |
| milcs_code | MILCS_TYPE_ID * | Media Item Lifecycle segment code. | | | | Valid MILCS_CODE in JTF_IH_MEDIA_ITM_LC_SEG_TYS. Media Item Lifecycle segment code, translates to the Media Items Lifecycle segment ID value via a lookup and the ID is recorded on the JTF_IH_MEDIA_ITEM_LC_SEGS record |

| Media Item LC Seg Record Value | Table column mapping in JTF_IH_MEDIA_ ITEM_LC_SEGS | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| | ACTIVE | Y/N indicator. Y if the interaction is open, N if it is closed. | Y | | Y/N | Set to 'Y' in Add_ MediaLifecycle and to 'N' in Close_MediaItem calls. If Create_MediaLifecycle is used, the Media Item is created as non-active, N. |
| | OBJECT_VERSION_ NUMBER | Standard Object Version Field | Y | | | Set to 1.0 |
| P_user_id | CREATED_BY | Standard who column - user who created this row (foreign key to FND_ USER.USER_ID) | | | | Populated using valid user id from standard API parameters |
| | CREATION_DATE | Standard who column - date when this row was created. | | | SYSDATE | |
| P_user_id | LAST_UPDATED_BY | Standard who column - user who last updated this row (foreign key to FND_ USER.USER_ID). | | | | Populated using valid user id from standard API parameters |
| | LAST_UPDATE_DATE | Standard Who column - date when a user last updated this row. | | | SYSDATE | |
| P_login_id | LAST_UPDATE_LOGIN | Standard who column - operating system login of user who last updated this row (foreign key to FND_ LOGINS.LOGIN _ID). | | | | Populated using valid login id from standard API parameters |
| bulk_writer_ code | bulk_writer_code | Internal Use Only | | | | |
| bulk_batch_type | bulk_batch_type | Internal Use Only | | | | |
| bulk_batch_id | bulk_batch_id | Internal Use Only | | | | |

| Media Item LC Seg Record Value | Table column mapping in JTF_IH_MEDIA_ ITEM_LC_SEGS | Description | Req. | Obsolete | Default | Validation/Notes |
|---|---|---|---|---|---|---|
| bulk_ interaction_id | bulk_interaction_id | Internal Use Only | | | | |
| | SECURITY_GROUP_ID | Hosting SGID | | | | Not set via API calls. |

# E

# Seeded Data

Topics include:



## E.1 Outcomes

The following table lists the seeded outcomes in Oracle Customer Interaction History.

| ID | Code | Short Description | Long Description | Positive Outcome | Result Required | Generate Callback | Telephony Recycling Code |
|---|---|---|---|---|---|---|---|
| 1 | No Ans | No Answer | | N | N | Y | |
| 2 | Busy | Busy | | N | N | N | |
| 3 | Wrong Num | Wrong Number | | N | N | | 3 |
| 4 | Not Avail | Not Available | | N | N | | 4 |
| 5 | Bad Pnum | Bad Phone Number | | N | N | | 33 |
| 6 | Ans Mach | Answering Machine | | N | N | | 34 |
| 7 | Contact | Contact | | N | N | | 12 |

| ID | Code | Short Description | Long Description | Positive Outcome | Result Required | Generate Callback | Telephony Recycling Code |
|---|---|---|---|---|---|---|---|
| 8 | Decease | Decease | | N | N | | 8 |
| 9 | Maint | Maintenance | | N | N | | 0 |
| 10 | Req Proc | Request Processed | | N | N | | 10 |
| 11 | Abandoned | Abandoned | | N | N | | 35 |
| 12 | Bypass | Bypass | | N | N | | 7 |
| 13 | Change Number | Change Number | | N | N | | 6 |
| 14 | Delete | Email Deleted | | N | N | | 8 |
| 15 | Facsimile Tone | Facsimile Tone | | N | N | | 38 |
| 16 | Incoming | Incoming | | N | N | | 13 |
| 17 | Invalid for Calling | Invalid for Calling | | N | N | | 33 |
| 18 | Modem Answer Tone | Modem Answer Tone | | N | N | | 36 |
| 19 | Not Set Yet | Not Set Yet | | N | N | | 10 |
| 20 | Priority Callback | Priority Callback | | N | N | | 5 |
| 21 | Requeued | Requeued | | N | N | | 11 |
| 22 | Sit Network Busy | Sit Network Busy | | N | N | | 30 |
| 23 | Sit Operator Intercept | Sit Operator Intercept | | N | N | | 31 |
| 24 | Sit Reorder | Sit Reorder | | N | N | | 32 |
| 25 | Sit Vacant | Sit Vacant | | N | N | | 33 |
| 26 | Unidentified Sit Tone | Unidentified Sit Tone | | N | N | | 37 |
| 27 | Withdrawn During Network Time | Withdrawn During Network Time | | N | N | | 40 |
| 28 | Withdrawn During Ringing | Withdrawn During Ringing | | N | N | | 41 |
| 29 | FHM Hangup | FHM Hangup | | N | N | | 50 |
| 30 | FHM Customer Hangup | FHM Customer Hangup | | N | N | | 51 |
| 31 | Flunked Stoplist | Failed Stoplist | | N | N | | 98 |
| 32 | Unknown | Unknown | | N | N | | 0 |
| 33 | Dialer Error | Dialer Error | "The call was not placed due to a temporary dialer error." | N | N | N | 101 |

