Oracle® Common Application Calendar

API Reference Guide

Release 12

Part No. B28855-01

December 2006

ORACLE

Oracle Common Application Calendar API Reference Guide, Release 12

Part No. B28855-01

Primary Author: Judy Wood

Contents

Send Us Your Comments

Preface

1 Introduction

Oracle Common Application Calendar APIs 1-1
Private APIs 1-1
Public APIs 1-2
Public, published APIs 1-2
Parameter Specifications 1-3
Standard IN Parameters 1-3
Standard OUT Parameters 1-6
Parameter Size 1-7
Missing Parameter Attributes 1-7
Parameter Validations 1-8
Invalid Parameters 1-8
Version Information 1-8
Status Messages 1-9

2 Task Manager Public APIs

Tasks Overview 2-1
Task Manager Public Packages 2-2
Package JTF_TASKS_PUB 2-5
Data Structure Specifications 2-5
Package JTF_TASK_ASSIGNMENTS_PUB 2-19
Data Structure Specifications 2-19

Package JTF_TASK_REFERENCES_PUB 2-23
Data Structure Specifications 2-24
Messages and Notifications 2-26
Common Messages 2-27
JTF_TASKS_PUB 2-32
JTF_TASK_ASSIGNMENTS_PUB 2-34
JTF_TASK_REFERENCES_PUB 2-35
Sample Code 2-36
Package JTF_TASKS_PUB 2-36
Package JTF_TASK_ASSIGNMENTS_PUB 2-40
Package JTF_TASK_REFERENCES_PUB 2-44

3 Notes Public APIs

Notes Overview 3-1
Package JTF_NOTES_PUB 3-1
Data Structure Specifications for JTF_NOTES_PUB 3-2
Notes Public APIs 3-3
Note Source and Note Context 3-3
Party Relationships 3-5
Secure_Create_Note 3-6
Secure_Update_Note 3-8
Secure_Delete_Note 3-8
Messages and Notifications 3-8
Sample Code 3-10
Package JTF_NOTES_PUB 3-10

Index

Send Us Your Comments

Oracle Common Application Calendar API Reference Guide, Release 12

Part No. B28855-01

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 Applications Release Online Documentation CD available on Oracle MetaLink 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 of the *Oracle Common Application Calendar API Reference Guide.*

This manual describes the Oracle Common Application Calendar's public APIs, and provides information to help you work effectively with these public APIs.

This document provides API information for the following Oracle Common Application Calendar components only:

- Task Manager
- Notes

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

- The principles and customary practices of your business area.
- The Oracle Common Application Calendar application
- Oracle Application Framework Applications
- Oracle Forms Applications

  To learn more about Oracle Forms Applications, read the *Oracle Applications User Interface Standards for Forms-Based Products* guide.

- Oracle Self-Service Web Applications
- The Oracle Applications graphical user interface

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

See Related Information Sources on page viii for more Oracle Applications product information.

TTY Access to Oracle Support Services

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

Documentation Accessibility

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Structure

1 Introduction
2 Task Manager Public APIs
3 Notes Public APIs

Related Information Sources

Oracle Common Application Calendar User Guide

The User Guide contains important reference and background information on each of the Oracle Common Application Calendar modules. In addition, it contains procedures and using information that describe the common user and tasks that are necessary to perform in each of the modules.

Oracle Common Application Calendar Implementation Guide

The Implementation Guide contains important reference and background information on each of the Oracle Common Application Calendar. In addition, it contains procedures and implementing and System Administration tasks that are necessary to perform in each of the modules.

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.

Do Not Use Database Tools to Modify Oracle Applications Data

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

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

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

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

1

Introduction

This chapter covers the following topics:

- Oracle Common Application Calendar APIs
- Private APIs
- Public APIs
- Public, published APIs
- Parameter Specifications
- Standard IN Parameters
- Standard OUT Parameters
- Parameter Size
- Missing Parameter Attributes
- Parameter Validations
- Invalid Parameters
- Version Information
- Status Messages

Oracle Common Application Calendar APIs

The Oracle Common Application Calendar contains the following types of APIs:

Private 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 e-Business Suite development environment.

Public APIs

Public APIs are designed for customers and Oracle consultants to integrate non-Oracle systems into Oracle e-Business Suite 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

Public Published APIs are guaranteed by Oracle to remain valid from release to release and that patches will not alter the API behavior. Public, published APIs are supported by Oracle to the same extent as released software.

For non-published APIs, Oracle expressly does not provide any guarantees regarding consistency of naming, usage, or behavior of any API (public or private) between releases. It is also possible that a patch could alter any characteristic of any non-published e-Business Suite 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.

> **Note:** In Applications release 11.5.9, many of the Applications' PL/SQL server side APIs have been enhanced to utilize the pass by reference semantics of PL/SQL. This improves performance considerably and reduces memory consumption. In the normal processing case (i.e. success), there is no change of behavior, and callers of these APIs are not impacted. However, in the case of exceptions, there is a behavior change which results in assignments being exposed to the caller, which are made in the API prior to any exceptions being raised. The previous behavior would rollback these assignments made by the API if an exception occurred in this API. Developers writing custom extensions to Oracle Applications, or third party integrators which use the standard Applications' APIs should be aware of this change in semantics.

The Common Application Calendar APIs

The public APIs provided by the Oracle Common Application Calendar and described in this document are divided into groups of public packages. There are one or more packages for each of the following Oracle Common Application Calendar modules covered here:

- Task Manager

- Notes

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 the Oracle Common Application Calendar 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 Common Application Calendar.

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 Common Application Calendar.

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.

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

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

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

Status Messages

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)

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

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 Oracle Common Application Calendar APIs do not provide a message for a return status of success.
E	Indicates that the API failed to perform one or more of the operations requested by its caller. 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.

2

Task Manager Public APIs

This chapter covers the following topics:

- Tasks Overview
- Task Manager Public Packages
- Package JTF_TASKS_PUB
- Data Structure Specifications
- Package JTF_TASK_ASSIGNMENTS_PUB
- Data Structure Specifications
- Package JTF_TASK_REFERENCES_PUB
- Data Structure Specifications
- Messages and Notifications
- Common Messages
- JTF_TASKS_PUB
- JTF_TASK_ASSIGNMENTS_PUB
- JTF_TASK_REFERENCES_PUB
- Sample Code
- Package JTF_TASKS_PUB
- Package JTF_TASK_ASSIGNMENTS_PUB
- Package JTF_TASK_REFERENCES_PUB

Tasks Overview

Oracle Task Manager provides methods for quantifying, responding to, and managing units of work generated by E-Business Suite applications. The unit of work is defined in Task Manager in the form of a task. A given task may be defined and bound by units of

time, may be assigned to one or more qualified resource, resource groups or teams, and may be tracked by the application that the work was generated from.

Task Manager Public Packages

All public procedures (APIs) relating to creating, updating, or deleting tasks, task references, or task assignments are stored in the following public packages:

- JTF_TASKS_PUB
- JTF_TASK_ASSIGNMENTS_PUB
- JTF_TASK_REFERENCES_PUB

The following tables describe the Task Manager public APIs.

Tasks APIs

Procedure	Description
Create_Task	Creates a task and provides the task with values supplied by the calling application. It also creates the dependencies, resource requirements, assignments, references, dates, recurrences, and contacts.
Update_Task	Updates a task with values supplied by the calling application.
Delete_Task	Soft deletes a task by marking the task as deleted without removing the task record from the database table in which it is stored.

Task Assignments APIs

Procedure	Description
Create_Task_Assignment	Creates a task assignment for a specified task. The level of effort can be specified for each resource assigned to the task.

Procedure	Description
Update_Task_Assignment	Updates a task assignment for a given task. For each resource assigned to the task, the actual effort can be specified.
Delete_Task_Assignment	Deletes a task assignment for a specified task.

Task References APIs

Procedure	Description
Create_References	Creates references to a task. For example, a task created for a service request can establish reference to an inventory part number using references.
Update_References	Updates references to a task.
Delete_References	Deletes references to a task.

Note:

- The following are referenced throughout the table's "Validations and Descriptions" column in all of the Task APIs.

  1. The Task APIs support calls by values not internal IDs. For example, the Account Number can be passed to the API instead of the Account ID. However, using these IDs improves the API performance.

  2. The customer account should belong to the same customer which is specified for the task.

  3. The address (*party_site_id* and *party_site_numbe*r) specified for the task, should be the address registered for the customer in the TCA schema. The address can only be specified if the customer for the task is specified. In addition, the customer address specified for the task must be specified for the same customer at the task level.

  4. To specifiy the unit of measure (UOM) for the effort and duration fields, the profile *jtf_time_uom_class* must be populated with the unit of measure class. For example, the customer can create a unit of measure class called TIME. The

profile must be populated with this value. Further, the customer should create unit of measure codes such as minutes, hours, and days for the Time class. While specifying values to the APIs, the customer can pass values for these unit of measure codes.

5. Child and Parent tasks must have the same source. For example, the child task cannot have the source as Lead and the Parent task cannot have the source as Service Request.

6. Each task must have an owner associated with it. The owner for a task can be populated with the following two fields:

   - *Owner_type_code* (for example *rs_employee)*
   - *Owner_id* (for example *resource_id* of the employee)

     The *owner_type_code* is a Foreign Key to *jtf_objects_vl* for objects with a usage of "Resources." The object definition in *jtf_objects* contains all the relevant information to create a PL/SQL statement. Using this PL/SQL statement, the *owner_id* is validated.

     For example, for *rs_employee,* the SELECT statement that is created at run time is:

     ```
     SELECT resource_id, full_name from jtf_rs_emp_dtls_vl
     ```

     Owner IDs passed to this API are validated against resources in the system.

7. Every task is associated with a source. Some examples of sources are: Service Requests, Leads, Opportunity, Campaigns, Defects, and Contracts. The source of a task drives most of the business rules for the task.

   For example, Service Request #100 is the source for Task #101, Task #105, and Task #107. Therefore, the parent entity, Service Request #100, can create multiple tasks with the same source.

   To create a source, the entity must be defined in jtf_objects. This can be accomplished by logging into **CRM Administrator Responsibility >Task and Escalation Manager > Setups > Objects Meta > Data.**

   Continuing with the previous example, you will have the values of the parameters to create a task with a source of Service Request # 100.

   p_source_object_type_code = SR

   p_source_object_id = 100

   p_source_object_name = A101

   The *Source_object_type_code* is a Foreign Key to the jtf_objects with a usage of Task.

The *source_object_id* is the internal identifier which represents the instance of the source, in this case, service request identifier. This is generally the Primary Key of the "Source table," in this case, service request table.

The *p_source_object_name* is the external identifier which represents the instance of the source, in this case, service request number. The source object name is displayed on the user interface.

The same concept is used to established references to a task.

8. While updating or deleting any of the three public packages: tasks, task assignments, or task references, the *object_version_number* must be passed to the APIs. If the *object_version_number* that is passed into the API is the same as in the tables (tasks, assignments, and references), then the attempt is made to update or delete the record. Otherwise, an error is returned. You should get the *object_version_number* when selecting the record, then pass it to the update and delete APIs.

