Oracle® Fusion Middleware

Developing Applications with Oracle Team Productivity Center

12*c* (12.1.2)

E23016-01

June 2013

Documentation for developers that describes the Oracle Team Productivity Center functionality available in the IDE, on the administration of Team Productivity Center, and on creating connectors for use with Team Productivity Center repositories.

ORACLE

Oracle Fusion Middleware Developing Applications with Oracle Team Productivity Center, 12*c* (12.1.2)

E23016-01

Primary Author: Scott Fisher

Contents

2 Working with Oracle Team Productivity Center

3 Working with Oracle Team Productivity Center Administration

4 Working with Oracle Team Productivity Center Connectors

Preface

Welcome to the *Developing Applications with Oracle Team Productivity Center*.

Audience

This document is intended for developers who intend to use Oracle Team Productivity Center with Oracle JDeveloper, and provides detailed information on the functionality available in the IDE, on the administration of Oracle Team Productivity Center, and on creating connectors for use with Oracle Team Productivity Center repositories.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

- *Installing Oracle Team Productivity Center Server*
- *Java API Reference for Oracle Team Productivity Center*
- *Tag Reference for Oracle Team Productivity Center Connectors*
- *Installing Oracle JDeveloper*
- *Developing Applications with Oracle JDeveloper*
- *Developing Extensions for Oracle JDeveloper*
- *Oracle JDeveloper 12c Online Help*
- *Oracle JDeveloper 12c Release Notes*, link included with your Oracle JDeveloper installation, and on Oracle Technology Network

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in This Guide

The following topics introduce the new and changed features of Oracle Team Productivity Center and other significant changes that are described in this guide, and provides pointers to additional information.

New and Changed Features for 12*c* (12.1.2)

Oracle Team Productivity Center 12*c* (12.1.2) includes the following new and changed development features for this document:

- Code review, which includes a set of features that permit team members to share modified files for review and approval by selected team members. See Section 1.5, "Understanding Code Reviews" and Section 2.3, "Working with Code Reviews".
- News feed selection, which allows you to select and filter news feeds to be displayed in JDeveloper. See Section 2.9, "Working with the News Panel."
- Lightweight Directory Access Protocol (LDAP), which simplifies the administration of team members. See Section 3.4, "Working with Users, Teams and Roles."

1

Introduction to Oracle Team Productivity Center

This chapter describes the basic operations of Oracle Team Productivity Center, including logging in, connecting and disconnecting, selecting your team, and using and interpreting the build summary.

This chapter includes the following sections:

1.1 About Oracle Team Productivity Center

Oracle Team Productivity Center is a Oracle JDeveloper feature that provides access to a set of application lifecycle management (ALM) tools and technologies.

ALM tools address the need to integrate the growing number of data tracking and data repository tools that are used by product development organizations throughout the product's lifecycle. Initial product development, for example, typically tracks product definition, specifications, and collaborative code development, as well as defects discovered during initial testing. As the product is released, customer requests and enhancements may be tracked in a third repository. When developers are later required to maintain a customer-released product in one development branch while also doing new feature development for upcoming releases in a different branch, the request and defect-tracking repositories need to be linked with the appropriate development branch, whether maintenance or new release, of the team's version-control system. Oracle Team Productivity Center lets teams integrate all these repositories, and any others they may use, directly in the integrated development environment.

In addition to bringing external data repositories into the IDE, Oracle Team Productivity Center supports a highly flexible hierarchy of teams and subteams that can be configured and maintained by local team administrators. Oracle Team Productivity Center is designed to:

- Increase value in existing investments by providing unified access to current artifact repositories (including bug databases, version repositories, and other content libraries) and allowing integration between them.

- Allow for quick and flexible assembly of teams and subteams.
- Collaborate with other team members through chat.
- Reduce the barriers between different teams during the development process, and eliminating many hard-wired boundaries through collaboration, shared labeling, and smooth information flow.
- Maximize current investments in skills, processes and technologies.
- Increase flexibility by reducing the time it takes to integrate new applications into the development team environment.

Oracle Team Productivity Center also presents information about the latest builds undergoing development. A build dashboard and build summary present at-a-glance information about the status of development builds, with links to let you quickly find the details of issues that interest you in the latest build.

1.1.1 How to Migrate from a Previous Version

When you install the latest version of Oracle JDeveloper containing Oracle Team Productivity Center, you have the option of migrating files from a previous version. If you have already been using Team Productivity Center and are upgrading to a new version, be sure to follow the installation instructions for migrating to the latest version. In particular, be sure to install and migrate to the newest versions of any connectors you use. This will ensure that you have the latest capabilities, and that your connectors are fully compatible with the latest Team Productivity Center client.

1.2 Connecting to Oracle Team Productivity Center

Oracle Team Productivity Center is made up of the following parts:

- The Oracle Team Productivity Center features of Oracle JDeveloper.
- The Oracle Team Productivity Center server software, which is available from Oracle Technology Network. The server software includes a standalone installation guide, also available from Oracle Technology Network, which includes help on installing the Oracle Team Productivity Center server and connectors
- Oracle Team Productivity Center connectors, which are available by selecting **Help > Check for Updates**. Connectors permit the Oracle Team Productivity Center client software, running inside JDeveloper, to interact with data repositories such as bug-tracking systems and feature databases.
- The Team Resource window, from which you can create connections to shared resources such as an application server, a database, or a file system, which will then be available to all team members.

To connect to the Oracle Team Productivity Center server:

- In the Teams accordion, click on **Connect to Team Server**, or select **Team > Connect to Team Server**.

This opens the Connect to Team Server dialog, described in the following section.

For more information on installing connectors from Check for Updates, see the “How to Install Extensions with Check for Updates" section in *Developing Applications with Oracle JDeveloper*.

1.2.1 How to Connect to the Team Productivity Center Server

Connecting to the Team Productivity Center server for the first time requires some information which can be provided by your administrator, displayed in the Connect to Team Server dialog.

Table 1–1 Connect to Team Server Dialog Options

Option	Description
Team Server	The machine name or IP address of your Oracle Team Productivity Center Server.
Port	The dedicated port used by your Team Server.
Use SSL	Select this if your Team Server connection requires a secure socket layer connection.
Username	The username assigned to you by the Oracle Team Productivity Center administrator.
Password	The password assigned to you by the Oracle Team Productivity Center administrator. To change this password, select **Manage Profile** after logging in with your existing password.
Remember Password	Select this to store your username and password, with other server information, for future logins.
Connect When JDeveloper Starts	With this option selected, you will be connected to Oracle Team Productivity Center every time you open JDeveloper.

When you have entered all the information and made all your selections, click **Connect**.

JDeveloper will store these settings. Note that if you select Remember Password, the password will be encrypted.

1.2.2 How to Disconnect From Oracle Team Productivity Center

You can disconnect from Team Productivity Center at any time.

To disconnect from Team Productivity Center:

1. Click the dropdown on the right-hand side of the currently selected team.
2. Select **Disconnect** from Team Server.

> **Note:** Team Productivity Center disconnects automatically when you quit JDeveloper.

You can also disconnect from Team Productivity Center by selecting **Team > Disconnect from Team Server**.