| ID | Code | Short Description | Long Description | Positive Outcome | Result Required | Generate Callback | Telephony Recycling Code |
|---|---|---|---|---|---|---|---|
| 34 | Discarded | Discarded | "The other party disconnected the call before it could be transferred." | N | N | N | 102 |
| 35 | No ringback | No ringback | "The call was placed, but no ringback was detected." | N | N | N | 103 |
| 36 | No dial tone | No dial tone | "The call was aborted because no dial tone was detected." | N | N | N | 104 |
| 37 | AO system | AO system | "This outcome code will be used for several Advanced Outbound internal system cases, which will be further qualified by the result code." | N | N | N | 105 |
| 38 | Failed release control | Failed release control | The record is recycled because it failed the user-defined release control. | N | N | N | 106 |
| 39 | Reply | Email Replied | | N | N | N | |
| 40 | Transfer | Email Transferred | | N | N | N | |
| 41 | Compose | Email Composed | | N | N | N | |
| 42 | WrANI | Wrong ANI Match | | N | N | N | |
| 43 | SysWrapUp | System Wrapped Up | | N | N | N | |
| 44 | AMS_DILG_START | Dialog is started with Customer | "The outcome of this interaction is a dialog started with a customer on the web" | N | N | N | |
| 45 | Left Message | Left Message | The agent left a message. | N | N | N | |
| 46 | Customer Abandon | Customer Abandoned Call | The customer abandoned the call. | N | N | N | |
| 47 | System Abandon | System Abandoned Call | The system abandoned the call. | N | N | N | |
| 48 | System Left Message | System Left Message | The system left a message. | N | N | N | |

| ID | Code | Short Description | Long Description | Positive Outcome | Result Required | Generate Callback | Telephony Recycling Code |
|---|---|---|---|---|---|---|---|
| 49 | Resolved | Email Automatically Resolved | The email interaction was completed without generating a response. | N | N | N | |
| 50 | Not Contacted | Not Contacted | The customer could not be contacted. | N | N | N | |
| 51 | Email Auto Redirected | Email Automatically Redirected | | N | N | N | |
| 52 | Email Auto Deleted | Email Automatically Deleted | | N | N | N | |
| 53 | Email Auto Replied | Email Automatically Replied | | N | N | N | |
| 54 | Auto Updated SR | Automatically Updated Service Request | | N | N | N | |
| 55 | Email Requeued | Email Requeued | | N | N | N | |
| 56 | Email Rerouted | Email Rerouted | | N | N | N | |
| 57 | Script_Completed | Script Completed | | Y | N | N | |
| 58 | Script_Aborted | Script Aborted | | N | N | N | |
| 59 | Script_Abandoned | Script Abandoned | | N | N | N | |
| 60 | Script_Saved_for_ Restart | Script Saved for Restart | | N | N | N | |
| 61 | Script_Restarted | Script Restarted | | N | N | N | |
| 62 | Forward | Email Forwarded | | N | N | N | |
| 63 | Resend | Email response resent | | N | N | N | |
| 65 | NOTIF_DELIVERED | Notification Delivered | | N | N | N | |

## E.2 Results

The following table lists the seeded results in Oracle Customer Interaction History.

| ID | Code | Short Description | Long Description | Positive Result | Reason Required |
|---|---|---|---|---|---|
| 1 | NoSale | No Sale | | N | N |
| 2 | Sale | Sale | | N | N |
| 3 | Compl | Customer Complaint | | N | N |

| ID | Code | Short Description | Long Description | Positive Result | Reason Required |
|---|---|---|---|---|---|
| 4 | Cb | Call Back | | N | N |
| 5 | Multi Act | Multiple Activities | | N | N |
| 6 | Completed | Completed Activity | | N | N |
| 7 | Incomplete | Incomplete Activity | | N | N |
| 8 | Sent | Message Sent | | N | N |
| 9 | Not Sent | Message not Sent | | N | N |
| 10 | Failed validation | The record failed to be validated. | | N | N |
| 11 | Cache expiration | The record expired in the cache. | | N | N |
| 12 | Email Transferred | Email Transferred | | N | N |
| 13 | Email Deleted | Message Deleted | | N | N |
| 15 | Email Sent | Email Sent | | N | N |
| 16 | Auto Updated SR | Automatically Updated Service Request | Automatically Updated the Service Request | N | N |
| 17 | Lead Created | Lead Created | A lead was created | Y | N |
| 18 | Opportunity Created | Opportunity Created | An opportunity was created | Y | N |
| 19 | Task Created | Task Created | A task was created | Y | N |
| 20 | No Contact | No Contact | The customer was not contacted | N | N |
| 21 | No Interest | No Interest | The customer was not interested. | N | N |
| 22 | Bad Data | Bad Data | Invalid data provided | N | N |
| 23 | Duplicate Record | Duplicate Record | Duplicate list record. | N | N |
| 24 | Wrong Territory | Wrong Territory | The customer is in the wrong territory | N | N |
| 25 | No Party Queried | No Party Queried | No Party Queried | N | N |
| 26 | Email Auto Resolved | Email Automatically Resolved | Email Automatically Resolved | N | N |
| 27 | Email Auto Redirected | Email Automatically Redirected | Email Automatically Redirected | N | N |
| 28 | Email Requeued | Email Requeued | Email Requeued | N | N |

| ID | Code | Short Description | Long Description | Positive Result | Reason Required |
|---|---|---|---|---|---|
| 29 | Email Rerouted Diff Class | Email Rerouted to a different Classification | Email Rerouted to a different Classification | N | N |
| 30 | Email Rerouted Diff Acct | Email Rerouted to a different Account | Email Rerouted to a different Account | N | N |
| 31 | campaign_response | Responded to Campaign | Responded to Campaign | Y | N |
| 32 | ProfileUpdated | Profile Updated | Profile Updated | Y | N |
| 33 | Redirected to an external email address | Redirected to an external email address | Redirected to an external email address | Y | N |