Package JTF_TASKS_PUB

The JTF_TASKS_PUB API contains the following APIs. Information is available from the Integration Repository.

- Create_Task
- Update_Task
- Delete_Task

Data Structure Specifications

The following data structures are used in the JTF_TASKS_PUB API:

- Task Assign Record Type, page 2-6
- Task Dependency Record Type, page 2-10
- Task Reference Record Type, page 2-11
- Task Recurrence Record Type, page 2-12
- Task Dates Record Type, page 2-14
- Task Contact Record Type, page 2-15
- Task Notes Record Type, page 2-15

- Sort Record Type, page 2-18

Task Assign Record Type

This composite record type enumerates task assignment details that are ultimately stored in the JTF_TASK_ASSIGNMENTS table.

```
TYPE task_assign_rec IS RECORD (
        resource_type_code
jtf_task_assignments.resource_type_code%type ,
        resource_id
jtf_task_assignments.resource_id%type  ,
        actual_start_date             DATE := NULL,
        actual_end_date               DATE := NULL,
        actual_effort
jtf_task_assignments.actual_effort%type  := NULL,
        actual_effort_uom
jtf_task_assignments.actual_effort_uom%type  := NULL,
        SCHED_TRAVEL_DISTANCE
jtf_task_assignments.SCHED_TRAVEL_DISTANCE%type  := NULL,
        SCHED_TRAVEL_Duration
jtf_task_assignments.SCHED_TRAVEL_Duration%type  := NULL,
        sched_travel_duration_uom
jtf_task_assignments.sched_travel_duration_uom%type  := NULL,
        actual_travel_distance
jtf_task_assignments.actual_travel_distance%type  := NULL,
        actual_travel_duration
jtf_task_assignments.actual_travel_duration%type  := NULL,
        actual_travel_duration_uom
jtf_task_assignments.actual_travel_duration_uom%type  := NULL,
        schedule_flag
jtf_task_assignments.schedule_flag%type  := NULL,
        alarm_type_code
jtf_task_assignments.alarm_type_code%type  := NULL,
        alarm_contact
jtf_task_assignments.alarm_contact%type  := NULL,
        palm_flag
jtf_task_assignments.palm_flag%type  := NULL,
        wince_flag
jtf_task_assignments.wince_flag%type  := NULL,
        laptop_flag
jtf_task_assignments.laptop_flag%type  := NULL,
        device1_flag
jtf_task_assignments.device1_flag%type  := NULL,
        device2_flag
jtf_task_assignments.device2_flag%type  := NULL,
        device3_flag
jtf_task_assignments.device3_flag%type  := NULL,
        resource_territory_id
jtf_task_assignments.resource_territory_id%type  := NULL,
        assignment_status_id
jtf_task_assignments.assignment_status_id%type  := NULL,
        shift_construct_id
jtf_task_assignments.shift_construct_id%type  := NULL,
        show_on_calendar
jtf_task_assignments.show_on_calendar%type := NULL,
        category_id
jtf_task_assignments.category_id%type := NULL
    );
```

The following table describes the fields associated with this record type.

task_assign_rec Fields

Field Name	Data Type	Required	Descriptions and Validations
resource_type_code	VARCHAR2(30)		Resource type code of the assignee
resource_id	NUMBER		Resource identifier of the assignee.
actual_start_date	DATE		Actual start date of the assignment.
actual_end_date	DATE		Actual end date of the assignment.
actual_effort	NUMBER		Actual effort duration for the assignment.
actual_effort_uom	VARCHAR2 (3)		UOM of actual effort duration.
sched_travel_distance	NUMBER		Scheduled distance to be traveled by the assignee.
sched_travel_duration	NUMBER		Scheduled duration of travel.
sched_travel_duration_uom	VARCHAR2 (3)		UOM of scheduled travel duration.
actual_travel_distance	NUMBER		Actual distance traveled by assignee.
actual_travel_duration	NUMBER		Actual duration of travel.
actual_travel_duration_uom	VARCHAR2 (3)		UOM of actual travel duration.

Field Name	Data Type	Required	Descriptions and Validations
schedule_flag	VARCHAR2 (1)		Indicates if assignment is flagged to be schedulable.
alarm_type_code	VARCHAR2 (30)		Alarm type code of the assignment.
alarm_contact	VARCHAR2 (200)		Alarm contact of the assignment.
palm_flag	VARCHAR2 (1)		Palm flag of the assignment.
wince_flag	VARCHAR2 (1)		Wince flag of the assignment.
laptop_flag	VARCHAR2 (1)		Laptop flag of the assignment.
device1_flag	VARCHAR2 (1)		Device flag of the assignment.
device2_flag	VARCHAR2 (1)		Device flag of the assignment.
device3_flag	VARCHAR2 (1)		Device flag of the assignment.
resource_territory_id	NUMBER		Territory ID that the assignee is associated to.
assignment_status_id	NUMBER		Status ID of the assignment.
shift_construct_id	NUMBER		Shift construct ID of the assignment.
show_on_calendar	VARCHAR2 (1)		Indicates if assignment to be shown on calendar.

Field Name	Data Type	Required	Descriptions and Validations
category_id	NUMBER		Category ID of the assignment.

Task Dependency Record Type

This composite record type enumerates the dependency information between two tasks that is ultimately stored in the JTF_TASK_DEPENDS table. When *template_flag* ='Y', the dependency exists between the templates. When this field is not set to 'Y' the dependency exists between the tasks.

When *validated_flag* is set to 'Y' and *template_flag* is not set to 'Y', then the API validates the dependency type between the tasks.

The *Adjustment_time* and *adjustment_time_uom* fields indicate the lead or lag time between the two tasks.

```
TYPE task_depends_rec IS RECORD (
        dependent_on_task_id          NUMBER := NULL,
        dependent_on_task_number      NUMBER := NULL,
        dependency_type_code          VARCHAR2(30),
        adjustment_time               NUMBER := NULL,
        adjustment_time_uom           VARCHAR2(3) := NULL,
        validated_flag                VARCHAR2(1) := NULL
    );
```

The following table describes the fields associated with this record type.

task_depends_rec Fields

Field Name	Data Type	Required	Descriptions and Validations
dependent_on_task_id	NUMBER		It defines the task id the current task is dependent on.
dependent_on_task_number	NUMBER		It defines the task number the current task is dependent on.
dependency_type_code	VARCHAR2 (30)		It defines the type of dependency

Field Name	Data Type	Required	Descriptions and Validations
adjustment_time	NUMBER		Adjustment time between two dependendent tasks.
adjustment_time_uom	VARCHAR2 (3)		UOM of adjustment time between two dependendent tasks.
validated_flag	VARCHAR2 (1)		Validation flag to validate the dependency.

Task Reference Record Type

This composite record type enumerates the reference details for a specific task that are ultimately stored in the JTF_TASK_REFERENCES_B/TL table. The *reference_code* field retreives the value from Lookup code stored in the jtf_task_reference_codes table.

```
TYPE task_refer_rec IS RECORD (
        object_type_code              jtf_objects_b.object_code%TYPE,
        object_type_name              jtf_objects_tl.name%TYPE,
        object_name                   VARCHAR2(80),
        object_id                     NUMBER,
        object_details                VARCHAR2(2000),
        reference_code                VARCHAR2(30),
        usage                         VARCHAR2(2000)
    );
```

The following table describes the fields associated with this record type.

task_refer_rec Fields

Field Name	Data Type	Required	Descriptions and Validations
object_type_code	VARCHAR2 (30)		Code of the referenced object type.
object_type_name	VARCHAR2 (30)		Not in use.
object_name	VARCHAR2 (80)		Name/number of the referenced object.

Field Name	Data Type	Required	Descriptions and Validations
object_id	NUMBER		Identifier of the referenced object.
object_details	VARCHAR2 (2000)		Details of the referenced object.
reference_code	VARCHAR2 (30)		Code of the reference.
usage	VARCHAR2 (2000)		Usage of the reference.

Task Recurrence Record Type

This composite record type enumerates recurring task information that is ultimately stored in the JTF_TASK_RECUR_RULES table.

```
TYPE task_recur_rec IS RECORD (
        occurs_which                    NUMBER := NULL,
        day_of_week                     NUMBER := NULL,
        date_of_month                   NUMBER := NULL,
        occurs_month                    NUMBER := NULL,
        occurs_uom                      VARCHAR2(3),
        occurs_every                    NUMBER := NULL,
        occurs_number                   NUMBER := NULL,
        start_date_active               DATE := NULL,
        end_date_active                 DATE := NULL
    );
```

The following table describes the fields associated with this record type.

task_recur_rec Fields

Field Name	Data Type	Required	Descriptions and Validations
resource_type_code	VARCHAR2(30)		Resource type code of the assignee

Field Name	Data Type	Required	Descriptions and Validations
occurs_which	NUMBER		Occurrence in which week/day of the month. in case of monthly/yearly occurrence (e.g., 1 - first, 2 - second, 3 - third, 4 - fourth, 99 - last)
day_of_week	NUMBER		Day of the week for weekly/monthly/yearly occurrence. (E.g., 1 - Sunday, 2 - Monday, 3 - Tuesday, 4 - Wednesday, 5 - Thursday, 6 - Friday, 7 - Saturday. These are available only for monthly/yearly occurrence. 0 - day, 8 - weekday, 9 - weekend day)
date_of_month	NUMBER		Date of monthly/yearly occurrence.
occurs_month	NUMBER		Month of yearly occurrence. (E.g., 1 to 12 for January to December)
occurs_uom	VARCHAR2 (3)		UOM of the occurrence. (E.g. 'DAY' - Daily, 'WK' - Weekly, 'MTH' - Monthly, 'YR' - Yearly)
occurs_every	NUMBER		Occurrence in every how many days/weeks/months/years

Field Name	Data Type	Required	Descriptions and Validations
occurs_number	NUMBER		Number of occurrences.
start_date_active	DATE		Start date of the occurrence.
end_date_active	DATE		End date of the occurrence.

Task Dates Record Type

This composite record type enumerates task date values that are ultimately stored in the JTF_TASK_DATES table. The *date_type_id* field is a foreign key to the jtf_task_date_types_b table.

```
TYPE task_dates_rec IS RECORD (
        date_type_id                    NUMBER DEFAULT NULL,
        date_type_name                  VARCHAR2(30) DEFAULT NULL,
        date_type                       VARCHAR2(30) DEFAULT NULL,
        date_value                      DATE
    );
```

The following table describes the fields associated with this record type.

task_dates_rec Fields

Field Name	Data Type	Required	Descriptions and Validations
date_type_id	NUMBER		Identifier of the date type.
date_type_name	VARCHAR2 (30)		Name of the date type.
date_type	VARCHAR2 (30)		Not in use.
date_value	DATE		Value of date type.

