Oracle® Customer Interaction History

Implementation Guide

Release 12.2

Part No. E48930-01

September 2013

ORACLE

Oracle Customer Interaction History Implementation Guide, Release 12.2

Part No. E48930-01

Primary Author: Ashita Mathur

Contents

5 Using Oracle Customer Interaction History

6 Data Migration

7 Concurrent Programs

8 API Reference

Send Us Your Comments

Oracle Customer Interaction History Implementation Guide, Release 12.2

Part No. E48930-01

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document. Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

Note: Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the new Oracle E-Business Suite Release Online Documentation CD available on My Oracle Support and www.oracle.com. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: appsdoc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at www.oracle.com.

Preface

Intended Audience

Welcome to Release 12.2 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
- Oracle Applications
- The Oracle Applications graphical user interface

See Related Information Sources on page x for more Oracle E-Business Suite product information.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Structure

Related Information Sources

Integration Repository

The Oracle Integration Repository is a compilation of information about the service endpoints exposed by the Oracle E-Business Suite of applications. It provides a complete catalog of Oracle E-Business Suite's business service interfaces. The tool lets users easily discover and deploy the appropriate business service interface for integration with any system, application, or business partner.

The Oracle Integration Repository is shipped as part of the E-Business Suite. As your instance is patched, the repository is automatically updated with content appropriate for the precise revisions of interfaces in your environment.

You can navigate to the Oracle Integration Repository through Oracle E-Business Suite Integrated SOA Gateway.

Online Documentation

All Oracle E-Business Suite documentation is available online (HTML or PDF).

- **PDF** - See the Oracle E-Business Suite Documentation Library for current PDF documentation for your product with each release. The Oracle E-Business Suite Documentation Library is also available on My Oracle Support and is updated frequently

- **Online Help** - Online help patches (HTML) are available on My Oracle Support.

- **Release Notes** - For information about changes in this release, including new features, known issues, and other details, see the release notes for the relevant product, available on My Oracle Support.

- **Oracle Electronic Technical Reference Manual -** The Oracle Electronic Technical Reference Manual (eTRM) contains database diagrams and a detailed description of database tables, forms, reports, and programs for each Oracle E-Business Suite product. This information helps you convert data from your existing applications and integrate Oracle E-Business Suite data with non-Oracle applications, and write custom reports for Oracle E-Business Suite products. The Oracle eTRM is available on My Oracle Support.

Guides Related to All Products

Oracle E-Business Suite User's Guide

This guide explains how to navigate, enter data, query, and run reports using the user interface (UI) of Oracle E-Business Suite. This guide also includes information on setting user profiles, as well as running and reviewing concurrent programs.

You can access this guide online by choosing "Getting Started with Oracle Applications" from any Oracle E-Business Suite product help file.

Guides Related to This Product

Oracle Interaction Center Server Manager Implementation Guide

Oracle Interaction Center Server Manager is the Java server process that runs on every Oracle Interaction Center machine. It enables you to balance load by creating server groups and configuring multiple server processes for them or assign servers to a standby system of nodes that you create by installing and running Oracle Interaction Center Server Manager on computers. You can add IP addresses to a node, refresh parameters of servers and server groups while the server is working, and use the Unit Test Server utility to test the validity of the CTI and switch integration. This guide also describes how to diagnose and troubleshoot operational issues.

Installation and System Administration

Oracle Alert User's Guide

This guide explains how to define periodic and event alerts to monitor the status of your Oracle E-Business Suite data.

Oracle E-Business Suite Concepts

This book is intended for all those planning to deploy Oracle E-Business Suite Release 12.2, or contemplating significant changes to a configuration. After describing the Oracle E-Business Suite architecture and technology stack, it focuses on strategic topics, giving a broad outline of the actions needed to achieve a particular goal, plus the installation and configuration choices that may be available.

Oracle E-Business Suite CRM System Administrator's Guide

This manual describes how to implement the CRM Technology Foundation (JTT) and use its System Administrator Console.

Oracle E-Business Suite Developer's Guide

This guide contains the coding standards followed by the Oracle E-Business Suite development staff. It describes the Oracle Application Object Library components needed to implement the Oracle E-Business Suite user interface described in the *Oracle E-Business Suite User Interface Standards for Forms-Based Products*. It also provides information to help you build your custom Oracle Forms Developer forms so that they integrate with Oracle E-Business Suite. In addition, this guide has information for customizations in features such as concurrent programs, flexfields, messages, and logging.

Oracle E-Business Suite Installation Guide: Using Rapid Install

This book is intended for use by anyone who is responsible for installing or upgrading Oracle E-Business Suite. It provides instructions for running Rapid Install either to carry out a fresh installation of Oracle E-Business Suite Release 12.2, or as part of an upgrade to Release 12.2.

Oracle E-Business Suite Maintenance Guide

This guide contains information about the strategies, tasks, and troubleshooting activities that can be used to help ensure an Oracle E-Business Suite system keeps running smoothly, together with a comprehensive description of the relevant tools and utilities. It also describes how to patch a system, with recommendations for optimizing typical patching operations and reducing downtime.

Oracle E-Business Suite Security Guide

This guide contains information on a comprehensive range of security-related topics, including access control, user management, function security, data security, and auditing. It also describes how Oracle E-Business Suite can be integrated into a single sign-on environment.

Oracle E-Business Suite Setup Guide

This guide contains information on system configuration tasks that are carried out either after installation or whenever there is a significant change to the system. The activities described include defining concurrent programs and managers, enabling Oracle Applications Manager features, and setting up printers and online help.

Oracle E-Business Suite User Interface Standards for Forms-Based Products

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

Other Implementation Documentation

Oracle Approvals Management Implementation Guide

This guide describes transaction attributes, conditions, actions, and approver groups that you can use to define approval rules for your business. These rules govern the process for approving transactions in an integrated Oracle application. You can define approvals by job, supervisor hierarchy, positions, or by lists of individuals created either at the time you set up the approval rule or generated dynamically when the rule is invoked. You can learn how to link different approval methods together and how to run approval processes in parallel to shorten transaction approval process time.

Oracle Diagnostics Framework User's Guide

This guide contains information on implementing, administering, and developing diagnostics tests for Oracle E-Business Suite using the Oracle Diagnostics Framework.

Oracle E-Business Suite Flexfields Guide

This guide provides flexfields planning, setup and reference information for the Oracle E-Business Suite implementation team, as well as for users responsible for the ongoing maintenance of Oracle E-Business Suite product data. This guide also provides information on creating custom reports on flexfields data.

Oracle E-Business Suite Integrated SOA Gateway Implementation Guide

This guide explains the details of how integration repository administrators can manage and administer the entire service enablement process based on the service-oriented architecture (SOA) for both native packaged public integration interfaces and composite services - BPEL type. It also describes how to invoke Web services from Oracle E-Business Suite by working with Oracle Workflow Business Event System, manage Web service security, and monitor SOAP messages.

Oracle E-Business Suite Integrated SOA Gateway User's Guide

This guide describes how users can browse and view the integration interface definitions and services that reside in Oracle Integration Repository.

Oracle E-Business Suite Multiple Organizations Implementation Guide

This guide describes how to set up multiple organizations and the relationships among them in a single installation of an Oracle E-Business Suite product such that transactions flow smoothly through and among organizations that can be ledgers, business groups, legal entities, operating units, or inventory organizations. You can use this guide to assign operating units to a security profile and assign this profile to responsibilities such that a user can access data for multiple operating units from a single responsibility. In addition, this guide describes how to set up reporting to generate reports at different

levels and for different contexts. Reporting levels can be ledger or operating unit while reporting context is a named entity in the selected reporting level.