1.2.3 How to Manage Your Login Profile

Your Oracle Team Productivity Center user profile contains your first and last name, your email, your Oracle Team Productivity Center password, your preference for connecting to Team Productivity Center, and your code review settings. On the Login tab of the Manage Profile dialog, you can also set Oracle Team Productivity Center to remember your password on future connections.

To set values in your login profile:

1. Select **Team > Manage Profile**.
2. Update any of your user information in the name and email fields in the upper part of the dialog.
3. To change your password, enter your old password, then enter your new password twice, once in the **New Password** field, then a second time to confirm in the **Retype Password** field.
4. If you would like Oracle Team Productivity Center to remember your password for future connections, select **Remember Password**. Note that you can only change the Remember Password settings if you have changed your password.
5. Click **OK**.

1.2.4 How to Manage Your Code Review Profile

On the Code Review tab of the Manage Profile dialog, you can also set Oracle Team Productivity Center to remember your preferences for how you wish to be notified of code review activity, and how you prefer to view code review content. You can choose to be notified by email or by popup when various conditions occur regarding your participation in code review.

To set values in your Code Review profile:

1. Select **Team > Manage Profile**.
2. Place a check mark in the box under Email or Popup for each of the conditions
3. To change your password, enter your old password, then enter your new password twice, once in the **New Password** field, then a second time to confirm in the **Retype Password** field.
4. If you would like Oracle Team Productivity Center to remember your password for future connections, select **Remember Password**. Note that you can only change the Remember Password settings if you have changed your password.
5. Click **OK**.

1.2.5 How to Manage Your Repository Accounts

When using Oracle Team Productivity Center, you will often be accessing information stored in data repositories, many of which require you to have a user account for access. For example, if you have an account in your organization's bug tracking database, you need to provide this information to Oracle Team Productivity Center so that you can view, edit and save bug report information inside Oracle Team Productivity Center.

Oracle Team Productivity Center can store your account information (user ID and password) to simplify access to these accounts while working. For more information about work item repositories, see Section 2.2, "Working with Work Items."

To manage your repository accounts:

1. From the Team Navigator menu, select **Manage Accounts**. The Team Navigator menu is the drop-down box at the right of the Team Productivity Center window.
2. From the left-hand pane of the dialog, select the work item repository for which you need to change your account information.

3. In the right-hand pane of the dialog, enter the up-to-date information for the account associated with the selected repository.
4. You can optionally verify that your user ID and password are correct for the account you are editing. To do this, click **Test Connection**. The result of the test is displayed in the Status box.
5. When you have verified that the information is correct, click **OK**.

1.3 Navigating Oracle Team Productivity Center

Navigating Oracle Team Productivity Center involves viewing team members, browsing versioning repositories, viewing or querying work items, creating work item templates, and participating in code reviews. You can also select your team and chat with team members.

You navigate Team Productivity Center through the Teams window, which opens beside the Applications window once you connect to Team Productivity Center.

Within the Teams window, additional accordions display the various components and interfaces of Team Productivity Center:

At the very top of the Teams, Oracle Team Productivity Center displays a drop-down list of teams available to you. Once you select a team from this list (unless there is only one default team, in which that team is selected automatically), the three accordion panels below display data for the selected team and its members. If you change to a different team, the accordion panels will be refreshed with the current team information.

Table 1–2 Team data available through Team Navigator

Name	Description
Team Members	Displays the members when you select a given team.
Work Items	Displays work item repositories connected for the selected team. Viewing a work item for the first time in a session may require connecting to that work item's repository with your username and password for the repository. After you connect to a repository, you can create queries to that repository. For more information about work items and repositories, see Section 2.2, "Working with Work Items."
Versioning	Displays the connected versioning repositories for the team. You can check out files from your versioning repository to your project/application (depending on the versioning system). You can view versioned files in read-only format. You can perform some repository administration tasks (such as create new directories), depending on the versioning system.

1.3.1 How to Select Your Team Productivity Center Team

When you connect to your Team Productivity Center server, Oracle Team Productivity Center displays a list of the teams to which your administrator has added you.

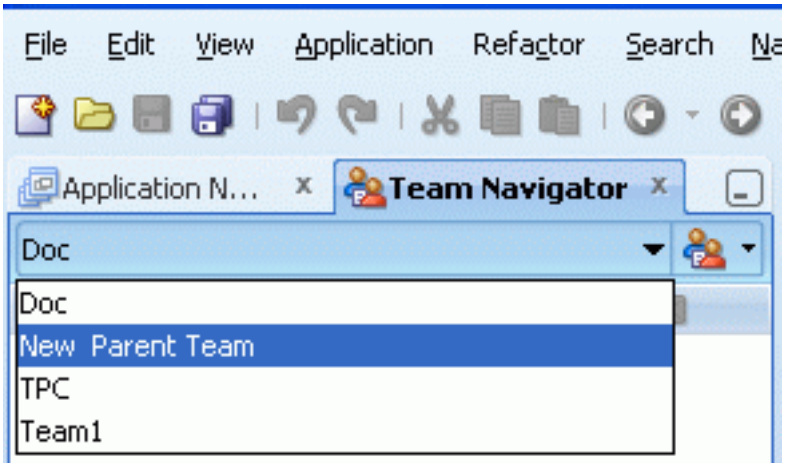
Once you have signed into Team Productivity Center, you can select from any available teams. (In some cases, you may only be a member of a single team, in which case no selection options will appear.)

To select a team:

1. Click on the dropdown list just below the Team Navigator title.

2. Select a team.

Figure 1–1 Selecting a team with the Team Navigator



You will notice that a list of team members including yourself appears. Below the Team Members accordion, you will see the Work Items and Versioning accordions. Each of these will provide access to information repositories needed by your team.

Oracle Team Productivity Center indicates whether or not a team member is connected to the chat server; a circle with slash indicates they are not connected. In addition, rolling the cursor over a team member's name displays that member's data: full name, their role in the team currently being displayed, and their email address. Click the right mouse button on the team member's name to display the following options:

Send Email

Opens a message in your system's default mail tool.

Chat

Initiates chat with the member. If you have not yet selected a chat application or server, Oracle Team Productivity Center displays the Connect to Chat Server dialog.

If you are signed in to your chat server, double-clicking a team member's name will initiate a chat window. If you are not signed in to your chat server, double-clicking opens an email window.

1.3.2 How to Use Team Productivity Center Chat

Once you have installed Team Productivity Center, you can chat with members of your team. Team Productivity Center chat incorporates real-time text chat inside JDeveloper. You can chat with your selected team members, or with buddies from your active XMPP/Jabber protocol.

Once you have installed Team Productivity Center, you can use chat by selecting **Team > Chat** at any time. You don't have to be connected to Team Productivity Center to do so: you can open the chat window at any time, regardless of whether you are connected to Team Productivity Center. However, you cannot start a chat from the Team Productivity Center navigator if you aren't connected.

To connect to Oracle Team Productivity Center chat:

- Select **Team > Connect to Chat Server**.

You can initiate a chat session in two ways:

- After selecting a team member, click the chat symbol adjacent to the Team Members accordion header.
- Double-click on the name of any team member.

1.3.2.1 Logging into the Chat Server

Oracle Team Productivity Center only supports XMPP/Jabber protocols, and does not support multiple simultaneous connections. The first time you log into the Team Member chat panel, you may be asked to provide some configuration information needed by Oracle Team Productivity Center. Table 1–3 contains the information that is typically included.