Task Contact Record Type

This composite record type enumerates customer contact information that is ultimately stored in the JTF_TASK_CONTACTS table.

```
TYPE task_contacts_rec IS RECORD (
        contact_id                    NUMBER DEFAULT NULL,
        CONTACT_TYPE_CODE             VARCHAR2(30) DEFAULT NULL,
        ESCALATION_NOTIFY_FLAG        varchar2(1) DEFAULT NULL,
        ESCALATION_REQUESTER_FLAG     varchar2(1) DEFAULT NULL
    );
```

The following table describes the fields associated with this record type.

task_contacts_rec Fields

Field Name	Data Type	Required	Descriptions and Validations
contact_id	NUMBER		Identifier of the contact.
contact_type_code	VARCHAR2 (30)		Code of the contact type.
escalation_notify_flag	VARCHAR2 (1)		Notification flag for escalation.
escalation_requester_ flag	VARCAHR2 (1)		Requester flag for escalation.

Task Notes Record Type

This composite record type enumerates notes information that is ultimately stored in the JTF_NOTES_B/TL table.

```
TYPE task_notes_rec IS RECORD
    (
    parent_note_id   number ,
    org_id       number ,
    notes            varchar2(4000) ,
    notes_detail     varchar2(32767),
    note_status  varchar2(1) ,
    entered_by       number ,
    entered_date     date ,
    note_type        varchar2(30),
    jtf_note_id      number,
    attribute1       varchar2(150),
    attribute2       varchar2(150),
    attribute3       varchar2(150),
    attribute4       varchar2(150),
    attribute5       varchar2(150),
    attribute6       varchar2(150),
    attribute7       varchar2(150),
    attribute8       varchar2(150),
    attribute9       varchar2(150),
    attribute10       varchar2(150),
    attribute11       varchar2(150),
    attribute12       varchar2(150),
    attribute13       varchar2(150),
    attribute14       varchar2(150),
    attribute15       varchar2(150),
    context           varchar2(30)
    );
```

The following table describes the fields associated with this record type.

task_notes_rec Fields

Field Name	Data Type	Required	Descriptions and Validations
parent_note_id	NUMBER		Parent note identifier.
org_id	NUMBER		Organization id.
notes	VARCHAR2 (4000)		Actual note text.
notes_detail	VARCHAR2 (32767)		To store larger notes of 32K.
note_status	VARCHAR2 (1)		Status of the note.
entered_by	NUMBER		User who has entered the note.

Field Name	Data Type	Required	Descriptions and Validations
entered_date	DATE		Date of entering the note.
note_type	VARCHAR2 (30)		Type of the note.
jtf_note_id	NUMBER		Not in use.
attribute1	VARCHAR2 (150)		Descriptive flex field segment.
attribute2	VARCHAR2 (150)		Descriptive flex field segment.
attribute3	VARCHAR2 (150)		Descriptive flex field segment.
attribute4	VARCHAR2 (150)		Descriptive flex field segment.
attribute5	VARCHAR2 (150)		Descriptive flex field segment.
attribute6	VARCHAR2 (150)		Descriptive flex field segment.
attribute7	VARCHAR2 (150)		Descriptive flex field segment.
attribute8	VARCHAR2 (150)		Descriptive flex field segment.
attribute9	VARCHAR2 (150)		Descriptive flex field segment.
attribute10	VARCHAR2 (150)		Descriptive flex field segment.
attribute11	VARCHAR2 (150)		Descriptive flex field segment.

Field Name	Data Type	Required	Descriptions and Validations
attribute12	VARCHAR2 (150)		Descriptive flex field segment.
attribute13	VARCHAR2 (150)		Descriptive flex field segment.
attribute14	VARCHAR2 (150)		Descriptive flex field segment.
attribute15	VARCHAR2 (150)		Descriptive flex field segment.
context	VARCHAR2 (30)		Short description.

Sort Record Type

This composite record type enumerates task sorting information that is ultimately stored in the SORT_REC table.

```
TYPE sort_rec IS RECORD
    (
    field_name      varchar2(30),
    asc_dsc_flag    char(1)        default 'A'
    );
```

The following table describes the fields associated with this record type.

sort_rec Fields

Field Name	Data Type	Required	Descriptions and Validations
field_name	VARCHAR2 (30)		Column name of jtf_tasks_v.
asc_dsc_flag	VARCAHR2 (1)		To sort in ascending or descending order. Default "A".

Package JTF_TASK_ASSIGNMENTS_PUB

The Task Assignment APIs contains the following APIs. Information is available in the Integration Repository.

- Create_Task_Assignment
- Update_Task_Assignment
- Delete_Task_Assignment

Data Structure Specifications

The following data structure is used in the JTF_TASK_ASSIGNMENTS_PUB API.

Task Assignments Record Type

The JTF_TASK_ASSIGNMENTS_REC creates the assignment details for a given task.

```
TYPE task_assignments_rec IS RECORD (
      task_assignment_id            NUMBER,
      object_version_number         NUMBER,
      task_id                       NUMBER,
      resource_type_code            VARCHAR2(30),
      resource_id                   NUMBER,
      assignment_status_id          NUMBER,
      actual_effort                 NUMBER,
      resource_territory_id         NUMBER,
      actual_effort_uom             VARCHAR2(3),
      schedule_flag                 VARCHAR2(1),
      alarm_type_code               VARCHAR2(30),
      alarm_contact                 VARCHAR2(200),
      shift_construct_id            NUMBER,
      sched_travel_distance         NUMBER,
      sched_travel_duration         NUMBER,
       sched_travel_duration_uom     VARCHAR2(3),
      actual_travel_distance        NUMBER,
      actual_travel_duration        NUMBER,
      actual_travel_duration_uom    VARCHAR2(3),
      actual_start_date             DATE,
      actual_end_date               DATE,
      palm_flag                     VARCHAR2(1),
      wince_flag                    VARCHAR2(1),
      laptop_flag                   VARCHAR2(1),
      device1_flag                  VARCHAR2(1),
      device2_flag                  VARCHAR2(1),
      device3_flag                  VARCHAR2(1),
      attribute1                    VARCHAR2(150),
      attribute2                    VARCHAR2(150),
      attribute3                    VARCHAR2(150),
      attribute4                    VARCHAR2(150),
      attribute5                    VARCHAR2(150),
      attribute6                    VARCHAR2(150),
      attribute7                    VARCHAR2(150),
      attribute8                    VARCHAR2(150),
      attribute9                    VARCHAR2(150),
      attribute10                   VARCHAR2(150),
      attribute11                   VARCHAR2(150),
      attribute12                   VARCHAR2(150),
      attribute13                   VARCHAR2(150),
      attribute14                   VARCHAR2(150),
      attribute15                   VARCHAR2(150),
      attribute_category            VARCHAR2(30),
      SHOW_ON_CALENDAR              VARCHAR2(1),
      CATEGORY_ID                   NUMBER
   );
```

The following table describes the fields associated with this record type.

Record Type jtf_task_assignments_rec Fields

Field Name	Data Type	Required	Description
resource_type_code	VARCHAR2(30)		Resource type code of the assignee

Field Name	Data Type	Required	Description
resource_id	NUMBER		Resource identifier of the assignee
assignment_status_id	NUMBER		Status ID of the assignment
actual_effort	NUMBER		Actual effort duration for the assignment
resource_territory_id	NUMBER		Territory ID that the assignee is associated to
actual_effort_uom	VARCHAR2(3)		UOM of actual effort duration
schedule_flag	VARCHAR2(1)		Indicates if assignment is flagged to be schedulable
alarm_type_code	VARCHAR2(30)		Alarm type code of the assignment
alarm_contact	VARCHAR2(200)		Alarm contact of the assignment
shift_construct_id	NUMBER		Shift construct id of the assignment
sched_travel_distance	NUMBER		Scheduled distance to be traveled by the assignee
sched_travel_duration	NUMBER		Scheduled duration of travel
sched_travel_duration_uom	VARCHAR2(3)		UOM of scheduled travel duration
actual_travel_distance	NUMBER		Actual distance traveled by assignee
actual_travel_duration	NUMBER		Actual duration of travel
actual_travel_duration_uom	VARCHAR2(3)		UOM of actual travel duration

Field Name	Data Type	Required	Description
actual_start_date	DATE		Actual start date of the assignment
actual_end_date	DATE		Actual end date of the assignment
palm_flag	VARCHAR2(1)		Palm flag of the assignment
wince_flag	VARCHAR2(1)		Wince flag of the assignment
laptop_flag	VARCHAR2(1)		Laptop flag of the assignment
device1_flag	VARCHAR2(1)		Device flag of the assignment
device2_flag	VARCHAR2(1)		Device flag of the assignment
device3_flag	VARCHAR2(1)		Device flag of the assignment
attribute1	VARCHAR2(150)		Descriptive flexfield segment column
attribute2	VARCHAR2(150)		Descriptive flexfield segment column
attribute3	VARCHAR2(150)		Descriptive flexfield segment column
attribute4	VARCHAR2(150)		Descriptive flexfield segment column
attribute5	VARCHAR2(150)		Descriptive flexfield segment column
attribute6	VARCHAR2(150)		Descriptive flexfield segment column
attribute7	VARCHAR2(150)		Descriptive flexfield segment column
attribute8	VARCHAR2(150)		Descriptive flexfield segment column

Field Name	Data Type	Required	Description
attribute9	VARCHAR2(150)		Descriptive flexfield segment column
attribute10	VARCHAR2(150)		Descriptive flexfield segment column
attribute11	VARCHAR2(150)		Descriptive flexfield segment column
attribute12	VARCHAR2(150)		Descriptive flexfield segment column
attribute13	VARCHAR2(150)		Descriptive flexfield segment column
attribute14	VARCHAR2(150)		Descriptive flexfield segment column
attribute15	VARCHAR2(150)		Descriptive flexfield segment column
attribute_category	VARCHAR2(30)		Descriptive flexfield structure defining column
show_on_calendar	VARCHAR2(1)		Indicates if assignment to be shown on calendar
category_id	NUMBER		Category IDof the assignment

Package JTF_TASK_REFERENCES_PUB

The JTF_Task_References_Pub package contains the following APIs:

- Create_References
- Update_References
- Delete_References

Data Structure Specifications

The following data structure is used in the JTF_TASK_REFERENCES_PUB API.