Oracle e-Commerce Gateway Implementation Guide

This guide describes implementation details, highlighting additional setup steps needed for trading partners, code conversion, and Oracle E-Business Suite. It also provides architecture guidelines for transaction interface files, troubleshooting information, and a description of how to customize EDI transactions.

Oracle e-Commerce Gateway User's Guide

This guide describes the functionality of Oracle e-Commerce Gateway and the necessary setup steps in order for Oracle E-Business Suite to conduct business with trading partners through Electronic Data Interchange (EDI). It also describes how to run extract programs for outbound transactions, import programs for inbound transactions, and the relevant reports.

Oracle iSetup User's Guide

This guide describes how to use Oracle iSetup to migrate data between different instances of the Oracle E-Business Suite and generate reports. It also includes configuration information, instance mapping, and seeded templates used for data migration.

Oracle Product Hub Implementation Guide

This guide explains how to set up hierarchies of items using catalogs and catalog categories and then to create user-defined attributes to capture all of the detailed information (such as cost information) about an object (such as an item or change order). It also explains how to set up optional features used in specific business cases; choose which features meet your business' needs. Finally, the guide explains the set up steps required to link to third party and legacy applications, then synchronize and enrich the data in a master product information repository.

Oracle Product Hub User's Guide

This guide explains how to centrally manage item information across an enterprise, focusing on product data consolidation and quality. The item information managed includes item attributes, categorization, organizations, suppliers, multilevel structures/bills of material, packaging, changes, attachments, and reporting.

Oracle Web Applications Desktop Integrator Implementation and Administration Guide

Oracle Web Applications Desktop Integrator brings Oracle E-Business Suite functionality to a spreadsheet, where familiar data entry and modeling techniques can be used to complete Oracle E-Business Suite tasks. You can create formatted spreadsheets on your desktop that allow you to download, view, edit, and create Oracle

E-Business Suite data, which you can then upload. This guide describes how to implement Oracle Web Applications Desktop Integrator and how to define mappings, layouts, style sheets, and other setup options.

Oracle Workflow Administrator's Guide

This guide explains how to complete the setup steps necessary for any Oracle E-Business Suite product that includes workflow-enabled processes. It also describes how to manage workflow processes and business events using Oracle Applications Manager, how to monitor the progress of runtime workflow processes, and how to administer notifications sent to workflow users.

Oracle Workflow Developer's Guide

This guide explains how to define new workflow business processes and customize existing workflow processes embedded in Oracle E-Business Suite. It also describes how to define and customize business events and event subscriptions.

Oracle Workflow User's Guide

This guide describes how Oracle E-Business Suite users can view and respond to workflow notifications and monitor the progress of their workflow processes.

Oracle XML Gateway User's Guide

This guide describes Oracle XML Gateway functionality and each component of the Oracle XML Gateway architecture, including Message Designer, Oracle XML Gateway Setup, Execution Engine, Message Queues, and Oracle Transport Agent. It also explains how to use Collaboration History that records all business transactions and messages exchanged with trading partners.

The integrations with Oracle Workflow Business Event System, and the Business-to-Business transactions are also addressed in this guide.

Oracle XML Publisher Administration and Developer's Guide

Oracle XML Publisher is a template-based reporting solution that merges XML data with templates in RTF or PDF format to produce outputs to meet a variety of business needs. Outputs include: PDF, HTML, Excel, RTF, and eText (for EDI and EFT transactions). Oracle XML Publisher can be used to generate reports based on existing Oracle E-Business Suite report data, or you can use Oracle XML Publisher's data extraction engine to build your own queries. Oracle XML Publisher also provides a robust set of APIs to manage delivery of your reports via e-mail, fax, secure FTP, printer, WebDav, and more. This guide describes how to set up and administer Oracle XML Publisher as well as how to use the Application Programming Interface to build custom solutions. This guide is available through the Oracle E-Business Suite online help.

Oracle XML Publisher Report Designer's Guide

Oracle XML Publisher is a template-based reporting solution that merges XML data with templates in RTF or PDF format to produce a variety of outputs to meet a variety of business needs. Using Microsoft Word or Adobe Acrobat as the design tool, you can create pixel-perfect reports from the Oracle E-Business Suite. Use this guide to design your report layouts. This guide is available through the Oracle E-Business Suite online help.

Training and Support

Training

Oracle offers a complete set of training courses to help you master your product and reach full productivity quickly. These courses are organized into functional learning paths, so you take only those courses appropriate to your job or area of responsibility.

You have a choice of educational environments. You can attend courses offered by Oracle University at any 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 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 your product 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 Oracle server, and your hardware and software environment.

Do Not Use Database Tools to Modify Oracle E-Business Suite Data

Oracle STRONGLY RECOMMENDS that you never use SQL*Plus, Oracle Data Browser, database triggers, or any other tool to modify Oracle E-Business Suite 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 E-Business Suite data, you risk destroying the integrity of your data and you lose the ability to audit changes to your data.

Because Oracle E-Business Suite tables are interrelated, any change you make using an Oracle E-Business Suite form can update many tables at once. But when you modify Oracle E-Business Suite data using anything other than Oracle E-Business Suite, 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 E-Business Suite.

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

1

Introduction

This chapter covers the following topics:

- Overview of Oracle Customer Interaction History

Overview of 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.

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.

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.

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.

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.

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.

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.

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.

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").

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.

2

Dependencies and Integration Points

This chapter covers the following topics:

- Mandatory Dependencies
- Accessing Oracle Customer Interaction History

Mandatory Dependencies

Install and implement the following components before you begin the implementation of Oracle Customer Interaction History:

- Resource Manager
- Task Manager
- Notes

Accessing Oracle Customer Interaction History

The product interfaces are accessed by providing the Uniform Resource Locator (URL) for the environment in an Oracle Applications 12*i*-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
```

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.

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. 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: • Launch the Customer Interaction History Test page form. • 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

Responsibility	Function	Interface
Interaction History JSP Admin	Access the administration console: • Create outcomes, results, reasons, activities, actions, and wrap ups. • Associate actions and activities. • View bulk processing errors.	E-Business Home Page or CRM Home Page
Interaction History JSP User	View interactions in the JSP interface: • Launch the Interaction History Viewer page. • View interactions and activities. • Search for interactions and activities.	E-Business Home Page or 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 or CRM Home Page
Interaction History Self Service	View interactions in the Oracle Applications Framework interface.	E-Business Home Page or CRM Home Page
System Administrator	Create a user for administering Oracle Customer Interaction History. Set values for profile options.	E-Business Home Page or 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.

3

Implementation Tasks

This chapter covers the following topics:

- Implementation Task Sequence
- Defining an Administrator

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.** You will use this user account to perform the remaining steps.	System Administrator
2	No	**Migrate outcomes-result pairs and result-reason pairs to wrap-ups tables.**	Interaction History Migration

Step	Required	Description	Responsibility
3	No	**Define interaction outcomes.** See:	Interaction History JSP Administrator
4	No	**Define interaction results.**	Interaction History JSP Administrator
5	No	**Define interaction reason codes.**	Interaction History JSP Administrator
6	No	**Define interaction activity types.**	Interaction History JSP Administrator
7	No	**Define interaction actions.**	Interaction History JSP Administrator
8	No	**Define interaction wrap ups.**	Interaction History JSP Administrator
9	No	**Associate an action with an activity type.**	Interaction History JSP Administrator
10	For applications that use One-to-One Fulfillment	**Set up Interaction History Bulk Processor.**	CRM Administrator

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. 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 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: • Create outcomes, results, reasons, activities, actions, and wrap ups. • Associate actions and activities. • View bulk processing errors.	E-Business Home Page or 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 or CRM Home Page

After use save the user record, 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.

4

Administration Tasks

This chapter covers the following topics:

- Outcome
- Creating an Outcome
- Updating an Outcome
- Removing an Outcome
- List of Results
- Creating a Result
- Updating a Result
- Removing a Result
- List of Reasons
- Creating a Reason
- Updating a Reason
- Removing a Reason
- Activity Type
- Creating an Activity Type
- Updating an Activity Type
- Removing an Activity Type
- List of Actions
- Creating an Action
- Updating an Action
- Removing an Action
- Wrap Up
- Creating a Wrap Up

- Updating a Wrap Up
- Removing a Wrap Up
- Action-Activity Type
- Creating an Action-Activity Type Pair
- Updating an Action-Activity Type Pair
- Removing an Action-Activity Type Pair

Outcome

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

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 code for outcome.

   Outcome Code cannot be changed for seeded values.

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**.

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**.

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.

List of Results

Use the Result hyperlink to maintain a list of results.

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.

   Result Code cannot be changed for seeded values.
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**.

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**.

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.

List of Reasons

Use the Reason hyperlink to maintain a list of reasons.

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**.

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**.

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.

Activity Type

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

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.

   Activity Code cannot be changed for seeded values.
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**.

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**.

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.

List of Actions

Use the Action hyperlink to maintain a list of actions.

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.

   Action Code cannot be changed for seeded values.
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**.

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**.

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.

Wrap Up

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

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**.

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**.

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.

Action-Activity Type

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

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**.

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**.

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.

5

Using Oracle Customer Interaction History

This chapter covers the following topics:

- Interactions
- Viewing Interactions (HTML)
- Searching for Interactions (HTML)
- Viewing Interactions (Self Service)
- Filtering Interactions (Self Service)
- Viewing Interactions (Forms)
- Filtering Interactions and Activities (Forms)
- Activities
- Viewing Activities (HTML)
- Searching for Activities (HTML)

Interactions

Use the Interaction History tab to search for customer interactions.

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.

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 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%.

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

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

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

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

   5. 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**.

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.

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 Viewing Interactions (Self Service)

Steps

1. To perform a simple search, do the following:
    1. From the Search list, select a type of search parameter.
    2. 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%.
    3. Click **Go**.
2. To perform an advanced search, do the following.
    1. Click the **Advanced Search** hyperlink.
    2. To display interactions that match all of the search parameters, select **Search results where each must contain all values entered**.
    3. To display interactions that match any of the search parameters, select **Search results where each may contain any value entered**.
    4. Optionally, in the Start Date field, select the match criteria and enter a start date.
    5. Optionally, in the End Date field, select the match criteria and enter an end date.
    6. Optionally, in the Resource Name field, select the match criteria and enter a resource name.
    7. Optionally, in the Interaction Type field, select the match criteria and enter an interaction type.
    8. 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.
    9. Click **Go**.

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.

Filtering Interactions and Activities (Forms)

The Customer Interaction History form lists 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 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%.

   1. Optionally, in the Agent field, select a agent name.
   2. Optionally, in the Activity Type field, select an activity type for the activities that you want to display.
   3. Optionally, in the Media Type field, select a media type for the activities that you want to display.
   4. Optionally, in the Account field, type an account for the activities that you want to display.
   5. Optionally, in the Application field, select the application that created the activities that you want to display.
   6. Optionally, in the Direction field, select the direction of the media for the activities that you want to display.

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

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

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

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

11. Click **Search**.

2. 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%.

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

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

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

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

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

   6. 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**.

   7. Click **Search**.

Activities

Use the Activities hyperlink to search for customer activities.

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:
    1. From the Media Type list, select a type of media.
    2. From the Source list, select a source.
    3. From the Activity list, select an activity type.
    4. In the Start Date field, select a start date.
    5. In the End Date field, select an end date.
3. Click **Apply**.

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%.

   1. In the Customer field, type a customer name or click **Go** to search for a customer name.
   2. In the Agent field, type a agent name or click **Go** to search for a agent name.
   3. In the Media Type field, type a media type name or click **Go** to search for a media type.
   4. In the Activity field, type an activity type name or click **Go** to search for an activity type.
   5. In the Direction field, type a direction or click **Go** to search for a direction.
   6. In the Handler field, type a handler or click **Go** to search for a handler.
   7. 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**.

6

Data Migration

This chapter covers the following topics:

- Understanding Data Migration
- Using Interaction History Migration
- Outcome-Result Administration
- Result-Reason Administration
- Creating a Result-Reason Pair
- Removing a Result-Reason Pair

Understanding Data Migration

Data migration is only necessary if you are upgrading from releases prior to Release 12. Usage of Outcome-Result and Result-Reason pairs was stopped in earlier releases and Wrap-ups was used to designate valid combinations of Outcome, Result and Reason values to drive application lists of values during interaction wrap-up.

> **Note:** Do not use the script JTFIHWRPCMP.sql, which was provided in release 11.5.9 for migrating outcome-result and result-reason pairs to wrap-ups. Instead, 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:
    1. Outcomes

2. Outcome-Result
3. Result-Reason
4. Interaction History
5. 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 to 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.

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.

Note: 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.

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.

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**.

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.

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.

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**.

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-Reason 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.

7

Concurrent Programs

This chapter covers the following topics:

- Interaction History Bulk Processor
- Interaction History Import
- Interaction History Purge

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*.

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 E-Business Suite User's Guide*.

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:

1. Writer Code
2. Request Type
3. Request Id

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.
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.

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".

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.

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 E-Business Suite User's Guide.

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.

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.

> **Caution:** 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 E-Business Suite 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.

Parameter	Required?	Definition
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.

Parameter	Required?	Definition
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.

8

API Reference

This chapter covers the following topics:

- Types of APIs
- Parameter Specifications
- Version Information
- Status Messages
- APIs
- Customer Interaction
- Package JTF_IH_PUB
- Non-cached Creation APIs
- Cached Creation APIs
- Counting APIs
- Messages and Notifications
- Sample Code

Types of APIs

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 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.

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.

Standard IN Parameters

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

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).

Parameter	Data Type	Required	Description
p_init_msg_list	VARCHAR2	Yes	The valid values for this parameter are: • True = FND_API.G_TRUE • False = FND_API.G_FALSE • Default = FND_API.G_FALSE 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". 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".

Parameter	Data Type	Required	Description
p_commit	VARCHAR2(1)	No	The valid values for this parameter are: • True = FND_API.G_TRUE • False = FND_API.G_FALSE • Default = FND_API.G_FALSE If set to true, then the API commits before returning to the calling program. To set to true, use the value, "T". If set to false, then it is the calling program's responsibility to commit the transaction. To set to false, use the value, "F".

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.

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: • FND_API.G_RET_STS_SUCCESS Success: Indicates the API call was successful • FND_API.G_RET_STS_ERROR Expected Error: There is a validation error, or missing data error. • FND_API.G_RET_STS_UNEXP_ERROR 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.

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.

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.

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.

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

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.

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.

Caution: 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.

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.

Status Message and Description

Status	Description
S	Indicates that the API performed all the operations requested by its caller. • A success return status may or may not be accompanied by messages in the API message list. • Currently, the CRM Foundation APIs do not provide a message for a return status of success.

Status	Description
E	Indicates that the API failed to perform one or more of the operations requested by its caller. 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. • For example, certain programming errors such as attempting to divide by zero causes this error. • 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.

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.

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.

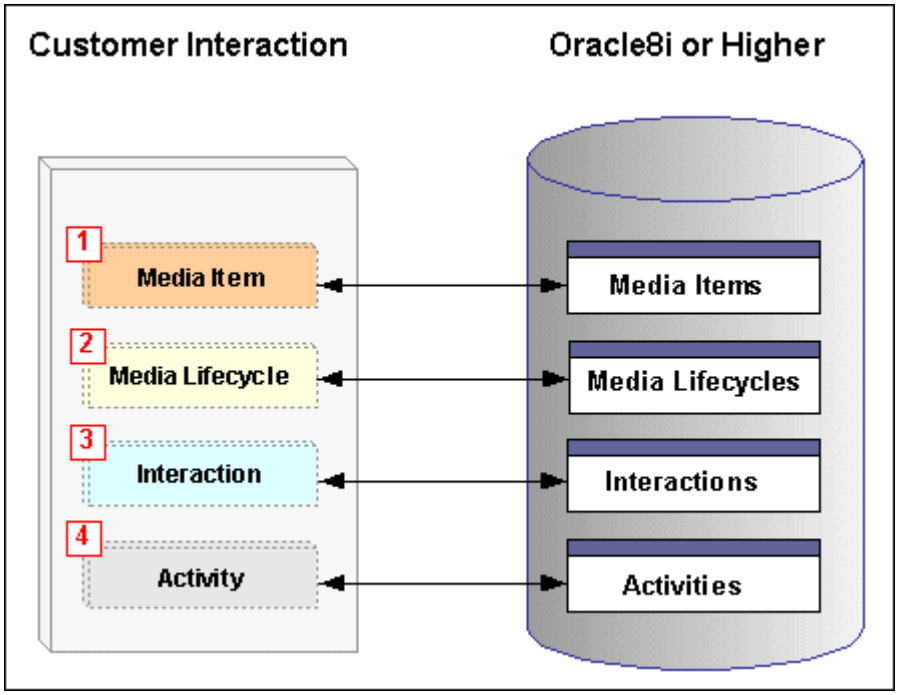
1. The customer interaction obtains its media item information from the Media Items table.

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.

Customer Interaction



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.

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.

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.

Interaction

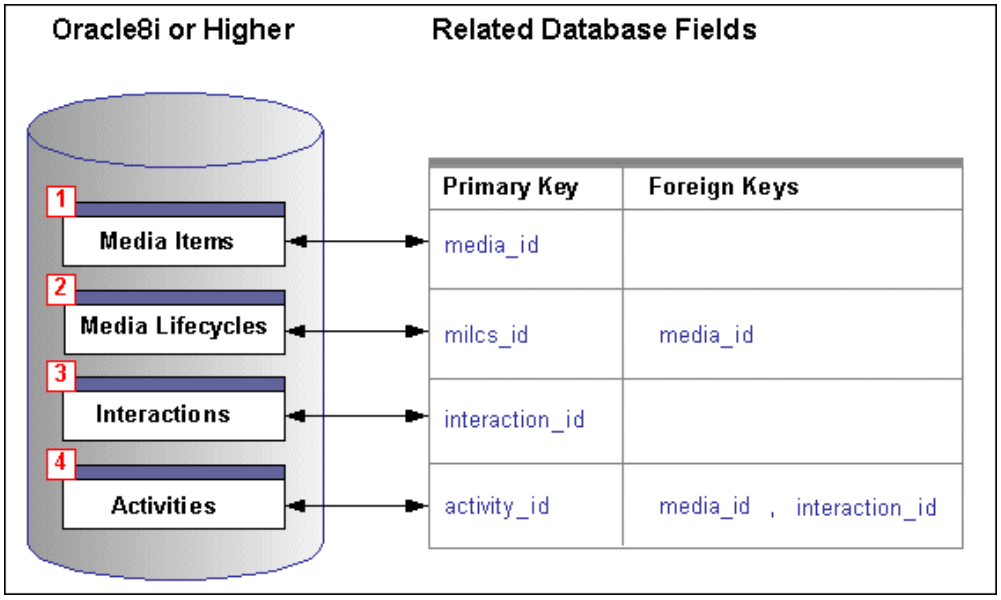
An interaction is a point of contact or touch point between a human or automated agent and a party such as a customer, a customer system, or a potential customer. An interaction is a timed entity with an outcome and a result that can be tracked. When an interaction is closed, it becomes an historical 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

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**.

Relating Customer Interaction Tables



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, and Counting APIs.

Data Structure Specifications

The Oracle Customer Interaction History APIs use the following data structures

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.

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.

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,
 );
```