## E.3 Reasons

The following table lists the seeded reasons in Oracle Customer Interaction History.

| ID | Code | Short Description | Long Description |
|---|---|---|---|
| 1 | No Money | No Money | |
| 2 | Gave | Already Gave | |
| 3 | Expense | Too Expensive | |
| 4 | No Work | Out of Work | |
| 5 | Gave Office | Gave at the Office | |
| 6 | Other | Other Reason | |
| 7 | Busy | Too Busy | |
| 8 | Sp Handl Req | Special Handling Required | |
| 9 | Inquiry | General Inquiry | |
| 10 | No Resp Req | No Response Required | |
| 11 | Incorrect Agent | Incorrect Agent | |
| 12 | Wrong Phone Number | Wrong Phone Number | The phone number was invalid or incorrect. |
| 13 | Other Unknown | Other Unknown | Other Unknown |
| 14 | Not Subject Expert | Not Subject Expert | |
| 15 | Incorrect Classification | Incorrect Classification | |

| ID | Code | Short Description | Long Description |
|---|---|---|---|
| 16 | Incorrect Account | Incorrect Account | |
| 17 | Complaint | Complaint | |
| 18 | Product Information | Product Information | |
| 19 | Customer Support | Customer Support | |
| 20 | Account Information | Account Information | |
| 21 | Bounced Message | Bounced Message | |
| 22 | Spam Mail | Spam Mail | |
| 23 | Invalid Address | Invalid Email Address | |
| 24 | Customer Auto Reply | Customer Auto Reply | |
| 25 | Follow Up | Customer Follow Up | |
| 26 | Announcement | Announcement | |
| 27 | Initiate Contact | Initiate Contact | |
| 28 | Out of Office | Out of Office | |
| 29 | Escalation | Escalation | |
| 30 | AddressChange | Address Change | |
| 31 | VoiceMail | Voice Mail | |
| 32 | ProductInterest | Interested In Product | |

## E.4 Activity Types

The following table lists the seeded activity types in Oracle Customer Interaction History.

| ID | Code | Short Description |
|---|---|---|
| 1 | Acct | Account |
| 2 | Cb | Callback |
| 3 | Coll | Collateral |
| 4 | Cust | Customer |
| 5 | Cust Prod | Customer Product |
| 6 | Dr | Depot Repair |
| 7 | Event | Event |
| 8 | Lead | Customer Lead |
| 9 | Message | Message |

| ID | Code | Short Description |
|---|---|---|
| 10 | Notes | Note |
| 11 | Order | Order |
| 12 | Password | Password |
| 13 | Prod | Product |
| 14 | Quote | Quote |
| 15 | Refer | Referral |
| 16 | Rma | Return Merchandise Authorization |
| 17 | Sr | Service Request |
| 18 | Srv Ord | Service Order |
| 19 | Suggest | Suggestion |
| 20 | Stop List | Stop List |
| 21 | Opportunity | Sales Opportunity |
| 22 | Saleslead | Sales Lead |
| 23 | Partysite | Address |
| 24 | Promotion | Response to Promotion |
| 25 | Call | Call |
| 26 | Wrapup | Wrap up time |
| 27 | PAYMENT | Payment |
| 28 | DISPUTE | Dispute |
| 29 | DUNNING | Dunning |
| 30 | REVERSAL | Reversal |
| 31 | CASE | Case |
| 32 | STRATEGY | Strategy |
| 33 | DIRECTORY_ ASSISTANCE | Directory Assistance |
| 34 | PROMISE_TO_PAY | Promise to Pay |
| 35 | PAYMNT_AUTH_ ERROR | Payment Processing Authorization/Error |
| 36 | CORRESPONDENCE | Collections Correspondence |
| 37 | COLLECTIONS_ CALL | Collections Call |

| ID | Code | Short Description |
|---|---|---|
| 38 | COLLECTIONS_ CAMPAIGN | Collections Campaign |
| 39 | SCRIPT | Script |
| 40 | SURVEY | Survey |
| 41 | Party | Either organization or person |
| 42 | Enrollment | Event enrollment |
| 43 | Task | Task |
| 44 | Cancel | Cancel an enrollment |
| 45 | Email | Email message |
| 50 | Terms | Terms |
| 51 | Invoice | Invoice |
| 52 | DebitMemo | Debit Memo |
| 53 | SendCopy | Send Copy |
| 54 | Contract | Contract |
| 55 | Lease Equipment Exchange Request | Lease Equipment Exchange Request |
| 56 | Lease Convert Interest Type Request | Lease Convert Interest Type Request |
| 57 | Lease Restructure Request | Lease Restructure Request |
| 58 | Lease Termination Quote | Lease Termination Quote |
| 59 | Lease Transfer and Assumption Request | Lease Transfer and Assumption Request |
| 60 | Lease Insurance Quote | Lease Insurance Quote |
| 61 | Optional Insurance Quote | Optional Insurance Quote |
| 62 | Insurance Policy | Insurance Policy |
| 63 | Third Party Policy | Third Party Policy for Lease Contracts |
| 64 | Lease Insurance Claim | Lease Insurance Claim |
| 65 | Optional Insurance Claim | Optional Insurance Claim for Lease Contracts |
| 66 | Credit Note | Credit Note |