References Record Type

The REFERENCES_REC creates the reference for a given task.

```
TYPE references_rec IS RECORD
(
 TASK_REFERENCE_ID                NUMBER,
 TASK_ID                          NUMBER,
 OBJECT_TYPE_CODE                 VARCHAR2(30),
 OBJECT_NAME                      VARCHAR2(80),
 OBJECT_ID                        NUMBER,
 OBJECT_DETAILS                   VARCHAR2(2000),
 REFERENCE_CODE                   VARCHAR2(30),
 ATTRIBUTE1                       VARCHAR2(150),
 ATTRIBUTE2                       VARCHAR2(150),
 ATTRIBUTE3                       VARCHAR2(150),
 ATTRIBUTE4                       VARCHAR2(150),
 ATTRIBUTE5                       vARCHAR2(150),
 ATTRIBUTE6                       VARCHAR2(150),
 ATTRIBUTE7                       VARCHAR2(150),
 ATTRIBUTE8                       VARCHAR2(150),
 ATTRIBUTE9                       VARCHAR2(150),
 ATTRIBUTE10                      VARCHAR2(150),
 ATTRIBUTE11                      VARCHAR2(150),
 ATTRIBUTE12                      VARCHAR2(150),
 ATTRIBUTE13                      VARCHAR2(150),
 ATTRIBUTE14                      VARCHAR2(150),
 ATTRIBUTE15                      VARCHAR2(150),
 ATTRIBUTE_CATEGORY               VARCHAR2(30),
 USAGE                            VARCHAR2(2000),
 OBJECT_VERSION_NUMBER            NUMBER
 );
```

The following table describes the fields associated with the references record type.

references_rec Fields

Field Name	Data Type	Required	Descriptions and Validations
task_reference_id	NUMBER		Unique task reference identifier
task_id	NUMBER		Unique identifier for the task to be used

Field Name	Data Type	Required	Descriptions and Validations
object_type_code	VARCHAR2 (30)		Object type code for reference creation
object_name	VARCHAR2 80)		Object name for reference creation
object_details	VARCHAR2 (2000)		Object details for reference creation
reference_code	VARCHAR2 (30)		Reference code for reference creation
attribute1	VARCHAR2(150)		Descriptive flexfield segment
attribute2	VARCHAR2(150)		Descriptive flexfield segment
attribute3	VARCHAR2(150)		Descriptive flexfield segment
attribute4	VARCHAR2(150)		Descriptive flexfield segment
attribute5	VARCHAR2(150)		Descriptive flexfield segment
attribute6	VARCHAR2(150)		Descriptive flexfield segment
attribute7	VARCHAR2(150)		Descriptive flexfield segment
attribute8	VARCHAR2(150)		Descriptive flexfield segment
attribute9	VARCHAR2(150)		Descriptive flexfield segment
attribute10	VARCHAR2(150)		Descriptive flexfield segment

Field Name	Data Type	Required	Descriptions and Validations
attribute11	VARCHAR2(150)		Descriptive flexfield segment
attribute12	VARCHAR2(150)		Descriptive flexfield segment
attribute13	VARCHAR2(150)		Descriptive flexfield segment
attribute14	VARCHAR2(150)		Descriptive flexfield segment
attribute15	VARCHAR2(150)		Descriptive flexfield segment
attribute_category	VARCHAR2(30)		Attribute category
usage	VARCHAR2(2000)		Flexfield segment. Usage for references
object_version_number	NUMBER		Object version number of the current reference record

Messages and Notifications

The APIs contained in the Task Manager public packages generate messages and notifications as needed. Some of these messages are common to all of the Task Manager APIs and others are generated by APIs in specific Task Manager packages. Both types of messages are detailed in the following sections.

- Common Messages, page 2-27
- JTF_TASK_ASSIGNMENTS_PUB, page 2-34
- JTF_TASK_REFERENCES_PUB, page 2-35
- JTF_TASKS_PUB, page 2-32

Note: It is not required that all status notifications provide a

number identifier along with the message, although, in many cases, it is provided.

Common Messages

The following table lists the messages and notifications generated by many of the APIs contained in Task Manager public packages.

Common Task API Messages

Number	Type	Name	Text
210021	E	JTF_TASK_INVALID_ASSIGNED_NAME	User &P_ASSIGNED_BY_NAME is invalid or disabled.
210022	E	JTF_TASK_INVALID_CURRENCY_CODE	Currency &P_CURRENCY_CODE is invalid or disabled. Please enter a valid currency.
210012	E	JTF_TASK_INVALID_CUST_ACCT_ID	Customer Account &P_CUST_ACCOUNT_ID is invalid or disabled. Please enter a valid customer account.
210046	E	JTF_TASK_INVALID_DATES	&P_DATE_TAG start date cannot be greater than &P_DATE_TAG end date.
210056	E	JTF_TASK_INVALID_DATE_TYPE_ID	Date type &P_DATE_TYPE_ID is invalid. Please enter a valid date type.
210020	E	JTF_TASK_INVALID_DEPENDS_ID	Dependency &P_DEPENDENCY_ID does not exist. Please enter a valid dependency ID.
210067	E	JTF_TASK_INVALID_ESC_DTLS	Invalid escalation details.

Number	Type	Name	Text
210018	E	JTF_TASK_INVALID_FLAG	&P_FLAG_NAME must be Yes or No.
210032	E	JTF_TASK_INVALID_OBJECT_CODE	&P_OBJECT_CODE is either disabled or invalid.
210031	E	JTF_TASK_INVALID_OBJECT_NAME	&P_OBJECT_TYPE_CODE &P_OBJECT_NAME is either disabled or invalid.
210068	E	JTF_TASK_INVALID_OWNER_ID	Owner &P_OWNER_ID is invalid.
210013	E	JTF_TASK_INVALID_PARTY_ID	Customer &P_PARTY_ID is invalid or disabled.
210014	E	JTF_TASK_INVALID_PARTY_NUMBER	Customer &PARTY_NUMBER is invalid or disabled.
210001	E	JTF_TASK_INVALID_PRIORITY_ID	Task Priority &P_TASK_PRIORITY_ID is either disabled or invalid.
210002	E	JTF_TASK_INVALID_PRIORITY_NAME	Task Priority &P_TASK_PRIORITY_NAME is either disabled or invalid.
210034	E	JTF_TASK_INVALID_QUANTITY	Value for &P_TAG should be greater than 0.
210015	E	JTF_TASK_INVALID_SITE_ID	Address &P_PARTY_SITE_ID is invalid or disabled.
210016	E	JTF_TASK_INVALID_SITE_NUMBER	Address &P_PARTY_SITE_NUMBER is invalid or disabled.
210047	E	JTF_TASK_INVALID_SOURCE_DTLS	Invalid source object details provided. Cause: SOURCE OBJECT TYPE CODE, SOURCE OBJECT ID AND NAME SHOULD BE PROVIDED.

Number	Type	Name	Text
210041	E	JTF_TASK_INVALID_STATUS_ID	Status &P_TASK_STATUS_ID is either disabled or invalid. Please enter a valid task status.
210037	E	JTF_TASK_INVALID_STATUS_NAME	Task status &P_TASK_STATUS_NAME is either disabled or invalid. Please enter a valid task status.
210051	E	JTF_TASK_INVALID_TASK_ID	Task &P_TASK_ID is invalid. Please enter a valid task ID.
210052	E	JTF_TASK_INVALID_TASK_NUMBER	Task number &P_TASK_NUMBER is invalid. Please enter a valid task number.
210049	E	JTF_TASK_INVALID_TEMP_GRP_ID	Task template group &P_TASK_TEMPLATE_GROUP_ID is either disabled or invalid.
210048	E	JTF_TASK_INVALID_TEMP_GRP_NAME	Task template group &P_TASK_TEMP_GROUP_NAME is either disabled or invalid. Please enter a valid task template name.
210006	E	JTF_TASK_INVALID_TEMP_NUMBER	Task Template &P_TASK_TEMPLATE_NUMBER is invalid. Please enter a valid template number.
210065	E	JTF_TASK_INVALID_TERR_ID	Territory &P_TERR_ID is disabled or invalid. Please enter a valid territory ID.
210027	E	JTF_TASK_INVALID_TIMEZONE_ID	Timezone for &P_TIMEZONE_ID is either disabled or invalid.
210026	E	JTF_TASK_INVALID_TIMEZONE_NAME	Timezone for &P_TIMEZONE_NAME is either disabled or invalid.

Number	Type	Name	Text
210003	E	JTF_TASK_INVALID_TYPE_ID	Task Type &P_TASK_TYPE_ID is either disabled or invalid. Please enter a valid task type.
210039	E	JTF_TASK_INVALID_TYPE_NAME	Task type &P_TASK_TYPE_NAME is either disabled or invalid. Please enter a valid task type.
210004	E	JTF_TASK_INVALID_UOM	Unit of Measure &P_UOM_CODE is either disabled or invalid. Check the profile option "Time unit of measure class" and make sure it matches the value of the class of unit of measure.
210024	E	JTF_TASK_MISSING_COST	Please enter a cost for &P_CURRENCY_CODE.
210035	E	JTF_TASK_MISSING_QUANTITY	Please enter a quantity for &P_TAG.
210055	E	JTF_TASK_MISSING_TASK	Task is missing.
210000	E	JTF_TASK_MISSING_UOM	Unit of measure for &P_TAG should be specified.
210070	E	JTF_TASK_PARENT	Parent task does not have the same source object details as the current task. Please enter proper parent task.
210069	E	JTF_TASK_PARENT_TYPE_CODE	Parent task should have the same source document types as the current task. Please enter proper parent task.
210029	U	JTF_TASK_UNKNOWN_ERROR	&P_TEXT. Please contact your system administrator.
210023	E	JTF_TASK_MISSING_CURRENCY_CODE	Please enter a currency code for &P_COST.

Number	Type	Name	Text
210066	E	JTF_TASK_INVALID_TERR_NAME	Territory &P_TERR_NAME is disabled or invalid. Please enter a valid territory name.
210118	E	JTF_TASK_MISSING_RETURN_STATUS	Return status from internal API hook is missing. Procedure name is &P_PROCEDURE.
210120	E	JTF_TASK_MISSING_LOOKUP	Lookup code (&P_LOOKUP_CODE) is not defined for lookup type &P_LOOKUP_TYPE.
N/A	E	JTF_TASK_DISTANCE_UNITS	Please enter a valid quantity for &P_DISTANCE_TAG.
N/A	E	JTF_TASK_INVALID_CONTACT	Please enter valid details for the contact.
N/A	E	JTF_TASK_INVALID_DEFAULT_OWNER	Incorrect default owner. Ensure that profiles for defaulting the owner and owner type code are set up correctly.
N/A	E	JTF_TASK_MISSING_CONTACT	Please enter details for the contact.
N/A	E	JTF_TASK_MISSING_USER_MAPPING	Please map the user to a resource.
N/A	E	JTF_TASK_MISSING_PHONE	Please enter phone details.
N/A	E	JTF_TASK_INVALID_PHONE_ID	Please enter valid phone details. (Contact point identifier = &P_PHONE_ID).
210005	E	JTF_TASK_INVALID_DEPENDS_CODE	Dependency Code &P_DEPENDENCY_CODE is either disabled or invalid. Please enter a valid dependency code.

Number	Type	Name	Text
210008	E	JTF_TASK_INVALID_TEMP_ID	Task Template &P_TASK_TEMPLATE_ID is invalid. Please enter a valid task template ID.

JTF_TASKS_PUB

The following table lists the messages and notifications generated by the APIs contained in the JTF_TASKS_PUB package.

JTF_TASK_ API Messages

Number	Type	Name	Text
210060	E	JTF_TASK_CONVERTING_NUMBER	Error converting template number to task number.
210021	E	JTF_TASK_INVALID_ASSIGNED_NAME	User &P_ASSIGNED_BY_NAME is invalid or disabled.
210046	E	JTF_TASK_INVALID_DATES	&P_DATE_TAG start date cannot be greater than &P_DATE_TAG end date.
210002	E	JTF_TASK_INVALID_PRIORITY_NAME	Task priority &P_TASK_PRIORITY_NAME is either disabled or invalid.
210051	E	JTF_TASK_INVALID_TASK_ID	Task &P_TASK_ID is invalid. Please enter a valid task ID.
210052	E	JTF_TASK_INVALID_TASK_NUMBER	Task number &P_TASK_NUMBER is invalid. Please enter a valid task number.
210011	E	JTF_TASK_INVALID_CUST_ACCT_NUM	Customer account &P_CUST_ACCOUNT_NUMBER is disabled or invalid. Please enter a valid customer account.