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,
 );
```

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,
 );
```

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, and Create_MediaLifecycle.

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.

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.

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.

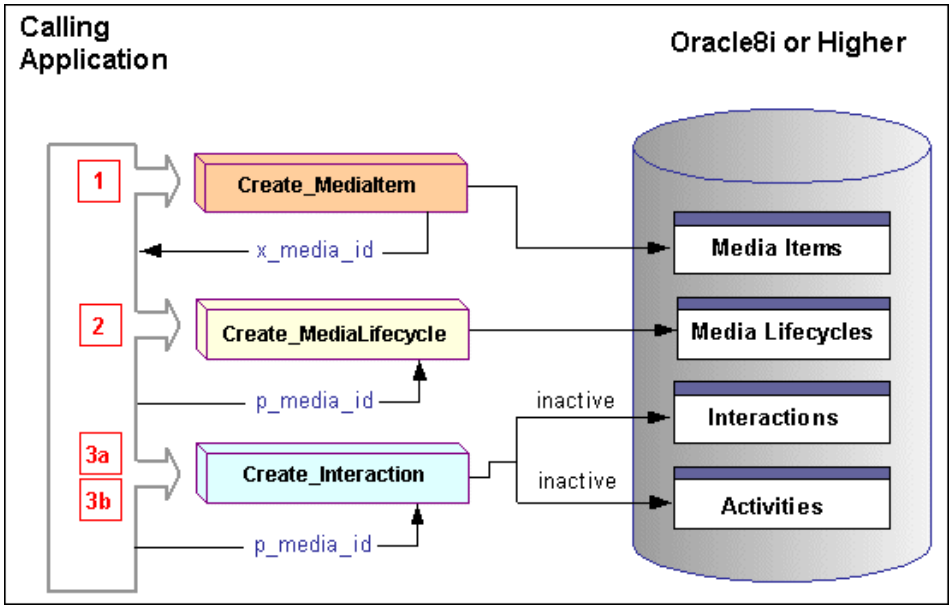
   Note: Media Item and Media Lifecycle calls are optional since interactions can exist without a media item or its associated media lifecycle record.

3. When the calling application invokes the Create_Interaction API, the following events occur:

    1. 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.

    2. 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

Non-cached creation API Process Flow



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 ass.ociate 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.

Create Media Item IN Parameters

Parameter	Data Type	Required	Descriptions and Validations
p_api_version	NUMBER	Yes	
p_init_msg_list	VARCHAR2	Yes	
p_commit	VARCHAR2	Yes	

Parameter	Data Type	Required	Descriptions and Validations
p_resp_appl_id	NUMBER	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.	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.
p_media	media_rec_type	Yes	
p_mlcs	mlcs_tbl_type	Yes	

[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.

Create Media Item OUT Parameters

Parameter	Data Type
x_return_status	VARCHAR2
x_msg_count	NUMBER
x_msg_data	VARCHAR2

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.

Create Media Lifecycle IN Parameters

Parameter	Data Type	Required	Descriptions and Validations
p_api_version	NUMBER	Yes	
p_init_msg_list	VARCHAR2	Yes	
p_commit	VARCHAR2	Yes	
p_resp_appl_id	NUMBER	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.	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.

Parameter	Data Type	Required	Descriptions and Validations
p_media_lc_rec	media_lc_rec_type	Yes	This record captures the media lifecycle. The following record parameters are always validated: • start_date_time If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place. • end_date_time

[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.

Create Media Lifecycle OUT Parameters

Parameter	Data Type
x_return_status	VARCHAR2
x_msg_count	NUMBER
x_msg_data	VARCHAR2

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	
p_init_msg_list	VARCHAR2	Yes	
p_commit	VARCHAR2	Yes	
p_resp_appl_id	NUMBER	No[1]	Application identifier
p_resp_id	NUMBER	No[*]	Responsibility identifier

Parameter	Data Type	Required	Descriptions and Validations
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 table FND_LOGINS, and identifies the login session.