| ID | Code | Short Description |
|---|---|---|
| 67 | Fixed Asset | Fixed Asset |
| 68 | Service Fee | Service Fee |
| 69 | Document | Document |
| 70 | Delinquency | Delinquency |
| 71 | WriteOff | Write-Off |
| 72 | Bankruptcy | Bankruptcy |
| 73 | Litigation | Litigation |
| 74 | Repossession | Repossession |
| 75 | Case Contact | Case Contact |
| 76 | Strategy Work Item | Strategy Work Item |
| 77 | Insurance Billing Hold | Created Insurance Billing Hold |
| 78 | PACKAGE_ITEM | Package Item |
| 79 | INQUIRY | Inquiry |
| 80 | Revision Rebook | Revision Rebook |
| 81 | Web Advertisement | Web Advertisement |
| 82 | Web Offer | Web Offer |
| 83 | Email Link | Email Link |
| 84 | Web_Collaborate | Web Collaborate |
| 85 | ADJUSTMNET | Adjustment |
| 86 | Proposal | Proposal |
| 87 | Credit Request | Credit Request |
| 88 | Bill To | Bill To |
| 89 | Contact | Contact |
| 90 | ContactPoint | Contact Point |
| 91 | Relationship | Relationship |
| 92 | WebCallback | Web Callback |
| 93 | Dunning Resend | Dunning Resend |
| 94 | Dunning Callback | Dunning Callback |
| 95 | Service Hold | Service Hold |
| 96 | Mass Promise | Mass Promise |
| 97 | Credit Hold | Credit Hold |

| ID | Code | Short Description |
|---|---|---|
| 98 | Label | Label |
| 99 | Physical Collateral | Physical Collateral |
| 100 | Web Product Recommendation | Web Product Recommendation |
| 101 | Lease Asset Summary | Lease Asset Summary |
| 102 | Lease Asset Serial Number | Lease Asset Serial Number |
| 103 | Lease Asset Tax Status | Lease Asset Tax Status |
| 104 | Lease Asset Liens | Lease Asset Liens |
| 105 | Lease Asset Registration | Lease Asset Registration |

## E.5 Actions

The following table lists the seeded actions in Oracle Customer Interaction History.

| ID | Code | Short Description |
|---|---|---|
| 1 | Add | Created |
| 2 | Del | Item Deleted |
| 3 | Inq | Inquired about |
| 4 | Reconcile | Reconciliation |
| 5 | Sent | Sent |
| 6 | Upd | Updated |
| 7 | Upsell | Up Sell |
| 8 | Xsell | Cross Sell |
| 9 | Wait | Wait for the Item |
| 10 | Answer | Answered |
| 11 | Transfer | Transferred |
| 12 | Interact | Interacted with |
| 13 | Cs Sr Ins | Service request created |
| 14 | Cs Sr Upd | Service request updated |

| ID | Code | Short Description |
|---|---|---|
| 15 | Cs Sr Chg Ins | Charge created |
| 16 | Cs Sr Chg Upd | Charge updated |
| 17 | Cs Sr Chg Sub | Charge Submitted |
| 18 | Task Created | A task is created |
| 19 | Task Updated | The task is updated |
| 20 | Renew | Renew |
| 21 | Terminate | Terminate |
| 22 | Email Sent | Copy of the Email Sent |
| 23 | SCRIPT USED | Used a script to interact with a customer. |
| 24 | RESP TO SURVEY | A customer responded to a survey. |
| 25 | Rank | Ranked |
| 26 | Decline | Declined |
| 27 | Close | Closed |
| 28 | Enroll | Enrolled |
| 29 | Query | Queried |
| 30 | Email Replied | Replied to an inbound email |
| 31 | Email Deleted | Deleted an inbound email |
| 32 | Email Transferred | Transferred an inbound email |
| 33 | Email Composed | Composed a new outbound email |
| 34 | Cc Inquiry | Customer Details Inquired |
| 35 | Cc Answer | Call Answered |
| 36 | Cc Email Recd | Email Received |
| 37 | Cc Ins | New Customer Created |
| 38 | Cc Upd | Customer Information Updated |
| 39 | Cc Note Ins | Note Created |