Number	Type	Name	Text
210036	E	JTF_TASK_MISSING_STATUS	Please enter a status for this task.
210055	E	JTF_TASK_MISSING_TASK	Task is missing.
210007	E	JTF_TASK_MISSING_TEMP_GRP	Task template group is missing.
210040	E	JTF_TASK_MISSING_TYPE	Please enter a task type.
210029	U	JTF_TASK_UNKNOWN_ERROR	&P_TEXT. Please contact your system administrator.
210109	E	JTF_TK_EXP_FILE_NAME_NULL	Please enter an output file name.
210098	E	JTF_TK_INV_QRY_NXT	Invalid value. Query or next code can be either Q or N.
210099	E	JTF_TK_INV_SHOW_ALL	Please enter Y (yes) or N (no).
210112	E	JTF_TK_NO_DATE	Please enter a Created date for the current task ID.
210101	E	JTF_TK_NULL_REC_WANT	Please enter the number of desired records.
210100	E	JTF_TK_NULL_STRT_PTR	Please enter the start record number.
210104	E	JTF_TK_OBJECT_TYPE_ID_RQD	Please enter both object ID and object type code.
210106	E	JTF_TK_QRY_NXT_INV_DT_TYPE	Please enter a valid date type.
210105	E	JTF_TK_QRY_NXT_INV_QRY_TYP	Please enter a valid query type.
210111	E	JTF_TK_QRY_NXT_INV_STRT_END_DT	Please enter valid start and end dates.
210107	E	JTF_TK_QRY_NXT_NUL_ASGND_BY	Please enter a value for the Assigned by ID field.

Number	Type	Name	Text
210108	E	JTF_TK_QRY_NXT_NUL_OWNER	Please enter both an owner identifier and an owner type.
210117	E	JTF_TASK_RESOURCE_LOCKED	Resource is locked or deleted.
N/A	E	JTF_TASK_MISSING_TASK_TYPE	Please enter a task type name or task type ID.
210103	E	JTF_TASK_INV_TK_NAME	Please enter a valid task name.
210110	E	JTF_TK_EXP_TABLE_EMPTY	Input task table is NULL. Please pass valid values.
211137	E	JTF_TASK_LOCATION_EXIST	Location ID already exists. You can't set address ID.
211136	E	JTF_TASK_LOCATION_VALIDATION	Task can have either location ID (party location) or address ID (party site), but not both.

JTF_TASK_ASSIGNMENTS_PUB

The following table lists the messages and notifications generated by the APIs contained in the JTF_TASK_ASSIGNMENTS_PUB package.

JTF_TASK_ASSIGNMENTS_PUB API Messages

Number	Type	Name	Text
210115	E	JTF_TASK_DELETING_TK_ASS	Unexpected errors while deleting task assignment.
210086	E	JTF_TASK_INV_ALA_CON	Alarm contact &P_ALARM_CONTACT is invalid or disabled. Please enter a valid alarm contact.
210085	E	JTF_TASK_INV_ALA_TYPE	Alarm &P_ALARM_TYPE_CODE is invalid or disabled.

Number	Type	Name	Text
210082	E	JTF_TASK_INV_RES_TYP_COD	Resource &P_RESOURCE_TYPE_CODE is invalid or disabled. Please enter a valid resource type.
210114	E	JTF_TASK_INV_TK_ASS	Task assignment &P_TASK_ASSIGNMENT_ID is invalid or missing. Please enter a valid task assignment.
210055	E	JTF_TASK_MISSING_TASK	Task is missing.
210080	E	JTF_TASK_NULL_RES_ID	Please enter a resource ID.
210113	E	JTF_TASK_NULL_TK_ASS	Task assignment ID is missing.
210029	U	JTF_TASK_UNKNOWN_ERROR	&P_TEXT. Please contact your system administrator.
210117	E	JTF_TASK_RESOURCE_LOCKED	Resource is locked or deleted.

JTF_TASK_REFERENCES_PUB

The following table lists the messages and notifications generated by the APIs contained in the JTF_TASK_REFERENCES_PUB package.

JTF_TASK_REFERENCES_PUB API Messages

Number	Type	Name	Text
210032	E	JTF_TASK_INVALID_OBJECT_CODE	&P_OBJECT_CODE is either disabled or invalid.
210044	E	JTF_TASK_INVALID_OBJECT_ID	&P_OBJECT_ID is invalid.
210031	E	JTF_TASK_INVALID_OBJECT_NAME	&P_OBJECT_TYPE_CODE &P_OBJECT_NAME is either disabled or invalid.

Number	Type	Name	Text
210038	E	JTF_TASK_INVALID_REFER	Task reference &P_TASK_REFERENCE_ID is invalid.
210050	E	JTF_TASK_INVALID_REFER_CODE	Reference code &P_REFERENCE_CODE is either disabled or invalid.
210053	E	JTF_TASK_INVALID_REFER_DETAILS	Invalid reference details.
210028	E	JTF_TASK_MISSING_OBJECT_CODE	Please enter a source object type code.
210025	E	JTF_TASK_MISSING_OBJECT_NAME	Object name is missing.
210042	E	JTF_TASK_MISSING_REFER	Please enter a task reference.
210055	E	JTF_TASK_MISSING_TASK	Task is missing .
210029	U	JTF_TASK_UNKNOWN_ERROR	&P_TEXT. Please contact your system administrator.
210117	E	JTF_TASK_RESOURCE_LOCKED	Resource is locked or deleted.

Sample Code

This section contains SQL scripts that call the Task Manager public APIs stored in the following packages and insert values as required:

- JTF_TASKS_PUB
- JTF_TASK_ASSIGNMENTS_PUB
- JTF_TASK_REFERENCES_PUB

Package JTF_TASKS_PUB

The SQL scripts in this section create a task by calling the Create_Task API contained in the JTF_TASKS_PUB package and by providing them with the required values.

Create_Task

This script calls the Create_Task API and creates tasks with the following information.

Create_Task API Sample Code Variables

Variable	Description
p_user_name	Login user name is entered by a user.
p_task_name	Task name is entered by a user.
p_task_type	The type of task is entered by a user, such as "Meeting", or "Lunch".
p_task_priority	Task priority is entered by a user, such as "High", "Medium", or "Low".
p_task_status	Task status is entered by a user, such as "Open".
p_show_on_cal	A flag used to determine if the task will be shown in the calendar view.
p_start_date	The planned start date is entered by a user.
p_end_date	The planned end date is entered by a user.

```
set serveroutput ON
accept p_user_name prompt 'Enter user name : '
accept p_task_name prompt 'Enter task name : '
accept p_task_type prompt 'Enter task type name (Default=Meeting): '
accept p_priority prompt 'Enter priority name (Default=Medium)  : '
accept p_task_status prompt 'Enter task status name (Default=Open) : '
accept p_show_on_cal prompt 'Enter Y if you want to show this task on
calendar view (Default=N) : '
accept p_start_date prompt 'Enter task start date (Format=MMDDYYYY
HH24MI) : '
accept p_end_date prompt 'Enter task end date (Format=MMDDYYYY HH24MI) :
'

declare
    l_user_name          fnd_user.user_name%TYPE :=
upper('&p_user_name');
    l_task_name          jtf_tasks_tl.task_name%TYPE := '&p_task_name');
    l_task_type          jtf_task_types_tl.name%TYPE :=
NVL('&p_task_type','Meeting');
    l_task_priority      jtf_task_priorities_tl.name%TYPE :=
NVL('&p_priority', 'Medium');
    l_task_status        jtf_task_statuses_tl.name%TYPE :=
NVL('&p_task_status','Open');
    l_show_on_cal        jtf_task_all_assignments.show_on_calendar%TYPE
:= NVL(upper('&p_show_on_cal'),'N');
    l_planned_start_date DATE := NVL(TO_DATE('&p_start_date','MMDDYYYY
HH24MI'),SYSDATE);
    l_planned_end_date   DATE := NVL(TO_DATE('&p_end_date','MMDDYYYY
HH24MI'),SYSDATE);

cursor c_login_user (b_user_name VARCHAR2) is
    select user_id
      from fnd_user
     where user_name = b_user_name;

    cursor c_owner (b_user_id NUMBER) is
    select resource_id
      from jtf_rs_emp_dtls_vl r
     where r.user_id = b_user_id;

    cursor c_task_type (b_name VARCHAR2) is
     select task_type_id
      from jtf_task_types_tl
     where name = b_name;

    cursor c_task_status (b_status VARCHAR2) is
     select task_status_id
      from jtf_task_statuses_vl
     where name = b_status
       and task_status_flag = 'Y';

    cursor c_task_priority (b_priority VARCHAR2) is
    select task_priority_id
      from jtf_task_priorities_tl
     where name = b_priority;

    l_user_id       NUMBER;
    l_resource_id   NUMBER;
    l_task_type_id  NUMBER;
    l_task_status_id  NUMBER;
    l_task_priority_id  NUMBER;
```

```
l_task_id       NUMBER;
   l_return_status VARCHAR2(1);
    l_msg_count     NUMBER;
    l_msg_data      VARCHAR2(1000);
    begin
    dbms_output.put_line('-----------------------------------------');

    open c_login_user(l_user_name);
    fetch c_login_user Into l_user_id;
    if c_login_user%NOTFOUND
    Then
        close c_login_user;
        raise_application_error(-20000,'User name '||l_user_name||' is
not found.');
    end if;
    close c_login_user;
    dbms_output.put_line('User Id : '||l_user_id);
    open c_owner (l_user_id);
    fetch c_owner Into l_resource_id;
    if c_owner%NOTFOUND
    Then
        close c_owner;
        raise_application_error(-20000,'Resource for the user name
'||l_user_name||' is not found.');
    end if;
    close c_owner;

    dbms_output.put_line('Owner Resource Id : '||l_resource_id);
    open c_task_type (l_task_type);
    fetch c_task_type into l_task_type_id;
    if c_task_type%NOTFOUND
    Then
        close c_task_type;
        raise_application_error(-20000,'Task type '||l_task_type||' is
not found.');
    end if;
    close c_task_type;

    open c_task_status(l_task_status);
    fetch c_task_status into l_task_status_id;
    if c_task_status%NOTFOUND
    Then
        close c_task_status;
        raise_application_error(-20000,'Task status '||l_task_status||'
is not found.');
    end if;
    close c_task_status;

    open c_task_priority(l_task_priority);
    fetch c_task_priority into l_task_priority_id;
    if c_task_priority%NOTFOUND
    Then
       close c_task_priority;
        raise_application_error(-20000,'Task status
'||l_task_priority||' is not found.');
    end if;
    close c_task_priority;

    fnd_global.apps_initialize(l_user_id, 0, 690);

    JTF_TASKS_PUB.CREATE_TASK(
```

```
p_api_version          => 1.0 ,
        p_init_msg_list        => fnd_api.g_true,
        p_commit               => fnd_api.g_false,
        p_task_name            => l_task_name,
        p_task_type_id         => l_task_type_id,
        p_task_status_id       => l_task_status_id,
        p_task_priority_id     => l_task_priority_id,
        p_owner_type_code      => 'RS_EMPLOYEE',
        p_owner_id             => l_resource_id,
        p_show_on_calendar     => l_show_on_cal,
        p_planned_start_date   => l_planned_start_date,
        p_planned_end_date     => l_planned_end_date,
        p_date_selected        => 'P',
        x_return_status        => l_return_status,
        x_msg_count            => l_msg_count,
        x_msg_data             => l_msg_data,
        x_task_id              => l_task_id
        p_location_id          => null
    );
    IF l_return_status <> fnd_api.g_ret_sts_success
    THEN
        IF l_msg_count > 0
        THEN
           l_msg_data := NULL;
           FOR i IN 1..l_msg_count LOOP
               l_msg_data := l_msg_data ||' '||fnd_msg_pub.get(1,'F');
           END LOOP;
           fnd_message.set_encoded(l_msg_data);
           dbms_output.put_line(l_msg_data);
        END IF;
        ROLLBACK;
    ELSE
        dbms_output.put_line('Task Id = '||l_task_id);
        dbms_output.put_line('Return Status = '||l_return_status);
        COMMIT;
    END IF;
    dbms_output.put_line('-----------------------------------------');
end;
```