Parameter	Data Type	Required	Descriptions and Validations
p_interaction_rec	INTERACTION_REC_TYPE	Yes	Contains the elements that comprise the interaction record. The following record parameters are always validated: • start_date_time If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place. • end_date_time • handler_id • outcome_id • result_id • reason_id • resource_id • party_id • source_code • source_code_id

Parameter	Data Type	Required	Descriptions and Validations
p_activities	activity_tbl_type	Yes	The following record parameters are always validated: • start_date_time If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place. • end_date_time • action_item_id • outcome_id • result_id • reason_id • action_id • source_code • source_code_id

[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.

Create Interaction OUT Parameters

Parameter	Data Type
x_return_status	VARCHAR2
x_msg_count	NUMBER
x_msg_data	VARCHAR2

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 cached creation APIs.

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.

Interaction History Cached Creation APIs

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

Procedure	Description
Open_Interaction	Creates an interaction in the Interactions table.
Update_Interaction	Updates the current interaction with values supplied by the calling application.
Create_Interaction_Activity	Creates an activity in the Activities table that is associated with the interaction passed by the calling application.
Update_Interaction_Activity	Updates the current activity with values supplied by the calling application.
Update_Interaction_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.

Process Flows

Because 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 API cannot be invoked unless the corresponding Create_Interaction_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.

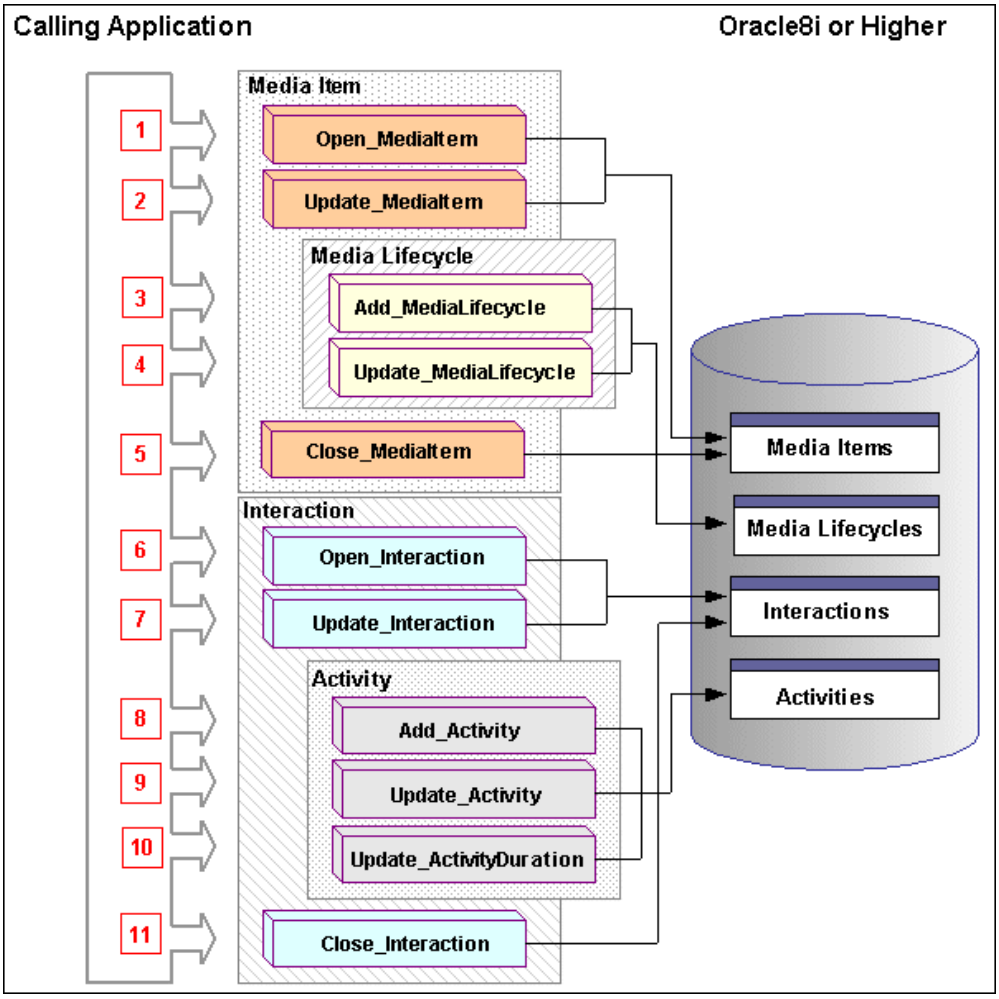
8. The calling application executes the first of three steps required to create an activity that is associated with the interaction by invoking the Create_Interaction_Activity API. This API inserts generic values in the Activities table.

9. The calling application executes the second of three steps required to create an activity that is associated with an interaction by invoking the 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 Duration 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.

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

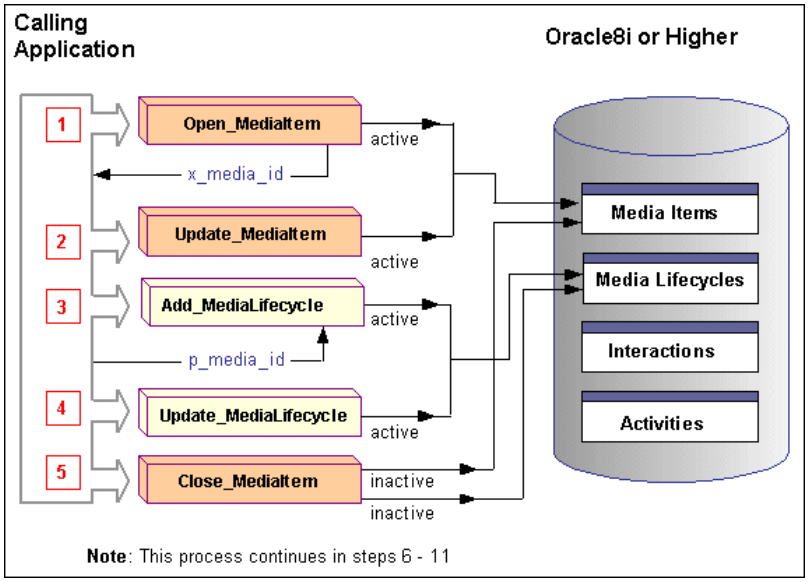
that it can be updated.

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.

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.

1. 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**.

2. 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.