| ID | Code | Short Description |
|---|---|---|
| 40 | Cc Note Upd | Note Updated |
| 41 | Cancel | Canceled |
| 42 | Payment Reversal | Reversal of previously processed payment |
| 43 | Credit Card | Credit Card |
| 44 | Bank Transfer | Bank Transfer |
| 45 | Promise to Pay | Promise to Pay |
| 46 | Purchase Card | Purchase Card |
| 47 | Terms Payment | Payment taken against a term |
| 48 | Invoice Dispute | Dispute taken against an invoice or bill |
| 49 | Debit Memo Dispute | Dispute taken against a debit memo |
| 50 | Send Copy | Send Copy |
| 51 | Dial Number | Dial Number |
| 52 | Create Case | Collector creates a new Case |
| 53 | Create Strategy | Collector creates a new Collections Strategy |
| 54 | Update | Update |
| 55 | Dunning | Dunning |
| 56 | Qualified | Qualified |
| 57 | Linked | Linked |
| 58 | Converted | Converted |
| 61 | Reverse | Reverse |
| 62 | Follow Up | Follow Up |
| 63 | Cc Transfer | Call Transferred |
| 64 | Cc Conferenced | Conferenced |
| 65 | "Email resolved, no reply" | Email was automatically resolved |
| 66 | Resource Assigned | A resource has been assigned |
| 67 | Resource Updated | A resource assignment has been updated |

| ID | Code | Short Description |
|---|---|---|
| 68 | Accept | Accepted |
| 69 | Activate | Activated |
| 70 | Import | Import |
| 71 | Request | Request |
| 72 | Email Auto Deleted | Email was automatically deleted |
| 73 | Email Auto Redirected | Email was automatically redirected |
| 74 | Email Auto Replied | Email was automatically replied |
| 75 | Auto Updated SR | Automatically updated service request |
| 76 | Email Requeued | Email was requeued |
| 77 | Email Rerouted Diff Class | Email was rerouted to a different classification |
| 78 | Email Rerouted Diff Acct | Email was rerouted to a different account |
| 79 | Responded To | Responded To |
| 80 | Cancel SR | Service Request Cancelled |
| 81 | Approve | Approved |
| 82 | Reject | Rejected |
| 83 | Email Auto Acknowledged | Auto Acknowledgement Email |
| 84 | Forwarded | Forwarded the email |
| 85 | Replied Again | Replied again to the inbound email |
| 86 | Email Resend | Resent an earlier email response |
| 87 | Received | Received |
| 88 | Initiate | Initiated |
| 89 | Restatus | Restatus |

| ID | Code | Short Description |
|---|---|---|
| 90 | Automated Email Sent | Email sent to Customer Contact by Automated system |
| 92 | Printed | Printed |
| 93 | Not Printed | Not Printed |
| 94 | Not Sent | Not Sent |

## E.6 Wrap Ups

The following table lists the seeded wrap ups in Oracle Customer Interaction History.

| ID | Outcome ID | Outcome | Result ID | Result | Reason ID | Reason | Effective Start Date | Effective End Date | Level |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | Busy | 0 | | 0 | | 7/31/2002 | 3/26/2004 | BOTH |
| 2 | 1 | No Answer | 0 | | 0 | | 7/31/2002 | | BOTH |
| 3 | 3 | Wrong Number | 0 | | 0 | | 7/31/2002 | | BOTH |
| 4 | 4 | Not Available | 0 | | 0 | | 7/31/2002 | | BOTH |
| 5 | 6 | Answering Machine | 0 | | 0 | | 7/31/2002 | | BOTH |
| 6 | 7 | Contact | 0 | | 0 | | 7/31/2002 | | BOTH |
| 7 | 7 | Contact | 2 | Sale | 0 | | 7/31/2002 | | BOTH |
| 8 | 7 | Contact | 1 | No Sale | 1 | No Money | 7/31/2002 | | BOTH |
| 9 | 7 | Contact | 1 | No Sale | 3 | Too Expensive | 7/31/2002 | | BOTH |
| 10 | 7 | Contact | 1 | No Sale | 7 | Too Busy | 7/31/2002 | | BOTH |
| 11 | 7 | Contact | 1 | No Sale | 6 | Other Reason | 7/31/2002 | | BOTH |
| 12 | 39 | Email Replied | 15 | Email Sent | 6 | Other Reason | 11/5/2002 | | BOTH |
| 13 | 14 | Email Deleted | 13 | Message Deleted | 10 | No Response Required | 11/5/2002 | | BOTH |
| 14 | 40 | Email Transferred | 12 | Email Transferred | 14 | Not Subject Expert | 11/5/2002 | | BOTH |
| 15 | 41 | Email Composed | 15 | Email Sent | 6 | Other Reason | 11/5/2002 | | BOTH |