Package JTF_TASK_ASSIGNMENTS_PUB

The SQL scripts in this section create a task assignment for a specific task by calling the Create_Task_Assignment API contained in the JTF_TASK_ASSIGNMENTS_PUB package and by providing them with the required values.

Create_Task_Assignment

This script calls the Create_Task_Assignment API and adds an assignee with the following information.

Create_Task_Assignment API Sample Code Variables

Variable	Description
p_user_name	Login user name is entered by a user.
p_task_id	Task ID is entered by a user.
p_emp_resource_id	Resource ID for an employee is entered by a user.
p_show_on_cal	A flag used to determine if the task will be shown in the calendar view.
p_assignment_status	Task assignment status is entered by a user.

```
set serveroutput ON
accept p_user_name prompt 'Enter user name : '
accept p_task_id prompt 'Enter task id : '
accept p_emp_resource_id prompt 'Enter resource id for an employee : '
accept p_show_on_cal prompt 'Do you want to show this task on calendar
view ? (Y or N) : '
accept p_assignment_status prompt 'Enter Assignment status name : '
accept p_enable_workflow prompt 'Do you want to launch workflow? (Y or
N):'
accept p_abort_workflow_prompt 'Do you want to abort previous workflow?
(Y or N):'

declare
    l_user_name fnd_user.user_name%TYPE := upper('&p_user_name');
    l_task_id jtf_tasks_b.task_id%TYPE := '&p_task_id';

    l_emp_resource_id jtf_rs_emp_dtls_vl.resource_id%TYPE :=
'&p_emp_resource_id';
    l_show_on_cal jtf_task_all_assignments.show_on_calendar%TYPE :=
NVL(upper('&p_show_on_cal'),'N');

    l_enable_workflow  VARCHAR2(1) :=NVL('&p_enable_workflow', 'N');
    l_abort_workflow   VARCHAR2(1) :=NVL('&p_abort_workflow', 'N');
    l_assignment_status jtf_task_statuses_vl.name%TYPE :=
NVL('&p_assignment_status', 'Accepted');

    cursor c_login_user (b_user_name VARCHAR2) is
    select user_id
      from fnd_user
     where user_name = b_user_name;

    cursor c_assignee is
    select resource_id
      from jtf_rs_emp_dtls_vl r
     where r.resource_id=l_emp_resource_id;

    cursor c_assignment_status (b_status VARCHAR2) is
    select task_status_id
      from jtf_task_statuses_vl
     where name = b_status
       and assignment_status_flag='Y';

    l_user_id               NUMBER;
    l_assignee_id           NUMBER;
    l_assignment_status_id  NUMBER;
    l_task_assignment_id    NUMBER;
    l_return_status VARCHAR2(1);
    l_msg_count     NUMBER;
    l_msg_data      VARCHAR2(1000);

Begin
    dbms_output.put_line('-----------------------------------------');

   open c_login_user(l_user_name);
    fetch c_login_user into l_user_id;
    if c_login_user%NOTFOUND
    Then
        close c_login_user;
        raise_application_error(-20000,'User name '||l_user_name||' is
not found.');
    end if;
```

```
close c_login_user;

    dbms_output.put_line('User Id : '||l_user_id);
    open c_assignee;
    fetch c_assignee into l_assignee_id;
    if c_assignee%NOTFOUND
    Then
        close c_assignee;
        raise_application_error(-20000,'Employee resource id
'||l_emp_resource_id||' is not found.');
    end if;
    close c_assignee;

    open c_assignment_status(l_assignment_status);
    fetch c_assignment_status into l_assignment_status_id;
    if c_assignment_status%NOTFOUND
    Then
        close c_assignment_status;
        raise_application_error(-20000,'Assignment status
'||l_assignment_status||' is not found.');
    end if;
    close c_assignment_status;

    fnd_global.apps_initialize(l_user_id, 0, 690);

    JTF_TASK_ASSIGNMENTS_PUB.CREATE_TASK_ASSIGNMENT(
        p_api_version          => 1.0,
        p_init_msg_list        => fnd_api.g_true,
        p_commit               => fnd_api.g_false,
        p_task_id              => l_task_id,
        p_resource_type_code   => 'RS_EMPLOYEE',
        p_resource_id          => l_assignee_id,
        p_assignment_status_id => l_assignment_status_id,
        p_show_on_calendar     => l_show_on_cal,
        p_enable_workflow      => l_enable_workflow,
        p_abort_workflow       => l_abort_workflow
        p_object_capacity_id   => null,
        p_free_busy_type       => null,
        x_return_status        => l_return_status,
        x_msg_count            => l_msg_count,
        x_msg_data             => l_msg_data,
        x_task_assignment_id   => l_task_assignment_id
    );
    IF l_return_status <> fnd_api.g_ret_sts_success
    THEN
        IF l_msg_count > 0
        THEN
           l_msg_data := NULL;
           FOR i IN 1..l_msg_count LOOP
               l_msg_data := l_msg_data ||' '||fnd_msg_pub.get(1,'F');
           END LOOP;
           fnd_message.set_encoded(l_msg_data);
           dbms_output.put_line(l_msg_data);
        END IF;
        ROLLBACK;
    ELSE
        dbms_output.put_line('l_task_assignment_id =
'||l_task_assignment_id);
        dbms_output.put_line('Return Status = '||l_return_status);
        COMMIT;
    END IF;
```

```
Dbms_output.put_line('-----------------------------------');
end;
```

Package JTF_TASK_REFERENCES_PUB

The SQL scripts in this section create references to a specific task by calling the Create_References API contained in the JTF_TASK_REFERENCES_PUB package and by providing them with the required values.

Create_References

This script calls the Create_References API and creates a reference with the following information.

Create_References API Sample Code Variables

Variable	Description
p_user_name	Login user name is entered by a user.
p_task_id	The ID of the task which the reference is added to is entered by a user.
p_source_object_type	Source object type is entered by a user, such as "Party".
p_source_object_name	Source object name is entered by a user, such as "Business World".
p_source_object_id	Source object ID is entered by a user, such as "1001".

```
set serveroutput ON
accept p_user_name prompt 'Enter user name : '
accept p_task_id prompt 'Enter task id : '
accept p_source_object_type prompt 'Enter source object type (e.g.
PARTY) : '
accept p_source_object_name prompt 'Enter source object name (Enter
party name if object type is PARTY): '
accept p_source_object_id prompt 'Enter source object id (Enter party id
if object type is PARTY): '

declare
    l_user_name           fnd_user.user_name%TYPE :=
upper('&p_user_name');
    l_task_id             jtf_tasks_b.task_id%TYPE := '&p_task_id';
    l_source_object_type  jtf_tasks_b.source_object_type_code%TYPE :=
upper('&p_source_object_type');
    l_source_object_name  jtf_tasks_b.source_object_name%TYPE :=
'&p_source_object_name';
    l_source_object_id    jtf_tasks_b.source_object_id%TYPE :=
'&p_source_object_id';

    cursor c_login_user (b_user_name VARCHAR2) is
    select user_id
      from fnd_user
     where user_name = b_user_name;

    l_user_id               NUMBER;
    l_task_reference_id     NUMBER;
    l_return_status VARCHAR2(1);
    l_msg_count     NUMBER;
    l_msg_data      VARCHAR2(1000);

begin
    dbms_output.put_line('-----------------------------------------');

    open c_login_user(l_user_name);
    fetch c_login_user into l_user_id;

    if c_login_user%NOTFOUND
    Then
        close c_login_user;
        raise_application_error(-20000,'User name '||l_user_name||' is
not found.');
    end if;
    close c_login_user;

    dbms_output.put_line('User Id : '||l_user_id);

    fnd_global.apps_initialize(l_user_id, 0, 690);

    JTF_TASK_REFERENCES_PUB.CREATE_REFERENCES(
        p_api_version         => 1.0,
        p_init_msg_list       => fnd_api.g_true,
        p_commit              => fnd_api.g_false,
        p_task_id             => l_task_id,
        p_object_type_code    => l_source_object_type,
        p_object_name         => l_source_object_name,
        p_object_id           => l_source_object_id,
        x_return_status       => l_return_status,
        x_msg_count           => l_msg_count,
        x_msg_data            => l_msg_data,
```

```
x_task_reference_id    => l_task_reference_id
    );
    IF l_return_status <> fnd_api.g_ret_sts_success
    THEN
        IF l_msg_count > 0
        THEN
           l_msg_data := NULL;
           FOR i IN 1..l_msg_count LOOP
               l_msg_data := l_msg_data ||' '||fnd_msg_pub.get(1,'F');
           END LOOP;
           fnd_message.set_encoded(l_msg_data);
           dbms_output.put_line(l_msg_data);
        END IF;
        ROLLBACK;
    ELSE
        dbms_output.put_line('l_task_reference_id =
'||l_task_reference_id);
        dbms_output.put_line('Return Status = '||l_return_status);
        COMMIT;
    END IF;
    Dbms_output.put_line('-----------------------------')
end;
```

3

Notes Public APIs

This chapter covers the following topics:

- Notes Overview
- Package JTF_NOTES_PUB
- Data Structure Specifications for JTF_NOTES_PUB
- Notes Public APIs
- Note Source and Note Context
- Party Relationships
- Secure_Create_Note
- Secure_Update_Note
- Secure_Delete_Note
- Messages and Notifications
- Sample Code

Notes Overview

A note provides a quick and easy way to enter in any information for an E-Business Suite business object. It is useful for attaching information such as directions, special instructions, or reminders. One or more people with access to the business object may enter in a note.