Table 1–3 Configuration Options

Option	Description
Host	The name of the host chat server URL.
Port	The assigned port number.
Domain	The assigned domain.
Resource	JDeveloper (typical). This is a user-defined name which can be unique, for example, if you are logged into a chat server from multiple JDeveloper instances.
User Name	Your chat server user name.
Password	Your chat server password.
Remember Password	Select to store your chat server password for future logins.
Connect when JDeveloper starts	(dependent on chat server configuration) Depending on the configuration of your chat server, the last two options may or may not work automatically. In addition, two buttons in the Chat pane toolbar provide a shortcut to connecting or editing the connection properties:
Connect	Connects to chat server. Uses same fields as Edit Connection Properties. Once you are connected, the Chat window displays two buttons on the toolbar of a chat that is in progress: Send, which sends the chat message, and Clear, which deletes all the text.

Once you are connected to the Chat server, JDeveloper displays an option to Edit Connection Properties, which includes a button to re-login to the chat server and an empty password field.

In addition, the Chat window contains four other buttons for controlling the chat:

Table 1–4 Chat window control buttons

Button	Description
Disconnect/Connect	When connected, clicking this button disconnects from the current chat server. When disconnected, this button opens the Connect to Chat Server dialog.

Table 1–4 (Cont.) Chat window control buttons

Button	Description
Show	Displays a drop-down menu with the following options: Show Offline Buddies: Display chat buddies who are offline as well as those on line and available for chat. Show Empty Groups: Display groups that have no members. Show Details: Display chat details. Add Group: Create a new group. Add Buddy: Add an available buddy to the chat.
Status	Displays a drop-down menu from which you can select from the server's available list of status messages (for example, Away, Available, In a Meeting, and others depending on the chat server). The message you select is displayed for other team members who view your icon in their chat window.
Edit Connection Properties	Opens the Edit Connection Properties dialog.

You can only connect to one chat server at a time. If your team uses multiple chat server, you must disconnect from one and update your chat properties to connect to another. To do this, click the **Edit Connection Property** button, or click the **Connect** toolbar button. Any of these will open the Connection Property dialog, from which you can click the Connect button.

1.4 Understanding the Build Dashboard and Build Summary

The Build Dashboard and Build Summary summarize the build status of projects your organization is tracking with a build tracking plugin such as Hudson or Cruise Control.

The Build Summary is displayed when you first log in to Team Productivity Center. JDeveloper shows a temporary snapshot of the Build Summary, containing information about the development/build branches your team is following, with status listings from the most recent builds and links to other results.

The Build Dashboard presents detailed statistics from builds of projects your organization is tracking with Team Productivity Center. The upper portion of the summary gives an overview of build information, test results, and the total transactions represented in the build. The lower portion shows related transactions from the source control management system in use, if Oracle Team Productivity Center was used to commit to the source control management system. You can open the build summary for a specific build listed in the Build Dashboard to view additional details on that specific build.

1.4.1 How to Open and Use the Build Dashboard

If the Build Dashboard is not visible, you can access it from the Team menu. This gives you easy visual access to the builds, branches, and test results that your team is following at this point in the project lifecycle.

To open the Build Dashboard:

- **Team > Build Dashboard**

You can also open the Build Dashboard by clicking the drop-down selector from the Teams window, then selecting Build Dashboard.

If you have not yet logged in to Oracle Team Productivity Center and you select the Build Dashboard, you will be asked to log in to Oracle Team Productivity Center first. For more information, see Section 1.2.1, "How to Connect to the Team Productivity Center Server."

1.4.1.1 Using the Build Dashboard

The Build Dashboard shows statistics from build status results that your team is tracking. You use the Build Dashboard to view transactions from your source control management system, test results, and work items affected by the build.

To use the Build Dashboard:

Clicking the branch name or the Total Transactions links of a row in the Build Dashboard displays the build results. This shows statistics from builds of projects your organization is tracking with Team Productivity Center.

Once you display the Build Dashboard, you can select from the latest builds and then view the information shown in Table 1–5.

Table 1–5 Build Dashboard Options

Option	Description
Branch	Click the dropdown to select the development branch you are interested in.
Build	The sequential list of builds in the selected branch. Click on the build of interest to display the Build Summary for the selected build. Below the list of builds in a branch, the Build Dashboard displays a list of related transactions for the selected build in the selected branch.
Related Transactions	The Build Dashboard displays the Change Set, Change Set Owner, Repository, Type, and Key for the selected branch. Of these, the Change Set and Key can be selected; the other display elements are read-only.
Change Set	The change set in the selected versioning system. Selecting an individual change set loads it into the Log window and displays elements assigned to the change set you have selected
Key	Displays the work item ID associated with the change set being displayed. Clicking on the key opens the work item detail editor, allowing you to view and modify the work item.

1.4.2 How to Use the Build Dashboard and Build Summary

All build/test system configuration takes place when the plugin is installed in the build/test environment. The build plugin is typically installed when Oracle Team Productivity Center Server is installed initially. After installation and configuration, build/test system results are returned to the Team Productivity Center client and made available to the Build Dashboard. You can customize the dashboard to display the results that are of interest to you.

To customize the build dashboard display:

- Click on the **Customize Branch and Favorite Notifications** icon (the wrench).

This opens the Manage Module and Display Notification dialog. From this dialog you can select the modules you wish to include in the dashboard, as well as whether you choose to subscribe to notification in JDeveloper when any new build results are

available for one of these modules. For more information while using the Manage Module and Display Notification dialog, press F1 or click **Help** at any time.

1.4.2.1 Selecting Builds from the Build Dashboard

The Build Summary presents statistics from one specific build your organization is tracking with Team Productivity Center. The upper portion of the summary gives an overview of build information and test results represented in the build.

To select a build from the Build Dashboard:

- Click on the **Branch** drop-down from the Build Dashboard, then double-click on the build of interest to select it. This displays the Build Summary for the selected build.

Table 1–6 describes the Build Summary toggle buttons, by which you can control which tests are displayed.

Table 1–6 Build Summary Toggle Buttons

Option	Description
Show	Select the statistics that you want the Build Summary to display. Clicking on one of these icons toggles it, choosing to display or hide results of the type you select:
Error	Show/hide the results of tests that contain errors.
Failed	Show/hide the results of tests that failed.
Passed	Show/hide the results of successful tests.
	Next to the statistic drop-down list, the Build Summary gives you the following ways of altering what information is displayed.
Favorites	Show/hide test results marked as favorites.
Unassigned	Show/hide selected results with no assigned owner.
Filter	Display selected results filtered by category and by a string. Click the binocular drop down, then select the area from which you wish to filter the results. You can choose to filter results from one of these categories: Module, Class, Test case, Change Set Owner, or Work Item.
	For example, when you have selected the category, type a string you wish the Build Summary to use as a search filter. The Build Summary displays only results that match the filter string.
Expand/Collapse	You can expand and collapse any of the individual results in the Build Summary by clicking the node icon at the left of the results in the Class column. You can also expand and collapse all results by clicking the Expand All (the double-plus sign icon) and Collapse All (double-minus sign icon) buttons next to the Filter field.
	Below the overview, the Build Summary provides detailed information about the module, class and tests performed, as well as links to the repositories your team is using to track bug information. The summary contains the following columns:

Table 1–7, " Build Summary Columns" lists the columns displayed in the Build Summary, containing the results of the selections you made with the Build Summary toggle buttons:

Table 1–7 Build Summary Columns

Option	Description
Test	The name of the test class run on the build. You can expand or collapse the hierarchical view of members in this class by clicking on the + or - sign at the left of the class name, or you can expand and collapse all classes by using the Expand All and Collapse All icons at the top of the display.
Tests Run	The number of tests run in each class. If you expand the class, the summary displays the number of tests run for each member in the class as well as the total number of tests run in the class
Failures	The number of tests in each class that resulted in a failure.
% Passed	The percentage of successful tests. Note that the percentages in each row reflect the ratio of passes to failures for each member of the class; if you collapse a row (which suppresses the display of elements in, and beneath, that class member), the percentage passed may not appear to reflect the visible numbers until you expand the row.
Status	A graphical display representing the status of each test, as a success, a caution, or a failure.
Results	The Results column displays a link to a detailed screen of information which summarizes the results of a test, including the specific error messages or output produced by that test. For failed test cases, this column normally displays more data, such as a full stack trace. Click the link to display a read-only dialog with full data.
Owner	The assigned team member responsible for the test on the selected row. You can send the owner email or open a chat session with the owner by right-clicking the owner's name. If no owner is assigned to a test, the owner field contains a button you can use to assign yourself as the owner. Tests without an assigned owner will be the only items displayed if you select the Unassigned Only checkbox. Additionally, this cell contains a toggle button: **Add me as a owner** and **Remove me as a owner**. To take ownership of a failed test, click **Add me as a owner**. To relinquish ownership, click **Remove me as a owner**.
Work Items	The numbers of any Oracle Team Productivity Center work items assigned to the test. Click on the work item number to display the details of that work item. To show all repositories the team can access, click **Create Work Item**. This displays a list of all the work item repositories available to that team. You can then create the appropriate work item for that test. Once the new work item is saved, the build summary page is updated with the new work item number.

1.5 Understanding Code Reviews

Oracle Team Productivity Center contains a number of features that allow a team to review code collaboratively, as part of the development process. The Code Reviews window is the primary way you access code reviews. While there are other ways you can search for, edit, and interact with code reviews, an overview of the Code Reviews window will give you an understanding of how to interact with code reviews.

1.5.1 How to Access the Code Reviews Window

You must be logged in to your Oracle Team Productivity Center server to access the Code Reviews window.

To access the Code Reviews window:

- Select **Team > Code Reviews**.

This opens the Code Reviews window, located by default next to the Teams window.

1.5.2 How to Navigate the Code Reviews Window

The Code Reviews window lets you browse through reviews you have already seen, created, or been assigned. It also includes four tabs that let you display comments made to a review, the details for that review, reviewers assigned to it, and attachments for the selected review.

At the top of the Code Reviews window, you will see a drop-down list box from which you can select from three folders:

- Reviews You Created
- Reviews Assigned to You
- Recently Viewed Reviews

Select one of these folders to display its contents; you will be able to select a specific review from the contents of each folder. If the folders are empty, you can search for reviews. Searching for reviews is covered in detail in Chapter 2, Section 2.3, "Working with Code Reviews"

To the right of the drop-down list box, there is an additional drop-down menu selector which displays the following options:

- New Review: Creates a new review.
- Refresh Review List: updates the content of the Code Reviews window if there have been changes (for example, if new reviews have been assigned to you).
- Mark Review Closed: Close the selected review. Closed reviews are no longer available for comments.
- Rename Review: Opens the Rename Review Name dialog. Enter the new review name, then click **OK**.
- Search For Reviews: opens the Search Code Reviews window. Searching for reviews is covered in detail in Chapter 2, Section 2.3, "Working with Code Reviews"

Below the drop-down lists, you will see four tabs that present more detailed information for a selected review:

- Comments
- Details
- Reviewers
- Attachments

These tabs give you access to different aspects of the Code Review.

1.5.2.1 Using the Comments Tab

The Comments tab displays two kinds of comments:

- General comments: these are comments meant to be applied to the overall review, rather than comments about a specific line of code or change made to a file under review.
- Files with comments: below the general comments, JDeveloper displays a list of files associated with this review, which have comments.

The list of files for a specific review lets you explore and interact with the files that your team have commented on in the process of the review.

To add a general comment:

- Select the Comments tab, then click the Add New General Comment icon. Enter the text of your comment, then press the Enter key when completed to save your comment.

To select a file from the Comments tab:

- Double-click the file name in the Comments tab.

This opens the selected file in the JDeveloper editing window, using the revision compare tool. This lets you scroll through two versions of the file being reviewed; you can use the navigation icons to jump to the next change, previous change, first change or last change.

You can reply to comments and also create new comments. See the section Section 2.3.5, "How To Comment on a Code Review" for details.

1.5.2.2 Filtering Review Comments

There are two ways you can filter which files and which review comments are displayed:

- By using the filter field to restrict display to comments matching the filter text.
- By selecting an option from the Show menu (funnel icon) to apply to all available review comments.

The filter field interactively displays only the comments that match the text you enter. For example, if you have comments on Line 4, Line 6, and Lines 8-12, typing the text “Lines” in the filter field causes only the comment on Lines 8-12 to be displayed.

Note that the filter field only applies to the titles of comments; it has no effect on the display of files associated with the review.

To determine which files are to be displayed, you can use the Show menu to select specific conditions under which files, and any comments they contain:

- Show All Files: Display all files associated with this review.
- Only Show Files With Comments: Suppress the display of any files that do not contain review comments.
- Show All Comment Threads: Display all comment threads, including the initial comment and any subsequent responses.
- Show Only Unresolved Comment Threads: Suppress the display of any comment threads that have been resolved.

Careful use of the filter field and the Show menu can help you focus on specific issues of concern to you, rather than having to search individually through all comments in a specific review. This can be particularly valuable in a complex, lengthy project with dozens or even hundreds of review comments.

1.5.2.3 Using the Details Tab

Table 1–8 illustrates the contents of the Details Tab

Table 1–8 Contents of the Details Tab

Option	Description
Name	The name of the code review you have selected.
Created By	The name of the team member who created the code review.
Revision Details	Contains the following information about the file revision being reviewed.
Notes	Displays any notes made by the initial creator of the code review.
Changeset Details	Displays the files included in the changeset to which this code review applies.
Associated Work Items	If this code review pertains to specific work items, they are included here. You can view the work item by double-clicking its name in this field.

1.5.2.4 Using the Reviewers Tab

The Reviewers tab lists all team members who have been added as reviewers.

If you are the owner of a review, you can add or delete team members to the reviewers list by clicking the appropriate icon at the top of the tab.

To add or remove a reviewer from your review:

- Click on the Add Reviewers (+) or Delete Reviewers (X) icon.

In addition, team member names in the Reviewers tab are active links to the chat feature. You can send a message to any of the reviewers in the tab by clicking their name.

1.5.2.5 Using the Attachments Tab

The Attachments tab lists any files added to the review. You can add or delete files to the review.

To add or remove a file from the review:

- Click on the Add Files (+) or Delete Files (X) icon.