| ID | Outcome ID | Outcome | Result ID | Result | Reason ID | Reason | Effective Start Date | Effective End Date | Level |
|---|---|---|---|---|---|---|---|---|---|
| 16 | 56 | Email Rerouted | 30 | Email Rerouted to a different Account | 16 | Incorrect Account | 11/5/2002 | | BOTH |
| 17 | 56 | Email Rerouted | 29 | Email Rerouted to a different Classification | 15 | Incorrect Classification | 11/5/2002 | | BOTH |
| 18 | 55 | Email Requeued | 28 | Email Requeued | 14 | Not Subject Expert | 11/5/2002 | | BOTH |
| 19 | 54 | Automatically Updated Service Request | 16 | Automatically Updated Service Request | 0 | | 11/5/2002 | 11/5/2002 | BOTH |
| 20 | 53 | Email Automatically Replied | 15 | Email Sent | 0 | | 11/5/2002 | 11/5/2002 | BOTH |
| 21 | 52 | Email Automatically Deleted | 13 | Message Deleted | 0 | | 11/5/2002 | 11/5/2002 | BOTH |
| 22 | 51 | Email Automatically Redirected | 33 | Redirected to an external email address | 0 | | 11/5/2002 | 11/5/2002 | BOTH |
| 23 | 49 | Email Automatically Resolved | 26 | Email Automatically Resolved | 0 | | 11/5/2002 | 11/5/2002 | BOTH |
| 24 | 39 | Email Replied | 15 | Email Sent | 9 | General Inquiry | 11/20/2002 | | BOTH |
| 25 | 39 | Email Replied | 15 | Email Sent | 17 | Complaint | 11/20/2002 | | BOTH |
| 26 | 39 | Email Replied | 15 | Email Sent | 18 | Product Information | 11/20/2002 | | BOTH |
| 27 | 39 | Email Replied | 15 | Email Sent | 19 | Customer Support | 11/20/2002 | | BOTH |
| 28 | 39 | Email Replied | 15 | Email Sent | 20 | Account Information | 11/20/2002 | | BOTH |
| 29 | 14 | Email Deleted | 13 | Message Deleted | 6 | Other Reason | 11/20/2002 | | BOTH |
| 30 | 14 | Email Deleted | 13 | Message Deleted | 21 | Bounced Message | 11/20/2002 | | BOTH |
| 31 | 14 | Email Deleted | 13 | Message Deleted | 22 | Spam Mail | 11/20/2002 | | BOTH |

| ID | Outcome ID | Outcome | Result ID | Result | Reason ID | Reason | Effective Start Date | Effective End Date | Level |
|---|---|---|---|---|---|---|---|---|---|
| 32 | 14 | Email Deleted | 13 | Message Deleted | 23 | Invalid Email Address | 11/20/2002 | | BOTH |
| 33 | 14 | Email Deleted | 13 | Message Deleted | 24 | Customer Auto Reply | 11/20/2002 | | BOTH |
| 34 | 41 | Email Composed | 15 | Email Sent | 25 | Customer Follow Up | 11/20/2002 | | BOTH |
| 35 | 41 | Email Composed | 15 | Email Sent | 26 | Announcement | 11/20/2002 | | BOTH |
| 36 | 41 | Email Composed | 15 | Email Sent | 27 | Initiate Contact | 11/20/2002 | | BOTH |
| 37 | 40 | Email Transferred | 12 | Email Transferred | 6 | Other Reason | 11/20/2002 | | BOTH |
| 38 | 40 | Email Transferred | 12 | Email Transferred | 28 | Out of Office | 11/20/2002 | | BOTH |
| 39 | 40 | Email Transferred | 12 | Email Transferred | 29 | Escalation | 11/20/2002 | | BOTH |
| 40 | 55 | Email Requeued | 28 | Email Requeued | 6 | Other Reason | 11/20/2002 | | BOTH |
| 41 | 55 | Email Requeued | 28 | Email Requeued | 28 | Out of Office | 11/20/2002 | | BOTH |
| 42 | 56 | Email Rerouted | 30 | Email Rerouted to a different Account | 6 | Other Reason | 11/20/2002 | | BOTH |
| 43 | 56 | Email Rerouted | 29 | Email Rerouted to a different Classification | 6 | Other Reason | 11/20/2002 | | BOTH |
| 44 | 62 | Email Forwarded | 15 | Email Sent | 0 | | 3/3/2003 | | BOTH |
| 45 | 63 | Email response resent | 15 | Email Sent | 0 | | 3/3/2003 | | BOTH |
| 46 | 7 | Contact | 17 | Lead Created | 32 | Interested In Product | 7/31/2003 | | BOTH |
| 47 | 7 | Contact | 18 | Opportunity Created | 32 | Interested In Product | 7/31/2003 | | BOTH |
| 48 | 7 | Contact | 32 | Profile Updated | 30 | Address Change | 7/31/2003 | | BOTH |
| 49 | 1 | No Answer | 1 | No Sale | 31 | Voice Mail | 7/31/2003 | | BOTH |

# E.7 Action-Activity Type Associations

The following table lists the seeded action-activity type associations in Oracle Customer Interaction History.

| Action ID | Action | Activity Type ID | Activity Type | Default Wrap ID | Outcome | Result | Reason |
|---|---|---|---|---|---|---|---|
| 1 | Add | 1 | Account | | | | |
| 6 | Upd | 1 | Account | | | | |
| 29 | Query | 1 | Account | | | | |
| 68 | Accept | 1 | Account | | | | |
| 1 | Add | 3 | Collateral | | | | |
| 5 | Sent | 3 | Collateral | | | | |
| 92 | Printed | 3 | Collateral | | | | |
| 93 | Not Printed | 3 | Collateral | | | | |
| 94 | Not Sent | 3 | Collateral | | | | |
| 1 | Add | 4 | Customer | | | | |
| 3 | Inq | 4 | Customer | | | | |
| 6 | Upd | 4 | Customer | | | | |
| 29 | Query | 4 | Customer | 48 | Contact | Profile Updated | Address Change |
| 51 | Dial Number | 4 | Customer | | | | |
| 28 | Enroll | 7 | Event | | | | |
| 29 | Query | 7 | Event | | | | |
| 1 | Add | 8 | Customer Lead | 46 | Contact | Lead Created | Interested In Product |
| 6 | Upd | 8 | Customer Lead | | | | |
| 25 | Rank | 8 | Customer Lead | | | | |
| 26 | Decline | 8 | Customer Lead | | | | |
| 29 | Query | 8 | Customer Lead | | | | |
| 1 | Add | 10 | Note | | | | |
| 6 | Upd | 10 | Note | | | | |
| 3 | Inq | 11 | Order | | | | |
| 3 | Inq | 13 | Product | | | | |
| 5 | Sent | 14 | Quote | | | | |