Package JTF_NOTES_PUB

All public procedures (APIs) relating to creating and updating notes are stored in the JTF_NOTES_PUB package. There are two public Note APIs:

- Secure_Create_Note

- Secure_Update_Note
- Secure_Delete_Note

Data Structure Specifications for JTF_NOTES_PUB

Package JTF_NOTES_PUB contain one data structure.

Record jtf_note_contexts_rec_type

The jtf_note_contexts_rec is used to store information about the context (a related business object) for the note. This record type is used to create both of the following:

- jtf_note_contexts_tab
- jtf_note_contexts_tab_dflt

```
TYPE jtf_note_contexts_rec_type IS RECORD
( note_context_id        NUMBER,
  jtf_note_id            NUMBER,
  note_context_type      VARCHAR2(240),
  note_context_type_id   NUMBER,
  last_update_date       DATE,
  last_updated_by        NUMBER(15),
  creation_date          DATE,
  created_by             NUMBER,
  last_update_login      NUMBER
);
```

These parameters have the same meaning as those defined for the individual Notes APIs.

The following table describes the fields associated with this record type.

jtf_note_contexts_tab_dflt Fields

Field Name	Data Type	Required	Descriptions and Validations
note_context_id	NUMBER		Note context identifier; this must be unique, if it is not, an error is raised. You must use the JTF_NOTES_S sequence to generate valid values.
jtf_note_id	NUMBER		JTF Notes identifier that refers to current note context.

Field Name	Data Type	Required	Descriptions and Validations
note_context_type	VARCHAR2 (240)		It is an object type code that is related to a given note. It is similar to p_source_object_code.
note_context_type_id	NUMBER		It is an identifier of the object related to a given note. It is similar to p_source_object_id.

Notes Public APIs

The following table describes the public APIs which are discussed in this chapter.

The Notes Public APIs

Procedure	Description
Secure_Create_Note, page 3-6	Creates a note (record) for a given business entity while applying appropriate security policies, and writes it to the database.
Secure_Update_Note, page 3-8	Updates an existing note (record) while applying appropriate security policies, and writes the updated version to the database.
Secure_Delete_Note, page 3-8	Deletes an existing note (record) from the database while applying appropriate security policies.

Note Source and Note Context

The following section details the relationship between both the note source and note context and the JTF_NOTES_PUB API package.

Note Source

The note source is always required when using the JTF_NOTES_PUB API package. The **note source** is a business object that owns the note. It is always one of the following:

- the creator of the note
- the initiator of the note
- the parent of the note

This could be a party, a service request, or an order, for example. In any case, the note does not make sense without the source. You must specify the source of a note when creating it.

Note Context

The **note context** is a business object that is related to the note, but is not its source. For example, a note is created for a task, but, in addition, it is related to a party, an opportunity and an employee. Note context is not required when you create a note.

The following table describes some of the important parameters related to both the note source and note context.

Note Source and Context Parameters

Parameter	Description
p_source_object_code	This parameter contains (references) the source business object. For example, this could be a party, a task, or a service request. This parameter is **not** the software module that created the note. For example, it is not Customer Care, or the Contact Center.
p_source_object_id	This parameter relates to the business object specified in *p_source_object_code*, but refers to a particular instance of the business object. For example, this could be the *party_id* for a party, or the *service_request_id* of a service request.
p_context	This is a short description field for the context business object.
p_note_context_type	This value is passed as a parameter in the internal call to the Create_Note_Context API. It is the business object that is related to, but is not the source of, the note. It is similar to *p_source_object_code*.

Parameter	Description
p_note_context_type_id	This value is passed as a parameter in the internal call to the *Create_Note_Context* API. It is similar to *p_source_object_id*.

The *p_note_context_type* parameter and the *p_note_context_type_id* parameter are stored as data items in the *p_jtf_note_contexts_tab* record. See "Record jtf_note_contexts_rec_type", page 3-2 section for details of this data structure.

Party Relationships

For notes related to a party business object (no matter what party type), use only the PARTY type definition. That is, do **not** use:

- PARTY_PERSON
- PARTY_GROUP
- PARTY_RELEATIONSHIP
- PARTY_ORGANIZATION

You must define the note as type PARTY to be able to find, across the entire Oracle E-Business Suite, any note created for a specific party.

> **Note:** It is not necessary to populate the source object into the notes context table. This action is performed automatically by the Notes APIs.

Note Contexts that Relate to a Customer

When a note is created for a source that contains customer information, then the customer information **must** be included as part of the context for that note. This ensures that all notes relating to that customer can be viewed in a single place. To do this, set the value for *p_note_context_type* to PARTY.

Note Contexts that Relate to a Relationship

If a note is created for a source that contains customer information that includes relationship information such as a member of an organization, then it must create three separate instances of *p_jtf_note_contexts_tab* records (discussed in "Record jtf_note_contexts_rec_type", page 3-2) that contain the following:

1. Information related to the organization in which the person works.

2. The party identifier for party of type "relationship." Do not use PARTY_RELATIONSHIP, instead use PARTY for the *p_note_context_type* parameter.

3. The party identifier (*party_id*) for the person.

Example of a Party Relationship

The following table lists information needed to specify the relationship of John Doe at Oracle (john.doe@oracle). It is stored in the PARTY tables in the database.

The Relationship of John Doe at Oracle

Party ID	Party Type	Party Name
1000	ORGANIZATION	Oracle
1100	PERSON	John Doe
1200	PARTY_RELATIONSHIP	John.Doe@Oracle

The following table lists the values for the context information needed to specify John.Doe@Oracle. The context business object is the p_note_context_type parameter. The context business object identifier is p_context_type_id.

Context Information for John Doe at Oracle

p_note_context_type	p_context_type_id
PARTY	1000
PARTY	1100
PARTY	1200

Secure_Create_Note

Use the Secure_Create_Note API to generate a text note. This API Performs the following tasks:

- It validates the note status, length, type, and ID.

- It applies appropriate security policies.
- It creates a note record and writes it to the database.
- It writes any related contexts to the database.

Example Note Parameter Values

The following table lists the input parameters associated with a note that was created and attached to service request 4039. This note was created as a private note, on September 2, 2000.

Note Parameter Value Example

Parameter	Value
p_parent_note_id	NULL
p_jtf_note_id	209
p_source_object_id	4039
p_source_object_code	SR
p_note_status	E
p_entered_by	1000194
p_entered_date	09/02/2000 12:42:22 PM
p_last_update_date	09/02/2000 12:42:22 PM
p_last_updated_by	1000194
p_creation_date	09/02/2000 12:42:22 PM
p_created_by	1000194
p_last_update_login	519018
p_note_type	AS_USER

Secure_Update_Note

Use the Secure_Update_Note API to modify an existing text note. This API Performs the following tasks:

- Validates the note status, length, type, and ID.
- It applies appropriate security policies.
- It update an existing note record and replaces the existing version in the database with the modified version.
- It updates any related context information in the database, if necessary.

Secure_Delete_Note

Use the Secure_Delete_Note API to modify an existing text note. This API Performs the following tasks:

- It deletes an existing note record in the database.
- It applies appropriate security policies.
- It deletes any related context information in the database.

Messages and Notifications

The APIs contained in package JTF_NOTES_PUB generate messages and notifications as needed.

> **Note:** Status notifications are not required to provide a number identifier along with the message, although, in many cases, it is provided.

JTF_NOTES_PUB

The following table lists the messages and notifications generated by the Secure_Create_Note, Secure_Delete_Note, and Secure_Update_Note APIs.

Create_Note and Update_Note Messages

Type	Name	Text
E	JTF_API_ERR_PRE_CUST_USR_HK	API Programming Error (&API_NAME): Returned Error Status from the Pre Customer User Hook.
E	JTF_API_ERR_PRE_VERT_USR_HK	API Programming Error (&API_NAME): Returned Error Status from the Pre Vertical User Hook.
E	JTF_API_ERR_POST_CUST_USR_HK	API Programming Error (&API_NAME): Returned Error Status from the Post Customer User Hook.
E	JTF_API_ERR_POST_VERT_USR_HK	API Programming Error (&API_NAME): Returned Error Status from the Post Vertical User Hook.
E	JTF_API_ALL_INVALID_ARGUMENT	API Programming Error (&API_NAME): The value of "&VALUE" for &PARAMETER is invalid.
E	JTF_API_ALL_MISSING_PARAM	API Programming Error (&API_NAME): Parameter &MISSING_PARAM is required and was not passed in.
E	JTF_API_ALL_NULL_PARAMETER	API Programming Error (&API_NAME): The value of &NULL_PARAM cannot be NULL.
E	JTF_API_ALL_VALUE_TRUNCATED	API Programming Warning (&API_NAME): The parameter &TRUNCATED_PARAM was truncated because the character value (&VAL_LEN) is longer than the defined width of the VARCHAR2 column (&DB_LEN).
E	JTF_UNABLE_TO_CHECK_FUNCTION	AOL Security Error: Unable to verify whether the function (&FUNCTION) is granted to this user or not. Contact your System Administrator.

Type	Name	Text
E	JTF_FUNCTION_NOT_GRANTED	AOL Security Error: function (&FUNCTION) is NOT granted to this user. Contact your System Administrator.
E	FND_LOCK_RECORD_ERROR	Unable to lock the record. Cause: The record is being modified by another user.

Sample Code

This section contains SQL scripts that call the Notes public APIs contained in the JTF_NOTES_PUB package and insert values as required.

Package JTF_NOTES_PUB

The SQL scripts in this section create and update a note by calling the Secure_Create_Note, Secure_Update_Note, and Secure_Delete_Note APIs in succession and by providing them with the required values.