1.5.2.6 Managing Your Review Preferences

You can control the way you receive notifications, and the number of reviews displayed in the Recently Viewed Reviews folder, by using the Reviews tab of the Manage Profile dialog.

To control how you receive notifications:

- From the Teams drop-down menu, select **Manage Profile**, then select the **Code Reviews** tab.

The Code Reviews tab lets you select whether you prefer to be notified by email or by popup under various conditions. ou can also select how many code reviews to be displayed in your Recently Viewed Reviews folder (5, 10, or 15), and you can clear the list of recently viewed reviews.

2

Working with Oracle Team Productivity Center

Work items are a representation of any Team Productivity Center artifact that a person can perform work on. Typical examples include program defect records, requirement database entries, test cases, project tasks, and use cases, but work items are not limited to these. Any Team Productivity Center artifact can be represented and displayed if a suitable connector exists.

This chapter includes the following sections:

2.1 About Working with Oracle Team Productivity Center

The Oracle Team Productivity Center Work Items accordion contains nodes that connect teams to their information repositories, such as bug tracking databases, systems that log feature requests, project tracking information, and many other types of shared technologies that are used to track major projects.

Work items are the mechanism that Team Productivity Center uses to represent individual elements of the information repository. For example, the work items in a bug-tracking database would be individual bug reports; a feature repository's work items would be the individual feature requests tracked in that repository.

Work item nodes typically include such product development tools as:

- Feature request tracking repositories, such as Atlassian's JIRA.
- Bug tracking databases, such as Bugzilla.
- Project tracking software, such as Microsoft Project Server.
- HPALM to support quality assurance tests.

- Oracle Team Productivity Center's own Task Repository, for tracking the status, priority, and ownership of projects within Team Productivity Center.

Through Oracle Team Productivity Center, you can save context, make an active item and add details of source-control system checkins. You can create queries against these repositories and then build upon this information by creating relationships and assigning tags to the various work item elements that are important to the current part of the development process.

In addition, each work item has its own set of relationships, which it maintains to other work items both in its own repository (that is, of its own type) and, optionally, with work items of other types. For example, a bug database entry might be linked to the requirements tracking system; a feature enhancement request might link to a team milestone chart. These relationships will vary depending on the nature of the work and of the specific repositories you use with Team Productivity Center.

2.1.1 How to Support an Information Repository for Oracle Team Productivity Center

Here are the requirements for a particular information repository to become available as a work item node:

1. An administrator needs to add the Team Productivity Center connector to the server.
2. The end user needs to install and add the same Team Productivity Center connector as an extension to their copy of JDeveloper.
3. Access to the repository has been set up for your team.
4. The repository must be on line.
5. If you have not previously done so, depending on the repository's login requirements, you will need to log into the repository through the Oracle Team Productivity Center Manage Accounts dialog. For more information on connecting to a repository, see Section 3.3, "Making Repositories Available in Oracle Team Productivity Center."

Work items can be thought of as records, fields, or simply items of information returned from repositories in response to Oracle Team Productivity Center queries.

Typically, work items returned by a query are presented in a selectable, tabular format, as determined by the Oracle Team Productivity Center Connector.

The range of what you can do with information returned from the underlying repository is determined by the design of each repository's connector. In a typical use case, you can double-click to edit a work item, and then interact with the work item in ways defined by the connector author. It is recommended that connector developers duplicate the functionality and user interface of the repository as far as possible when developing a connector for use inside JDeveloper, providing the team with a consistent user experience and the functionality within the repository that the team desires. For more information on developing connectors, see Chapter 4, "Working with Oracle Team Productivity Center Connectors."

2.1.2 Selecting Repositories for Team Use

In addition, Team Productivity Center users will need to make sure they have downloaded the connectors for the repositories your team uses. You can download these from **Help > Check for Updates**.

The Team Administrator can add repositories, based on the installed connectors. You may find it useful to add a different server that points to the same repository, a

common situation if your team is distributed throughout the world. Adding a local server for different branches of your organization can improve performance and efficiency for your organization.

To select repositories for team use:

- **Team Navigator > Team Administration > Teams > Team Repositories**

Most administrative tasks involving repositories begin from the Team Administration dialog. You access this dialog by selecting **Team Navigator > Team Administration > Repositories**. The Team Administration dialog lets you add, remove, and modify repositories.

2.2 Working with Work Items

Work items are elements of a data repository that encapsulate the data you are tracking in that repository (such as a bug report, which encapsulates the bug description, version numbers, and other information important to your product lifecycle management methodology).

Another way of looking at work items is to consider them the result of a query you have made to a data repository. Once Team Productivity Center returns the work item, you can interact with it in a number of ways.

2.2.1 How to Connect to a Repository from Team Productivity Center

Note that your team may have one or more repository servers for the same Repository, with different values in the parameters. For example, there may be two mirror sites for the repository, based on geographical preference. In addition, after a corporate merger, the two companies might use the same repository and different configurations. Users can choose which server to connect to through the Manage Accounts dialog when they log in, but still maintain access to the same data repository.

You typically connect to a repository by double-clicking

To connect to a repository:

1. Double-click the repository in the Work Items accordion. If your repository login credentials are correct, JDeveloper connects you automatically.

If your repository login credentials are not correct, JDeveloper will notify you that it cannot access the repository and will then prompt you to go the Account Manager to correct the problem. From the Account Manager you can update your login credentials.

To update your repository login credentials:

1. Select one of the repositories under the Accounts column.
2. From the Repository Server drop-down, select the preferred server for the repository you selected in the preceding step.
3. Enter the login credentials (user name and password) for the server associated with this repository, and then click **OK**.

   You can also check the connection to the repository by clicking **Test Connection**. This will allow you to verify the connection data and make changes if required.

2.2.2 How to Access Work Items through a Team Query

Team Productivity Center uses queries to access work items in the selected repository. Queries allow you to specify elements of the work item (for example, the bug status or range of ID numbers in a bug database repository) to return just the work item or range of work items you are interested in.

A team query is a query which has been set up by your administrator to be available to all members of your team. In addition to team queries, Team Productivity Center also lets you create user queries. For more information, see Section 2.6, "Working with Queries."

Once you are logged in, you use these queries to access individual work items from the available repositories. When you first begin working in Team Productivity Center, you will most likely not yet have created any user queries, but you might have team queries available. The following procedure assumes that your team administrator has saved a number of team queries for one or more of your repositories.

Before you can use a particular Work Item node, you need to log into the repository through the Manage Accounts dialog. If you have not logged into the repository, Team Productivity Center will instruct you to go to the Manage Accounts dialog if you try to access a work item before being logged in. Once you are logged in to a repository, JDeveloper displays a small yellow in the upper right-hand corner of the repository icon.

To access Work Items through a team query:

1. Click on the + next to the repository's name to expand its listing in the Work Items accordion
2. Click on the + next to Team Queries to expand the listing.
3. Double-click on one of the team queries. JDeveloper displays the result of this query in the central pane.

Depending on the structure of the query, you may see a large number of work items displayed in the central pane. Scroll up and down to see the range of work items returned by the query you selected. Later, if you have team query privileges, you can refine the results of this query to restrict (or expand) the number of work items returned. If you do not have team query privileges, you can create user queries of your own to help you focus on the specific kinds of work items that reflect your part of the project team.