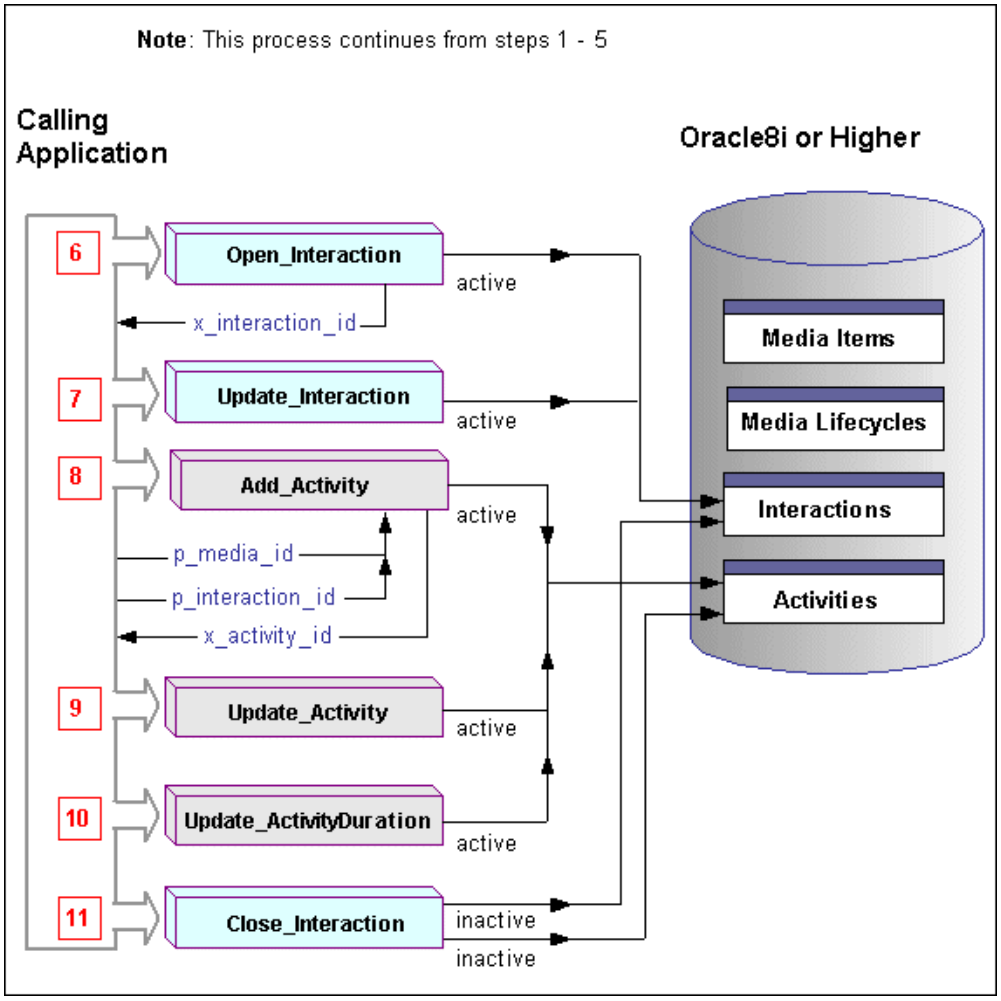
3. The calling application invokes the Create_Interaction_Activity API, inputs the media item's unique identifier as **p_media_id**, and inputs the interaction's unique identifier as **p_interaction_id**. The API sends a sequence generated identifier to the calling application as **x_activity_id**, inserts generic records in the Activities table, and sets the activity status to active so that it can be updated.

4. The API updates the Activities table with values supplied by the calling application and retains the activity's active status.

5. The Duration 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.

6. 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.

Cached Creation API Process Flow: Steps 6 -1 1



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.

Open Media Item IN Parameters

Parameter	Data Type	Required	Descriptions and Validations
p_api_version	NUMBER	Yes	
p_init_msg_list	VARCHAR2	Yes	
p_commit	VARCHAR2	Yes	
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.

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	media_rec_type		Enumerates the elements that comprise a media record. The following record parameters are always validated: • start_date_time If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place. • end_date_time

[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.

Open Media Item OUT Parameter

Parameter	Data Type	Description
x_return_status	VARCHAR2	
x_msg_count	NUMBER	
x_msg_data	VARCHAR2	

Parameter	Data Type	Description
x_media_id	NUMBER	The record number for the created media item. It is automatically generated by sequence JTF_IH_MEDIA_ITEMS_S1.

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
  );

PROCEDURE

);
```

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.

Update Media Item IN Parameters

Parameter	Data Type	Required	Descriptions and Validations
p_api_version	NUMBER	Yes	
p_init_msg_list	VARCHAR2	Yes	
p_commit	VARCHAR2	Yes	
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 table FND_LOGINS, and identifies the login session.
p_media_rec	media_rec_type		Enumerates the elements that comprise a media record. The following record parameters are always validated: • start_date_time If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place. • end_date_time

[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.

Update Media Item OUT Parameters

Parameter	Data Type
x_return_status	VARCHAR2
x_msg_count	NUMBER
x_msg_data	VARCHAR2

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.

Add Media Lifecycle IN Parameters

Parameter	Data Type	Required	Descriptions and Validations
p_api_version	NUMBER	Yes	
p_init_msg_list	VARCHAR2	Yes	
p_commit	VARCHAR2	Yes	
p_resp_appl_id	NUMBER	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.	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.

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. The following record parameters are always validated: • start_date_time If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place. • end_date_time

[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.

Add Media Lifecycle OUT Parameter

Parameter	Data Type	Description
x_return_status	VARCHAR2	
x_msg_count	NUMBER	
x_msg_data	VARCHAR2	
x_milcs_id	NUMBER	Corresponds to the sequence generated media lifecycle identifier for the record created.

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.

Update Media Lifecycle IN Parameters

Parameter	Data Type	Required	Descriptions and Validations
p_api_version	NUMBER	Yes	
p_init_msg_list	VARCHAR2	Yes	
p_commit	VARCHAR2	Yes	

Parameter	Data Type	Required	Descriptions and Validations
p_resp_appl_id	NUMBER	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.	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.

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. The following record parameters are always validated: • start_date_time If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place. • end_date_time

[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 table describes the OUT parameters associated with this API.

Update Media Lifecycle OUT Parameters

Parameter	Data Type
x_return_status	VARCHAR2
x_msg_count	NUMBER
x_msg_data	VARCHAR2

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.

Close Media Item IN Parameters

Parameter	Data Type	Required	Descriptions and Validations
p_api_version	NUMBER	Yes	
p_init_msg_list	VARCHAR2	Yes	
p_commit	VARCHAR2	Yes	
p_resp_appl_id	NUMBER	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.	Application identifier
p_resp_id	NUMBER	No*	Responsibility identifier

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_rec	media_rec_type	Yes	Enumerates the elements that comprise a media record. The following record parameters are always validated: • start_date_time If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place. • end_date_time 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.

[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.

Close Media Item OUT Parameters

Parameter	Data Type
x_return_status	VARCHAR2
x_msg_count	NUMBER
x_msg_data	VARCHAR2

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_media_rec          in      media_rec_type
  );
```

Current Version

```
1.0
```

Parameter Descriptions

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

Open Interaction IN Parameters

Parameter	Data Type	Required	Descriptions and Validations
p_api_version	NUMBER	Yes	

Parameter	Data Type	Required	Descriptions and Validations
p_init_msg_list	VARCHAR2	Yes	
p_commit	VARCHAR2	Yes	
p_resp_appl_id	NUMBER	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.	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.

Parameter	Data Type	Required	Descriptions and Validations
p_interaction_rec	interaction_rec_type	Yes	Contains the elements that comprise the interaction record. The following record parameters are always validated: • start_date_time If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place. • end_date_time • handler_id • outcome_id • result_id • reason_id • resource_id • party_id • source_code • source_code_id

[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.

Open Interaction OUT Parameter

Parameter	Data Type	Description
x_return_status	VARCHAR2	
x_msg_count	NUMBER	
x_msg_data	VARCHAR2	
x_interaction_id	NUMBER	Corresponds to a sequence generated reference for the interaction record.

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 multiparty validation.

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.

Update Interaction IN Parameters

Parameter	Date Type	Required	Descriptions and Validations
p_api_version	NUMBER	Yes	
p_init_msg_list	VARCHAR2	Yes	
p_commit	VARCHAR2	Yes	
p_resp_appl_id	NUMBER	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.	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.

Parameter	Date Type	Required	Descriptions and Validations
p_interaction_rec	interaction_rec_type	Yes	Used in updating the interaction record. The following record parameters are always validated: • start_date_time If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place. • end_date_time • handler_id • outcome_id • result_id • reason_id • resource_id • party_id • source_code • source_code_id

[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.