Secure_Create_Note

This script calls the Secure_Create_Note API and creates a note with the following information:

- Source object ID: 42256
- Source object: Task Manager
- Note status: Public
- Note type: General
- Note text: These are the directions to my office

```
DECLARE
  l_api_version         NUMBER;
  l_init_msg_list       VARCHAR2(1);
  l_validation_level    NUMBER;
  l_commit              VARCHAR2(1);
  l_return_status       VARCHAR2(1);
  l_msg_count           NUMBER;
  l_msg_data            VARCHAR2(2000);
  l_jtf_note_id         NUMBER;
  l_source_object_id    NUMBER;
  l_source_object_code  VARCHAR2(8);
  l_note_status         VARCHAR2(8);
  l_note_type           VARCHAR2(80);
  l_notes               VARCHAR2(2000);
  l_notes_detail        VARCHAR2(8000);
  l_last_update_date    DATE;
  l_last_updated_by     NUMBER;
  l_creation_date       DATE;
  l_created_by          NUMBER;
  l_last_update_login   NUMBER;
  l_entered_by          NUMBER;
  l_entered_date        DATE;
  l_note_contexts       JTF_NOTES_PUB.jtf_note_contexts_tbl_type;
  l_msg_index           NUMBER;
  l_msg_index_out       NUMBER;
BEGIN
  -- Initialize the Notes parameters you want to create
  l_api_version         := 1.0;
  l_init_msg_list       := FND_API.g_true;
  l_validation_level    := FND_API.g_valid_level_full;
  l_commit              := FND_API.g_true;
  l_msg_count           := 0;
  l_source_object_id    := 42256;
  l_source_object_code  := 'TASK';
  l_note_status         := 'I';
  l_note_type           := 'GENERAL';
  l_notes               := 'These are the directions to my office';
  l_notes_detail        := 'These are the directions to my office';
  l_entered_by          := FND_GLOBAL.user_id;
  l_entered_date        := SYSDATE;
  -- Initialize Who columns
  l_last_update_date := SYSDATE;
  l_last_updated_by  := FND_GLOBAL.user_id;
  l_creation_date    := SYSDATE;
  l_created_by       := FND_GLOBAL.user_id;
  l_last_update_login:= FND_GLOBAL.LOGIN_ID;
  -- Initialize the Notes Context parameters (optional)
  l_note_contexts(1).note_context_type    := 'TASK';

  l_note_contexts(1).note_context_type_id := 42256;
  l_note_contexts(1).note_context_type    := 'TASK';
  l_note_contexts(1).last_update_date     := SYSDATE;
  l_note_contexts(1).last_updated_by      := FND_GLOBAL.user_id;
  l_note_contexts(1).creation_date        := SYSDATE;
  l_note_contexts(1).created_by           := 0;
  l_note_contexts(1).last_update_login    := FND_GLOBAL.LOGIN_ID;
   -- Call the API
  jtf_notes_pub.Secure_Create_note
  (  p_api_version           => l_api_version,
     p_init_msg_list         => l_init_msg_list,
     p_commit                => l_commit,
```

```
p_validation_level      => l_validation_level,
     x_return_status          => l_return_status,
     x_msg_count              => l_msg_count,
     x_msg_data               => l_msg_data,
     p_jtf_note_id            => l_jtf_note_id,
     p_entered_by             => l_entered_by,
     p_entered_date           => l_entered_date,
     p_source_object_id       => l_source_object_id,
     p_source_object_code     => l_source_object_code,
     p_notes                  => l_notes,
     p_notes_detail           => l_notes_detail,
     p_note_type              => l_note_type,
     p_note_status            => l_note_status,
     x_jtf_note_id            => l_jtf_note_id,
     p_last_update_date       => l_last_update_date,
     p_last_updated_by        => l_last_updated_by,
     p_creation_date          => l_creation_date,
     p_created_by             => l_created_by,
     p_last_update_login      => l_last_update_login
   , p_use_AOL_security       => 'F'
  );

  -- Check for errors
  IF (fnd_msg_pub.count_msg > 0)
  THEN
    FOR i IN 1..fnd_msg_pub.count_msg
    LOOP
      fnd_msg_pub.get
      ( p_msg_index     => i,
        p_encoded       => 'F',
        p_data          => l_msg_data,
        p_msg_index_out => l_msg_index_out
      );
      DBMS_OUTPUT.PUT_LINE('API ERROR: ' || l_msg_data);
    END LOOP;
  ELSE
    DBMS_OUTPUT.PUT_LINE('Created note : ' || to_char(l_jtf_note_id));
  END IF;
END;
```

Secure_Update_Note

This script calls the Secure_Update_Note API and updates the previously created note with the following values:

- Note status: E (Private)
- Note type: Call Back

```
DECLARE
l_api_version         NUMBER;
l_init_msg_list       VARCHAR2(1);
l_validation_level    NUMBER;
l_commit              VARCHAR2(1);
l_return_status       VARCHAR2(1);
l_msg_count           NUMBER;
l_msg_data            VARCHAR2(2000);
l_jtf_note_id         NUMBER;
l_note_status         VARCHAR2(8);
l_note_type           VARCHAR2(80);
l_notes               VARCHAR2(2000);
l_notes_detail        VARCHAR2(32000);
l_append_flag         VARCHAR2(1);
l_last_update_date    DATE;
l_last_updated_by     NUMBER;
l_last_update_login   NUMBER;
l_entered_by          NUMBER;
l_note_contexts       JTF_NOTES_PUB.jtf_note_contexts_tbl_type;
l_msg_index           NUMBER;
l_msg_index_out       NUMBER;
BEGIN
-- Initialize the API
l_api_version         := 1.0;
l_init_msg_list       := FND_API.g_true;
l_validation_level    := FND_API.g_valid_level_full;
l_commit              := FND_API.g_true;
l_msg_count           := 0;
--  Initialize Notes parameters you want to update
l_jtf_note_id         := 69944;
l_note_status         := 'E';
l_note_type           := 'ASF_CALLBACK';
l_notes               := 'These are the directions to my office';
l_notes_detail        := 'These are the directions to my office';
l_entered_by          := FND_GLOBAL.user_id;
-- Initialize Who columns
l_last_update_date := SYSDATE;
l_last_updated_by  := FND_GLOBAL.USER_ID;
l_last_update_login:= FND_GLOBAL.LOGIN_ID;
-- Initialize the Notes Context parameters (optional)
l_note_contexts(1).jtf_note_id          := 69944;
l_note_contexts(1).note_context_id      := 69945;
l_note_contexts(1).note_context_type_id := 42256;
l_note_contexts(1).note_context_type    := 'TASK';
l_note_contexts(1).last_update_date     := SYSDATE;
l_note_contexts(1).last_updated_by      := FND_GLOBAL.user_id;
l_note_contexts(1).last_update_login    := FND_GLOBAL.LOGIN_ID;
-- Call the API
jtf_notes_pub.Secure_Update_note
(  p_api_version           => l_api_version,
p_init_msg_list         => l_init_msg_list,
p_commit                => l_commit,
p_validation_level      => l_validation_level,
x_return_status         => l_return_status,
x_msg_count             => l_msg_count,
x_msg_data              => l_msg_data,
p_jtf_note_id           => l_jtf_note_id,
p_entered_by            => l_entered_by,
p_notes                 => l_notes,
p_notes_detail          => l_notes_detail,
p_append_flag           => l_append_flag,
```

```
p_note_type             => l_note_type,
p_note_status           => l_note_status,
p_last_update_date      => l_last_update_date,
p_last_updated_by       => l_last_updated_by,
p_last_update_login     => l_last_update_login,
p_jtf_note_contexts_tab => l_note_contexts
, p_use_AOL_security     => 'F"
);
-- Check for errors
IF (fnd_msg_pub.count_msg > 0)
THEN
FOR i IN 1..fnd_msg_pub.count_msg
LOOP
fnd_msg_pub.get
( p_msg_index     => i,
p_encoded       => 'F',
p_data          => l_msg_data,
p_msg_index_out => l_msg_index_out
);
      DBMS_OUTPUT.PUT_LINE('API ERROR: ' || l_msg_data);

END LOOP;
ELSE
DBMS_OUTPUT.PUT_LINE('Updated note : ' || to_char(l_jtf_note_id));
END IF;
END;
```

Secure_Delete_Note

This script calls the Secure_Delete_Note API and deletes the previously created note.

```
DECLARE
  l_api_version         NUMBER;
  l_init_msg_list       VARCHAR2(1);
  l_validation_level    NUMBER;
  l_commit              VARCHAR2(1);
  l_return_status       VARCHAR2(1);
  l_msg_count           NUMBER;
  l_msg_data            VARCHAR2(2000);
  l_jtf_note_id         NUMBER;
  l_msg_index           NUMBER;
  l_msg_index_out       NUMBER;
BEGIN
  -- Initialize the API
  l_api_version         := 1.0;
  l_init_msg_list       := FND_API.g_true;
  l_validation_level    := FND_API.g_valid_level_full;
  l_commit              := FND_API.g_true;
  l_msg_count           := 0;
  -- Initialize Notes parameters you want to update
  l_jtf_note_id         := 69942;

  -- Call the API
  jtf_notes_pub.Secure_Delete_note
  (  p_api_version           => l_api_version,
     p_init_msg_list         => l_init_msg_list,
     p_commit                => l_commit,
     p_validation_level      => l_validation_level,
     x_return_status         => l_return_status,
     x_msg_count             => l_msg_count,
     x_msg_data              => l_msg_data,
     p_jtf_note_id           => l_jtf_note_id,
     p_use_AOL_security      => 'F'
  );
  -- Check for errors
  IF (fnd_msg_pub.count_msg > 0)
  THEN
    FOR i IN 1..fnd_msg_pub.count_msg
    LOOP
      fnd_msg_pub.get
      ( p_msg_index     => i,
        p_encoded       => 'F',
        p_data          => l_msg_data,
        p_msg_index_out => l_msg_index_out
      );
      DBMS_OUTPUT.PUT_LINE('API ERROR: ' || l_msg_data);
    END LOOP;
  ELSE
    DBMS_OUTPUT.PUT_LINE('Delete note : ' || to_char(l_jtf_note_id));
  END IF;
END;
```

Index

C

Create_References, 2-3
Create Note, 3-3, 3-6
Create Task, 2-2, 2-5
Create Task Assignment, 2-2, 2-19
Create Task Reference, 2-23

D

Delete_References, 2-3
Delete Note, 3-3, 3-8
Delete Task, 2-2, 2-5
Delete Task Assignment, 2-3, 2-19
Delete Task Reference, 2-23

J

JTF_NOTES_PUB, 3-1
  Create Note, 3-6
JTF_TASK_ASSIGNMENTS_PUB, 2-19
JTF_TASKS_PUB, 2-5
JTF_TASKS_REFERENCES_PUB, 2-23

N

Notes
  JTF_NOTES_PUB
    Create Note, 3-3
    Data Structure Specifications, 3-2
    Messages and Notifications, 3-8
    Update Note, 3-8, 3-8
  Note Source, 3-3
  Party Relationships, 3-5

P

Parameter Specifications, 1-3, 1-6, 1-7
  Invalid Parameters, 1-8
  Parameter Validations, 1-8
  Standard IN Parameters, 1-3

R

References Record Type, 2-24

S

Sort Record Type, 2-6
Status Messages, 1-9
  Error, 1-9
  Success, 1-9
  Unexpected error, 1-9
  Warning and Information Messages, 1-10

T

Task Assign Record Type, 2-5, 2-6
Task Contact Record Type, 2-5, 2-15
Task Dates Record Type, 2-5, 2-14
Task Dependency Record Type, 2-5, 2-10
Task Manager, 2-2
  Data Structure Specifications, 2-5
    References Record Type, 2-24
    Task Assignment Record Type, 2-6
    Task Contact Record Type, 2-15
    Task Dates Record Type, 2-14
    Task Dependency Record Type, 2-10
    Task Recurrence Record Type, 2-12

Task Reference Record Type, 2-11
JTF_TASK_ASSIGNMENTS_PUB, 2-19
JTF_TASKS_PUB, 2-5
JTF_TASKS_REFERENCES_PUB, 2-23
Messages and Notifications, 2-26
Task Notes Record Type, 2-5
Task Recurrence Record Type, 2-5, 2-12
Task Reference Record Type, 2-5, 2-11

U

Update_References, 2-3
Update Note, 3-3, 3-8
Update Task, 2-2, 2-5
Update Task Assignment, 2-3, 2-19
Update Task Reference, 2-23