You can also choose from a list of available operations on a query on the work item by right-clicking on the work item. JDeveloper displays a menu of operations specific to that query.

2.2.3 How to Create Work Items

You can create a new work item in either of two ways:

- by selecting **New** from the repository context menu.
- by creating a work item from a template.

To create a new work item from the menu:

1. Click the right mouse button on the repository in which you wish to create the work item, then select New *repository name*.
2. Enter the information for the work item. This will vary depending on the connector and the repository in which you are creating the work item.

3. When you are finished, click **Submit**.

To create a new work item from a template:

1. Click the right mouse button on the repository in which you wish to create the work item, then select New *repository name* from User Template. This displays a pop-up menu of all the available templates for that repository.
2. Select the desired template from the menu.
3. Modify the preset data to fit the specific work item you wish to create. The fields and options in the work item will vary depending on the connector and the repository in which you are creating the work item.
4. When you are finished, click **Submit**.

2.2.4 How to Use Work Items

Before accessing work items, you need to log into the associated repository, through the Manage Accounts dialog. Once in the repository, you can run a query to return specific records from that repository. If you double-click on an individual record, JDeveloper opens that record as a work item. For more information, see Section 2.6.2, "How to Run a Query."

After selecting a work item from the repository, you can perform the following operations on those work items that are part of the tasks you are performing:

1. Link a change list to a work item when you check a file in to the version control system your team uses. This way, other team members who access the same work item will be able to refer to the change list, with changes and comments you have made. For more information, see Section 2.2.5, "How to Use Work Items with the Change List."
2. Save the work item as your Active Work Item. This makes it easy to access the same work item (for example, a bug you are currently working to resolve) after you exit JDeveloper and re-enter later. For more information, see Section 2.2.6, "How to Set and Display the Active Work Item."
3. Save the context (all open files, etc.) for this work item. This stores the application, files, and other hierarchical elements associated with the work item you are working on.

   You can save and restore context from any work item, not just the active work item. If you were last working on your active work item and closed JDeveloper, then on restart all your editors will open as they were when you closed. If you want to go to another work item (or if the last thing you did before closing JDeveloper was something other than working on your active work item), you can restore the context and quickly get back to where JDeveloper was when you last worked on that work item
4. Choose **Restore Context** for a specific work item (either the active work item or any other for which you have saved context) to easily repopulate JDeveloper with the elements associated with the project you are working on through the work item you created. For more information, see Section 2.2.8, "How to Work with Context."
5. Add a work item to a tag list. For more information, see Section 2.7, "Working with Tags."
6. Create a relationship to another work item. Section 2.8, "Working with Relationships."

7. Create a code review. See Section 2.3, "Working with Code Reviews."

2.2.5 How to Use Work Items with the Change List

When you commit files to your version control system, you can choose to associate them back to your version repository by linking the file to one or more work items. The Changes tab on each work item lists the file (or files) associated with that work item.

When you open a work item, it may have checkin information stored with it. This allows you to see a listing of all the checkins done against that item (there may be multiple checkins over time), the details of each file included in the checkin, and any checkin comments.

To add files to a work item as a change list:

1. In the Change List tab, select a list of files to be checked in and open the checkin dialog for your selected version control system.
2. Select the work item to associate these files with.
3. Check in these files to your selected version control system.

   The files will be listed on the Change List tab of the selected work item in Oracle Team Productivity Center.

JDeveloper stores the list of files with their checkin version ID, checkin date and any checkin comments written by the developer against each of the selected work items in the appropriate repository

2.2.6 How to Set and Display the Active Work Item

The active work item in JDeveloper is a work item that you wish to make available for easy access, and which will persist when you shut down and restart JDeveloper. You can make any work item as active at any time; only one work item can be active for a given team at a time, so making a new work item active replaces any previously selected active work item for that team.

To set the Active Work Item:

- Click **Make Active** in the work item's tool bar.

You can also right-click on a selected work item in a list of work items returned by a query, then select **Make Active**.

The Work Items accordion includes a link to the work item you have marked as an active work item, above the list of repositories. The selected work item is persistent for each team you are part of. If you shut down and restart JDeveloper, the active work item link will appear above the Work Items accordion when you restart the IDE later. This gives you an easy way, much like a bookmark, of returning to an item that you expect to be working on as a high priority.

To display the Active Work Item in JDeveloper:

- Choose the link on the top portion of the work item accordion to open the active work item. To select an entry in the query result list, click the right mouse button and then choose **Make Active** to mark a work item as active.

JDeveloper retrieves the work item from the appropriate repository and displays the work item that you had previously set to be the active work item.

2.2.7 How to Associate Work Items with Project Files

You can associate work items with the files you are working on in a project when you commit the files in your selected version control system. When Oracle Team Productivity Center is active, the Commit dialog (or in some systems, the checkin dialog) contains an additional field from which you can specify work items to associate with the file when you commit your changes.

When using Oracle Team Productivity Center, the Commit dialog for your selected version control system will include additional options designed to support Oracle Team Productivity Center's work items. For help at any time on your software's Commit dialog, click **Help** or press F1.

To associate work items with project files:

1. Commit your changes from the project file(s) you are working on, in the selected source control management system.
2. Select **Associate with work items** from the Commit dialog.

   To associate an Oracle Team Productivity Center work item with the file or files being committed, click on the green + sign above the Associate with Workitems panel of the commit dialog. This lets you choose a work item from the Select Work Item dialog.

   The Associate with Workitems panel of the commit or checkin dialog displays four columns of information about each work item you add:

Table 2–1 Information for associating work items with files being committed

Column	Description
Repository	The repository in which the work item is stored. You must have that repository's connector installed in order to access the work item.
Type	The type of work item. This is dependent on the repository in which the work item is stored. For example, work items in the Task repository are of type Task; other repositories may have additional types. See the documentation on the specific repository or connector for more information.
ID	The work item's ID—for example, a bug number or feature request ID.
Subject	The subject of the work item, imported from the repository in which the work item is stored.

   If you associate more than one work item with the file or files being committed, you can move selected work items up and down in the list by selecting the work item and then clicking the **Move Up** icon (double upward arrows).

3. When you are finished, click on **Commit**.

Note that the Commit dialog may have different options, depending on the source control management software your team uses. For help at any time on your software's Commit dialog, click **Help** or press F1.

2.2.7.1 Removing a Work Item's Association

From time to time, you may find it necessary to remove a work item's association from the project file. For example, if you restructure your project so that the work item actually refers to content in another file (such as moving a JavaScript function from an

HTML file to a.JS library), you can remove the obsolete association from the HTML file so that it is no longer connected to the work item.

To remove a work item's association from a file:

1. Commit your changes from the project file(s) you are working on, in the selected source control management system. For example: **Team > Subversion > Commit.**
2. Select the work item from the list, and then click on the red X.

   The work item is deleted from the commit menu immediately. If you delete a work item in error, add it back.
3. When you are finished, click on **Commit**.

2.2.8 How to Work with Context

Sometimes, the process of development involves having a number of files and windows open in JDeveloper—for example, while developing Java classes and visualizing those classes in a diagram. The challenge can be to remember which files and windows are open for any given task, especially if it becomes necessary to close JDeveloper, or to switch to another project with its own set of files.