Parameter	Data Type
x_return_status	VARCHAR2
x_msg_count	NUMBER

Parameter	Data Type
x_msg_data	VARCHAR2

Create_Interaction_Activity

The Create_Interaction_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 Create_Interaction_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.

Create_Interaction_Activity IN Parameters

Parameter	Data Type	Required	Descriptions and Validations
p_api_version	NUMBER	Yes	
p_init_msg_list	VARCHAR2	Yes	
p_commit	VARCHAR2	Yes	

Parameter	Data Type	Required	Descriptions and Validations
p_resp_appl_id	NUMBER	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.	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.

Parameter	Data Type	Required	Descriptions and Validations
p_activity_rec	activity_rec_type	Yes	Used in updating the interaction record. The following record parameters are always validated: • start_date_time If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place. • end_date_time • action_item_id • outcome_id • result_id • reason_id • action_id • source_code • source_code_id

[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.

Create_Interaction_Activity OUT Parameters

Parameter	Data Type	Description
x_return_status	VARCHAR2	
x_msg_count	NUMBER	

Parameter	Data Type	Description
x_msg_data	VARCHAR2	
x_activity_id	NUMBER	Corresponds to the sequence generated activity identifier for the record created.

Update_Interaction_Activity

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

Procedure Specification

```
PROCEDURE Update_Interaction_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_Interaction_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.

Update_Interaction_Activity IN Parameters

Parameter	Data Type	Required	Descriptions and Validations
p_api_version	NUMBER	Yes	

Parameter	Data Type	Required	Descriptions and Validations
p_init_msg_list	VARCHAR2	Yes	
p_commit	VARCHAR2	Yes	
p_resp_appl_id	NUMBER	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.	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.

Parameter	Data Type	Required	Descriptions and Validations
p_activity_rec	activity_rec_type	Yes	Used in updating the interaction record. The following record parameters are always validated: • start_date_time If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place. • end_date_time • action_item_id • outcome_id • result_id • reason_id • action_id • source_code • source_code_id

[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.

Update_Interaction_Activity OUT Parameters

Parameter	Data Type
x_return_status	VARCHAR2
x_msg_count	NUMBER

Parameter	Data Type
x_msg_data	VARCHAR2

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.

Update Activity Duration IN Parameters

Parameter	Data Type	Required	Descriptions and Validations
p_api_version	NUMBER	Yes	
p_init_msg_list	VARCHAR2	Yes	

Parameter	Data Type	Required	Descriptions and Validations
p_commit	VARCHAR2	Yes	
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.

Update Activity Duration OUT Parameters

Parameter	Data Type
x_return_status	VARCHAR2
x_msg_count	NUMBER
x_msg_data	VARCHAR2

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.

Parameter Descriptions

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

Close Interaction IN Parameters

Parameter	Data Type	Required	Descriptions and Validations
p_api_version	NUMBER	Yes	
p_init_msg_list	VARCHAR2	Yes	
p_commit	VARCHAR2	Yes	
p_resp_appl_id	NUMBER	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.	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.

Parameter	Data Type	Required	Descriptions and Validations
p_interaction_rec	interaction_rec_type	Yes	Contains the elements that comprise the interaction record. The following record parameters are always validated: • start_date_time If the *start_date_time* parameter is not supplied, then *sysdate* is inserted in its place. • end_date_time 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. • handler_id • outcome_id • result_id • reason_id • resource_id • party_id • source_code • source_code_id

[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.

Close Interaction OUT Parameters

Parameter	Data Type
x_return_status	VARCHAR2
x_msg_count	NUMBER
x_msg_data	VARCHAR2

Counting APIs

The following are counting APIs:

- Get_InteractionCount

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.

Counting APIs

Procedure	Description
Get_InteractionCount	Retrieves the interaction count from Interaction table.

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.

Get Interaction Count IN Parameters

Parameter	Data Type	Required	Descriptions and Validations
p_api_version	NUMBER	Yes	
p_init_msg_list	VARCHAR2	Yes	

Parameter	Data Type	Required	Descriptions and Validations
p_resp_appl_id	NUMBER	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.	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_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.

Parameter	Data Type	Required	Descriptions and Validations
p_attribute1	VARCHAR2(150)	No[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.	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.

Parameter	Data Type	Required	Descriptions and Validations
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.

Get Interaction Count OUT Parameter

Parameter	Data Type	Description
x_return_status	VARCHAR2	
x_msg_count	NUMBER	
x_msg_data	VARCHAR2	
x_interaction_count	NUMBER	Corresponds to the number of interactions found.

Messages and Notifications

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

JTF_IH_PUB

Create_Interaction

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

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.

Type	Name	Text
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.

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.

Type	Name	Text
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.

Open_MediaItem

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

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.

Update_MediaItem

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

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.

Add_MediaLifecycle

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

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.

Type	Name	Text
E	JTF_API_ALL_INVALID_ARGUMENT	API Programming Error (JTF_IH_PUB. Add_Medialifecycle): The value <parameter value> for milcs_type_id is invalid.

Close_MediaItem

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

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.

Open_Interaction

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

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.

Type	Name	Text
E	JTF_API_ALL_INVALID_ARGUMENT	API Programming Error (JTF_IH_PUB.Open_Interaction): The value <parameter value> for reason_id is invalid.
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.

Type	Name	Text
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.

Update_Interaction

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

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.

Type	Name	Text
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.

Type	Name	Text
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.
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.

Type	Name	Text
E	JTF_API_ALL_INVALID_ARGUMENT	API Programming Error (JTF_IH_PUB.Update_Interaction): The value <parameter value> for interaction is not active.

Add_Activity

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

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.

Type	Name	Text
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.
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.

Update_Interaction_Activity

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

Update Activity Messages

Type	Name	Text
E	JTF_API_ALL_INVALID_ARGUMENT	API Programming Error (JTF_IH_PUB.Update_Interaction_Activity): The value <parameter value> for outcome_id is invalid.
E	JTF_API_ALL_INVALID_ARGUMENT	API Programming Error (JTF_IH_PUB.Update_Interaction_Activity): The value <parameter value> for result_id is invalid.
E	JTF_API_ALL_INVALID_ARGUMENT	API Programming Error (JTF_IH_PUB.Update_Interaction_Activity): The value <parameter value> for reason_id is invalid.
E	JTF_API_ALL_INVALID_ARGUMENT	API Programming Error (JTF_IH_PUB.Update_Interaction_Activity): The value <parameter value> for action_item_id is invalid.
E	JTF_API_ALL_INVALID_ARGUMENT	API Programming Error (JTF_IH_PUB.Update_Interaction_Activity): The value <parameter value> for action_id is invalid.
E	JTF_API_ALL_INVALID_ARGUMENT	API Programming Error (JTF_IH_PUB.Update_Interaction_Activity): The value <parameter value> for source_code_id set and source_code not set is invalid.

Type	Name	Text
E	JTF_API_ALL_INVALID_ARGUMENT	API Programming Error (JTF_IH_PUB.Update_Interaction_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_Interaction_Activity): The value <parameter value> for end_date_time is invalid.
E	JTF_API_ALL_INVALID_ARGUMENT	API Programming Error (JTF_IH_PUB.Update_Interaction_Activity): The value <parameter value> for activity_id is invalid.
E	JTF_API_ALL_INVALID_ARGUMENT	API Programming Error (JTF_IH_PUB.Update_Interaction_Activity): The value <parameter value> for cust_account_id is invalid.
E	JTF_API_ALL_INVALID_ARGUMENT	API Programming Error (JTF_IH_PUB.Update_Interaction_Activity): The value <parameter value> for active is not active.

Close_Interaction

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

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.

Type	Name	Text
E	JTF_API_ALL_INVALID_ARGUMENT	API Programming Error (JTF_IH_PUB.Close_Interaction): The value <parameter value> for reason_id is invalid.
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.

Type	Name	Text
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.

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

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.

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);
```

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);
```

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 ');
```

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.