| Action ID | Action | Activity Type ID | Activity Type | Default Wrap ID | Outcome | Result | Reason |
|---|---|---|---|---|---|---|---|
| 1 | Add | 17 | Service Request | | | | |
| 3 | Inq | 17 | Service Request | | | | |
| 6 | Upd | 17 | Service Request | | | | |
| 13 | Cs Sr Ins | 17 | Service Request | | | | |
| 90 | Automated Email Sent | 17 | Service Request | | | | |
| 1 | Add | 21 | Sales Opportunity | 47 | Contact | Opportunity Created | Interested In Product |
| 27 | Close | 21 | Sales Opportunity | | | | |
| 29 | Query | 21 | Sales Opportunity | | | | |
| 3 | Inq | 23 | Address | | | | |
| 6 | Upd | 23 | Address | | | | |
| 10 | Answer | 25 | Call | | | | |
| 11 | Transfer | 25 | Call | | | | |
| 51 | Dial Number | 25 | Call | 49 | No Answer | No Sale | Voice Mail |
| 64 | Cc Conferenced | 25 | Call | | | | |
| 88 | Initiate | 25 | Call | | | | |
| 1 | Add | 27 | Payment | | | | |
| 1 | Add | 28 | Dispute | | | | |
| 6 | Upd | 28 | Dispute | | | | |
| 5 | Sent | 29 | Dunning | | | | |
| 1 | Add | 30 | Reversal | | | | |
| 6 | Upd | 30 | Reversal | | | | |
| 1 | Add | 31 | Case | | | | |
| 1 | Add | 32 | Strategy | | | | |
| 26 | Decline | 32 | Strategy | | | | |
| 27 | Close | 32 | Strategy | | | | |
| 29 | Query | 33 | Directory Assistance | | | | |
| 1 | Add | 34 | Promise to Pay | | | | |

| Action ID | Action | Activity Type ID | Activity Type | Default Wrap ID | Outcome | Result | Reason |
|---|---|---|---|---|---|---|---|
| 41 | Cancel | 34 | Promise to Pay | | | | |
| 89 | Restatus | 34 | Promise to Pay | | | | |
| 12 | Interact | 35 | Payment Processing Authorization/Error | | | | |
| 1 | Add | 36 | Collections Correspondence | | | | |
| 1 | Add | 37 | Collections Call | | | | |
| 1 | Add | 38 | Collections Campaign | | | | |
| 1 | Add | 39 | Script | | | | |
| 23 | SCRIPT USED | 39 | Script | | | | |
| 24 | RESP TO SURVEY | 40 | Survey | | | | |
| 1 | Add | 43 | Task | | | | |
| 6 | Upd | 43 | Task | | | | |
| 54 | Update | 43 | Task | | | | |
| 66 | Resource Assigned | 43 | Task | | | | |
| 67 | Resource Updated | 43 | Task | | | | |
| 22 | Email Sent | 45 | Email message | 24 | Email Replied | Email Sent | General Inquiry |
| 30 | Email Replied | 45 | Email message | 24 | Email Replied | Email Sent | General Inquiry |
| 31 | Email Deleted | 45 | Email message | 13 | Email Deleted | Message Deleted | No Response Required |
| 32 | Email Transferred | 45 | Email message | 14 | Email Transferred | Email Transferred | Not Subject Expert |
| 33 | Email Composed | 45 | Email message | 34 | Email Composed | Email Sent | Customer Follow Up |
| 65 | "Email resolved, no reply" | 45 | Email message | 23 | Email Automatically Resolved | Email Automatically Resolved | |
| 72 | Email Auto Deleted | 45 | Email message | 21 | Email Automatically Deleted | Message Deleted | |

| Action ID | Action | Activity Type ID | Activity Type | Default Wrap ID | Outcome | Result | Reason |
|---|---|---|---|---|---|---|---|
| 73 | Email Auto Redirected | 45 | Email message | 22 | Email Automatically Redirected | Redirected to an external email address | |
| 74 | Email Auto Replied | 45 | Email message | 20 | Email Automatically Replied | Email Sent | |
| 75 | Auto Updated SR | 45 | Email message | 19 | Automatically Updated Service Request | Automatically Updated Service Request | |
| 76 | Email Requeued | 45 | Email message | 18 | Email Requeued | Email Requeued | Not Subject Expert |
| 77 | Email Rerouted Diff Class | 45 | Email message | 17 | Email Rerouted | Email Rerouted to a different Classification | Incorrect Classification |
| 78 | Email Rerouted Diff Acct | 45 | Email message | 16 | Email Rerouted | Email Rerouted to a different Account | Incorrect Account |
| 83 | Email Auto Acknowledged | 45 | Email message | | | | |
| 84 | Forwarded | 45 | Email message | 44 | Email Forwarded | Email Sent | |
| 85 | Replied Again | 45 | Email message | | | | |
| 86 | Email Resend | 45 | Email message | 45 | Email response resent | Email Sent | |
| 3 | Inq | 51 | Invoice | | | | |
| 3 | Inq | 54 | Contract | | | | |
| 18 | Task Created | 54 | Contract | | | | |
| 19 | Task Updated | 54 | Contract | | | | |
| 29 | Query | 54 | Contract | | | | |
| 39 | Cc Note Ins | 54 | Contract | | | | |
| 40 | Cc Note Upd | 54 | Contract | | | | |
| 1 | Add | 55 | Lease Equipment Exchange Request | | | | |
| 54 | Update | 55 | Lease Equipment Exchange Request | | | | |