Team Productivity Center addresses this issue through the ability to save and restore context for work items. If you are logged into Team Productivity Center and are tracking the task through a Work Item, Team Productivity Center lets you save the context of those files and IDE layout against that work item, using Save Context. Finally, you can delete a saved context when that phase of the project is done.

Later, you can return to that set of files and layout (for example, if you are working on multiple tasks concurrently) using Restore Context.

2.2.8.1 Saving Work Item Context

Saving work item context lets you easily associate the state of JDeveloper, including open files, windows and accordion panes, so that you can easily resume editing for a particular work item at a later date.

To save context for a work item:

1. Open a work item.
2. Select the **Save Context** icon. This saves the state of JDeveloper against the specific Work Item you are editing.

2.2.8.2 Restoring Context

If you have previously saved the context of a work item, you can restore it at a later date. Restoring the context of a work item opens windows, files, and accordion panes, making it easy to resume work if it becomes necessary to close and re-open JDeveloper.

To restore a previously saved context:

1. Open a work item.
2. Select the **Restore Context** icon. This returns the IDE to the saved state for that work item.

   In addition, you can click the right mouse button on an entry in the query result list and then choose **Restore Context**.

2.2.8.3 Deleting Saved Context

As useful as saving and restoring context can be, it may be necessary to delete a previously saved context, especially if the context for a work item changes (for example, if additional file libraries become involved in the resolution of a work item). To simplify, you can delete a saved context. (You can then save the context again, taking the new files, windows, or panes into account.)

To delete a saved context:

1. Open a work item.
2. Select the **Delete Context** icon. This removes any saved context from the work item.

2.2.9 How to View Work Item History

The Changes tab of the work item view lets you view the details of changes to a work item over time. You can select individual revisions, then look at the details of included files and related builds.

To view work item history:

1. Click the Changes tab of the work item you have selected. This displays a table showing the revisions associated with this work item, along the date each revision was committed, the name of the team member who committed it, and any comments made during the commit.
2. Double-click one of the revisions. This loads the details for the selected revision into the Included Files table, which lists the path to the file and the specific version associated with this revision in the work item history. Additionally, if available, the related build information is displayed.

2.3 Working with Code Reviews

Code reviews are integrated into many components of Oracle Team Productivity Center, and there are many ways to access them. They are associated with the files waiting to be checked in to your version control system. This portion of the guide will take you through the tasks that make up a typical work flow:

- Creating a new code review
- Adding reviewers to a code review
- Searching available code reviews
- Browsing and selecting an individual code review
- Commenting on a code review
- Adding content (such as a link, a code snippet, or a file) to a code review.

Each of these tasks is discussed individually.

2.3.1 How To Create a Code Review

You can create a code review in a number of ways, in the following elements of the JDeveloper IDE:

- The New Review menu item from the drop-down menu at the right of the Code Reviews panel.
- The Reviews tab of a work item: Click on the green plus sign.

2.3.1.1 Creating a Code Review from a Changeset

You can also create a code review from the changeset of your selected version control system.

To create a code review from the changeset:

1. Make sure that the Pending Changes window is displayed for your selected version control system. If it is not visible, select **Team > Subversion > Pending Changes** (or select the version control system your team is using).
2. Select the file in the Pending Changes window, then click the Create Code Review button.

Oracle Team Productivity Center creates a code review associated with the selected file.

2.3.2 How To Add Reviewers to a Code Review

Any team member can add other team members to a code review.

2.3.2.1 Adding Reviewers to a Code Review

Once you have created a code review, you can add reviewers from your team.

To add reviewers to a Code Review:

- Click the **Add Reviewer** icon (the silhouette with the green plus sign) in the upper-right corner of the list of reviewers.

The team members you add will now be able to access the review, make comments, and follow other reviewers' comments.

Other reviewers can also add reviewers in the same way.

2.3.2.2 Contacting a Reviewer about a Code Review

Communication between reviewers and developers is the key to successfully developing in teams. Whether you are reading others' reviews of code you are responsible for, or reviewing a team member's code, you can communicate directly with other team members through the Code Review feature. You can do this either through chat or email.

To contact a reviewer about a code review:

- Click the Reviewers tab of the Code Reviews window, right-click the name of the reviewer and then select **Chat** or **Send Email**.

If you are not connected to your chat server, JDeveloper displays the Connect to Chat Server dialog. See the section Section 1.3.2, "How to Use Team Productivity Center Chat".

2.3.3 How to Search Code Reviews

You can search the code reviews for specific information.

This window lets you search by characteristics of the review (for example, by review name, by date created, etc.), and allows multiple search criteria. The following table lists the criteria you can use for your search:

Element	Description
Created Date	Search for reviews by the date they were created. Criteria for comparison: Equals - return reviews created on a specific date. Not equal - return reviews created on any date except that entered Greater than - return reviews created after a specific date Less than - return reviews created before a specific date Greater than or equal - return reviews created on OR after a specific date Less than or equal - return reviews created on OR before a specific date Use the calendar icon to the right of the search box to browse for a date and enter it in the correct format (YYYY-MM-DD).
Review name	Search for reviews by the name of the review, as entered in the search field. Criteria for comparison: Like - return reviews that are similar (that is, have some of the words in the search field) in their name. For example, to find all reviews with the word “bug” in their name, select Like, then type “bug” in the search field. Note that search is case-sensitive, so searching for “bug” will not find reviews with “Bug” in the name. Equals - return reviews with the exact name entered in the search field. Not equal - return all reviews that do not have the exact name entered in the search field.
Review status	Search for reviews by their status (Under Review, or Closed). Criteria for comparison: Equals - return reviews that match the selected status. Not equal - return reviews that do not match the selected status.
Submitter Name	Search for reviews by a specific submitter, by name. Criteria for comparison: Like - look for strings similar to the submitter name you entered.
Submitter ID	Search for reviews by a specific submitter. Criteria for comparison: Like - look for strings similar to the submitter name you entered. Equals - return reviews that match a specific submitter. Not equal - return all reviews that do not match the name of the submitter.

To search by more than one characteristic, click the green + sign and specify the criteria for comparison. For example, you can search all reviews after January 1, 2012 that are still open and that have the word “Bug” in their name by adding three search criteria fields: one that searches for reviews on or after 2012-01-01, one that searches for status of Open, and one that searches for reviews with a name like “Bug”.

You can further refine the search by selecting one of the radio buttons below the search fields:

- **Match All**: return only reviews which match all the criteria entered in the search fields. The fields will be evaluated in the order they appear on the screen (that is,

criteria are evaluated as a logical "AND"). If you only specify one search field, Match All is selected as the default.

- **Match Any**: At present, Match Any is not supported. It is greyed out.

Selecting Match Any or Match All can have a large impact on the number of reviews returned. For example, if you are searching for all reviews during the month of March that have the word "Bug" in their title, selecting Match All will restrict the returned reviews to those which have both. Selecting Match Any will return all reviews during the month of March as well as all reviews with the word "Bug" in the title.

To conduct the search, click **Search**. JDeveloper returns all reviews that match the combination of criteria you have entered.

Once JDeveloper returns a list of reviews based on your search criteria, you can further filter the results by typing in the Filter field. This limits the displayed reviews to those which contain the string in the Filter field.

To select a review from the list, double-click the review. This places the selected review in the Code Reviews pane, listing all files included in that review. To view the review's details, double-click on one of the files. This opens the file in a review window.