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;
```

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);
```

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);
```

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);
```

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;
/
```

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);
```

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);
        --
```

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(' ');
```

Update_Interaction_Activity

This script calls the Update_Interaction_Activity API and provides the following values using the Update_Interaction_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_id1;
            l_activity_rec.description := 'Test Update Activity';
```

```
JTF_IH_PUB.Update_Interaction_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;
```

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',
                                        -1,
                                        690,
                                        2877,
                                        -1,
                                        l_return_status,
                                        l_msg_count,
                                        l_msg_data,
                                        l_activity_rec,
                                        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 ');
```

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;
/
```

Counting APIs

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));
```

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));
```

9

Data Validations

This chapter covers the following topics:

- Interaction Validations and Defaults
- Activity Validations and Defaults
- Media Item Validations and Defaults
- Media Item Life-cycle Segment Validations and Defaults

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 that are not exposed by way of 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 by way of 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.

Interaction Record Value	Table column mapping in JTF_IH_INTERACTIONS and JTF_IH_INTERACTIONS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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.

Interaction Record Value	Table column mapping in JTF_IH_INTERACTIONS and JTF_IH_INTERACTIONS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
Contact_party_id	CONTACT_PARTY_ID	ID of the person party that is the contact 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.
resource_id	RESOURCE_ID	ID of the agent/user of the person who performs the interaction with the customer. FK to JTF_RS_RESOURCE_EXTN.	N			Valid RESOURCE_ID in JTF_RS_RESOURCE_EXTN

Interaction Record Value	Table column mapping in JTF_IH_INTERACTIONS and JTF_IH_INTERACTIONS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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 the 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

Interaction Record Value	Table column mapping in JTF_IH_INTERACTIONS and JTF_IH_INTERACTIONS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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, and so on.				Valid ARC_SOURCE_CODE_FOR in AMS_SOURCE_CODES. Marketing SOURCE_TYPE.
object_id	OBJECT_ID	The ID of the Campaign, Event, and so on. 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.

Interaction Record Value	Table column mapping in JTF_IH_INTERACTIONS and JTF_IH_INTERACTIONS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
start_date_time	START_DATE_TIME	The date and time, to the second, that the interaction started.			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, to the second, that the interaction ended.			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.

Interaction Record Value	Table column mapping in JTF_IH_INTERACTIONS and JTF_IH_INTERACTIONS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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.				
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.				

Interaction Record Value	Table column mapping in JTF_IH_INTERACTIONS and JTF_IH_INTERACTIONS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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				

Interaction Record Value	Table column mapping in JTF_IH_INTERACTIONS and JTF_IH_INTERACTIONS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
attribute12	ATTRIBUTE12	Descriptive Flexfield Segment				
attribute13	ATTRIBUTE13	Descriptive Flexfield Segment				
attribute14	ATTRIBUTE14	Descriptive Flexfield Segment				
attribute15	ATTRIBUTE15	Descriptive Flexfield Segment				
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.

Interaction Record Value	Table column mapping in JTF_IH_INTERACTIONS and JTF_IH_INTERACTIONS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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_EMPLOYEE'	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.REFERENCE_FORM

Interaction Record Value	Table column mapping in JTF_IH_INTERACTIONS and JTF_IH_INTERACTIONS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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
	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		

Interaction Record Value	Table column mapping in JTF_IH_INTERACTIONS and JTF_IH_INTERACTIONS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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				

Interaction Record Value	Table column mapping in JTF_IH_INTERACTIONS and JTF_IH_INTERACTIONS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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

Interaction Record Value	Table column mapping in JTF_IH_INTERACTIONS and JTF_IH_INTERACTIONS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
	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.
	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.

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.
interaction_id	INTERACTION_ID	Unique Interaction Identifier. FK to JTF_IH_INTERACTIONS.INTERACTION_ID	Y			Valid INTERACTION_ID in JTF_IH_INTERACTIONS.

Activity Record Value	Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ACTIVITIES_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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

Activity Record Value	Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ACTIVITIES_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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
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 ORGANIZATION Party_ID of the customer's business.				Not validated in API. Should be a valid Party_id in HZ_PARTIES where PARTY_TYPE = 'ORGANIZATION'

Activity Record Value	Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ACTIVITIES_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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

Activity Record Value	Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ACTIVITIES_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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
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.

Activity Record Value	Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ACTIVITIES_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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.

Activity Record Value	Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ACTIVITIES_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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.

Activity Record Value	Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ACTIVITIES_STG	Description	Req.	Obsolete	Default	Validation/ Notes
interaction_action_type	INTERACTION_ACTION_TYPE	TBD				Need to determine current use/need for this column.
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				

Activity Record Value	Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ACTIVITIES_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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				

Activity Record Value	Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ACTIVITIES_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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

Activity Record Value	Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ACTIVITIES_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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

Activity Record Value	Table column mapping in JTF_IH_ACTIVITIES and JTF_IH_ACTIVITIES_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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.

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_MediaI Item 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

Media Item Record Value	Table column mapping in JTF_IH_MEDIA_ITEMS and JTF_IH_MEDIA_ITEMS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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.

Media Item Record Value	Table column mapping in JTF_IH_MEDIA_ITEMS and JTF_IH_MEDIA_ITEMS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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.
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 Item Record Value	Table column mapping in JTF_IH_MEDIA_ITEMS and JTF_IH_MEDIA_ITEMS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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.				

Media Item Record Value	Table column mapping in JTF_IH_MEDIA_ITEMS and JTF_IH_MEDIA_ITEMS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
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. For Phone Calls Inbound = Number dialed from (ANI) Outbound = Number dialed (DNIS) For Email Inbound = email address sent from Outbound = email address sent to For Fax Outbound = dialed fax number For Printer Outbound = printer address				

Media Item Record Value	Table column mapping in JTF_IH_MEDIA_ITEMS and JTF_IH_MEDIA_ITEMS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
	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_Media Item 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. The 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

Media Item Record Value	Table column mapping in JTF_IH_MEDIA_ITEMS and JTF_IH_MEDIA_ITEMS_STG	Description	Req.	Obsolete	Default	Validation/ Notes
	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.

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 by way of 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

Media Item LC Seg Record Value	Table column mapping in JTF_IH_MEDIA_ITEM_LC_SEGS	Description	Req.	Obsolete	Default	Validation/Notes
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_Media Lifecycle calls. If a value is passed it is used in place of SYSDATE. Must be less then or equal to END_DATE_TIME.

Media Item LC Seg Record Value	Table column mapping in JTF_IH_MEDIA_ITEM_LC_SEGS	Description	Req.	Obsolete	Default	Validation/Notes
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_Media Lifecycle 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.
type_type	TYPE_TYPE	Identifies the system or application responsible for handling the Lifecycle segment.				

Media Item LC Seg Record Value	Table column mapping in JTF_IH_MEDIA_ITEM_LC_SEGS	Description	Req.	Obsolete	Default	Validation/Notes
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

Media Item LC Seg Record Value	Table column mapping in JTF_IH_MEDIA_ITEM_LC_SEGS	Description	Req.	Obsolete	Default	Validation/Notes
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
	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_Media Lifecycle is used, the Media Item is created as non-active, N.
	OBJECT_VERSION_NUMBER	Standard Object Version Field	Y			Set to 1.0

Media Item LC Seg Record Value	Table column mapping in JTF_IH_MEDIA_ITEM_LC_SEGS	Description	Req.	Obsolete	Default	Validation/Notes
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

Media Item LC Seg Record Value	Table column mapping in JTF_IH_MEDIA_ITEM_LC_SEGS	Description	Req.	Obsolete	Default	Validation/Notes
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.

Index

J

O

P

S

U