| Action ID | Action | Activity Type ID | Activity Type | Default Wrap ID | Outcome | Result | Reason |
|---|---|---|---|---|---|---|---|
| 1 | Add | 56 | Lease Convert Interest Type Request | | | | |
| 54 | Update | 56 | Lease Convert Interest Type Request | | | | |
| 1 | Add | 57 | Lease Restructure Request | | | | |
| 54 | Update | 57 | Lease Restructure Request | | | | |
| 1 | Add | 58 | Lease Termination Quote | | | | |
| 54 | Update | 58 | Lease Termination Quote | | | | |
| 1 | Add | 59 | Lease Transfer and Assumption Request | | | | |
| 54 | Update | 59 | Lease Transfer and Assumption Request | | | | |
| 1 | Add | 60 | Lease Insurance Quote | | | | |
| 68 | Accept | 60 | Lease Insurance Quote | | | | |
| 1 | Add | 61 | Optional Insurance Quote | | | | |
| 68 | Accept | 61 | Optional Insurance Quote | | | | |
| 1 | Add | 62 | Insurance Policy | | | | |
| 2 | Del | 62 | Insurance Policy | | | | |
| 41 | Cancel | 62 | Insurance Policy | | | | |
| 1 | Add | 63 | Third Party Policy for Lease Contracts | | | | |

| Action ID | Action | Activity Type ID | Activity Type | Default Wrap ID | Outcome | Result | Reason |
|---|---|---|---|---|---|---|---|
| 54 | Update | 63 | Third Party Policy for Lease Contracts | | | | |
| 1 | Add | 64 | Lease Insurance Claim | | | | |
| 1 | Add | 65 | Optional Insurance Claim for Lease Contracts | | | | |
| 1 | Add | 66 | Credit Note | | | | |
| 54 | Update | 67 | Fixed Asset | | | | |
| 1 | Add | 68 | Service Fee | | | | |
| 5 | Sent | 69 | Document | | | | |
| 1 | Add | 71 | Write-Off | | | | |
| 1 | Add | 72 | Bankruptcy | | | | |
| 1 | Add | 73 | Litigation | | | | |
| 1 | Add | 74 | Repossession | | | | |
| 1 | Add | 76 | Strategy Work Item | | | | |
| 89 | Restatus | 76 | Strategy Work Item | | | | |
| 1 | Add | 77 | Created Insurance Billing Hold | | | | |
| 1 | Add | 78 | Package Item | | | | |
| 5 | Sent | 78 | Package Item | | | | |
| 71 | Request | 78 | Package Item | | | | |
| 1 | Add | 79 | Inquiry | | | | |
| 6 | Upd | 79 | Inquiry | | | | |
| 70 | Import | 79 | Inquiry | | | | |
| 79 | Responded To | 81 | Web Advertisement | | | | |
| 79 | Responded To | 82 | Web Offer | | | | |
| 79 | Responded To | 83 | Email Link | | | | |
| 12 | Interact | 84 | Web Collaborate | | | | |

| Action ID | Action | Activity Type ID | Activity Type | Default Wrap ID | Outcome | Result | Reason |
|---|---|---|---|---|---|---|---|
| 1 | Add | 85 | Adjustment | | | | |
| 5 | Sent | 86 | Proposal | | | | |
| 81 | Approve | 87 | Credit Request | | | | |
| 82 | Reject | 87 | Credit Request | | | | |
| 3 | Inq | 88 | Bill To | | | | |
| 1 | Add | 89 | Contact | | | | |
| 1 | Add | 90 | Contact Point | | | | |
| 6 | Upd | 90 | Contact Point | | | | |
| 1 | Add | 91 | Relationship | | | | |
| 6 | Upd | 91 | Relationship | | | | |
| 87 | Received | 92 | Web Callback | | | | |
| 1 | Add | 93 | Dunning Resend | | | | |
| 1 | Add | 94 | Dunning Callback | | | | |
| 1 | Add | 95 | Service Hold | | | | |
| 1 | Add | 96 | Mass Promise | | | | |
| 1 | Add | 97 | Credit Hold | | | | |
| 92 | Printed | 98 | Label | | | | |
| 93 | Not Printed | 98 | Label | | | | |
| 5 | Sent | 99 | Physical Collateral | | | | |
| 94 | Not Sent | 99 | Physical Collateral | | | | |
| 79 | Responded To | 100 | Web Product Recommendation | | | | |
| 6 | Upd | 101 | Lease Asset Summary | | | | |
| 6 | Upd | 102 | Lease Asset Serial Number | | | | |
| 6 | Upd | 103 | Lease Asset Tax Status | | | | |
| 6 | Upd | 104 | Lease Asset Liens | | | | |
| 6 | Upd | 105 | Lease Asset Registration | | | | |