To modify an existing search, add a new search field by clicking the green + sign, or remove an existing search field by clicking the red X. This lets you add or remove individual search criteria to refine your search.

To conduct a completely new search, click **Clear**. This clears all returned reviews, and also clears any search criteria fields you have entered.

2.3.4 How to Browse and Select Code Reviews

Once you have searched for and displayed the code reviews that match your search criteria, you can browse the returned reviews as follows:

- Sort the reviews by the contents of each column: Click on the column heading to sort by the values in that field. For example, to display reviews sorted by name in alphabetical order, click on the Review Name field. To reverse the sort, click Review Name again.
- Control which fields are displayed in the return list: Click on the drop-down list box at the right edge of the heading field to display the list of column names (Review Name, Submitter, Review Status, Created Date). A checkmark beside each column name includes that column in the display; to remove that column from the display, click the check mark to remove it. Note that the Review Name column cannot be removed.
- Open a review: Double-click on the review name, or click the right mouse button and select **Open Review.**

This loads into the Code Review tab. The drop-down list box in the Code Review tab displays three folders:

- Reviews You Created: this lists reviews for which you were the submitter.
- Reviews Assigned to You: this lists reviews that you have been assigned as a reviewer.
- Recently Viewed Reviews: this lists reviews that you have already opened and viewed. You can clear this by clicking the drop-down list box to the right and selecting **Refresh Review List.**

Once you have selected one of these folders, JDeveloper displays the reviews contained in that folder.

To select a code review:

1. Scroll to the review you wish to open.
2. Double-click on the review name.

This places the code review in your Code Reviews window. Once it is there, you can scroll through the individual files displayed in the Code Review tab.

2.3.4.1 Selecting Files Associated With a Code Review

Once you have selected a code review and can see its files displayed in the Comments tab of the Code Reviews window, you can select a specific file containing review comments.

To select a specific file to review:

- Double-click the name of the file you wish to review.

2.3.5 How To Comment on a Code Review

There are two kinds of comment you can make to a code review:

- General comments. These are meant to apply to the entire code review, and are displayed in the Comments tab, at the top of the list of review comments.
- Code review comments. These are typically meant to apply to a specific change in one of the files in the changeset, and are displayed under the general comments, grouped by the file to which the code review comments were made.

You can add these kinds of comment as follows.

2.3.5.1 Adding General Comments to a Code Review

General comments are intended to provide information about the entire code review, for example linking the review to a release, to a feature, or to some other part of the project which includes more than one file or more than one change to files in the changeset.

To add a general comment to a code review:

- Click the **Add New General Comment** icon (beside the Publish drop-down list) above the list of review comments.

When you have finished typing your general comment, press **Enter**. The general comment will not be visible to other team members until you click on the Publish drop-down and select **Publish Comments**.

2.3.5.2 Adding Code Review Comments to a Code Review

Code review comments are more focused in nature than general comments. Code review comments pertain to a specific change in one of the files in the changeset, and are meant to provide information about the change, or to ask for clarification or otherwise engage the team in specific conversation about the file on the change set.

To add comments to a code review:

1. In the file you are reviewing, scroll to the line about which you wish to add a comment and select a section of code.
2. Click the right mouse button, then select **Add Review Comment**.
3. Type your comment, then press the **Enter** key.

The comment will be visible to other team members when you click on the Publish drop-down and select **Publish Comments**.

2.3.6 How To Add Content to a Code Review

In addition to making comments, you can add specific content types to a code review: attachments, links, and code snippets. You can attach files to a review at the top level; you can also attach files, add links, and include code snippets to individual comments.

2.3.6.1 Adding Attachments to a Code Review

You can attach files to a code review, at the top level of the review (rather than to a specific comment inside the review).

To add an attachment to a code review:

1. Select a review, then select the Attachments tab.
2. Click the **Add Files** button (the green plus sign).
3. Browse to the file you wish to add, then select **Save**.

2.3.6.2 Adding Content to a Code Review Comment

You can also attach files, links, and snippets (short sections of code meant to exemplify a technique or methodology) to a specific comment inside a code review.

Note that to add a snippet of code, you will need to select the snippet from another file (either in JDeveloper or in any text editor) and save it by typing **Ctrl+C**.

Similarly, you will need to prepare a link by placing the cursor over the URL in your Web browser and copying the link as required by the browser of choice. For example, in Firefox, roll over the link, click the right mouse button, and select **Copy Link Location**.

To add content to a code review:

1. Select or create a review comment.
2. Click the **Add Content** button (the green plus sign).
3. Select the type of content you wish to add by selecting the appropriate menu item: **Add Links**, **Add Snippets**, or **Add Files**.
4. For links or snippets, type Ctrl+V to paste the previously saved snippet or link into the field.
   For files, browse to the location of the file you wish to add, select the file, and click **Save**.

2.4 Working with the Task Repository

The Oracle Team Productivity Center Task repository contains work items intended as a shared "to do" list for your Oracle Team Productivity Center team. To access the Task Repository, you must install the Task connector, then use the Manage Accounts dialog to access the Task repository for the first time. Access to the Task Repository does not require a user ID and password.

When Team Productivity Center shows a list of Task work items, the display contains the following columns:

Table 2–2 Task work item fields

Column Header	Description
Task ID	A unique identifier for this work item in the Task Repository
Task Name *	The name given to this task. Mandatory field.
Created By	The user who created this task. This field is read-only.
Team *	The name of the team to which this task is assigned. Mandatory field.
Assigned To	The owner of this task
Priority	The relative importance assigned to this task. Values can be High, Medium, Low or None
Status	The current status of this task. Values can be Assigned, Blocked, Done, or In Progress
Start Date	The date on which this task began.
End Date	The date on which this task is to end.
Description	Enter any descriptive comments which will be of use in defining or completing the task.

You specify or change the values that appear in each column when you create or edit a Task work item. See Section 2.4.2, "How to Create a Task Work Item."

2.4.1 How to Find Tasks in the Task Repository

As with other work item repositories, you query the Task Repository to return one or more work items that match your query terms.

To query the Task Repository:

1. Select **Task > My Queries** (or **Task > Team Queries**).
2. Double-click the query you wish to run.

Oracle Team Productivity Center displays the work items which match the query you entered. For example, to display a list of all Task work items in the repository, create a query that returns Task work items with a Task ID greater than or equal to 1.

For more information about creating and working with queries, see Section 2.6, "Working with Queries."

2.4.2 How to Create a Task Work Item

You create a Task work item from the Work Items accordion.

To create a Task work item:

1. In the Work Items accordion, right-click **Task**, and then select **New Task**. This opens the Task work item editor.
2. Enter the information for the task in the fields. For more information about any of the fields in the Task work item, press F1 at any time.
3. When you are finished, click **Submit**.

The next time you view all available Task work items, the newly created Task will be visible in the Work Item pane. For more information, see Section 2.2, "Working with Work Items."

2.4.3 How to Edit a Task Work Item

You can edit any Task work item visible in the Work Item pane, after running a query to return the Task work items of interest to you. For example, if you have recently completed work on a Task work item, you can change the Status field to **Done**.

To edit a Task work item:

1. In the Work Item pane, locate the Task work item you wish to edit.
2. Double-click the selected work item. This opens the Task work item editor.
3. Enter the information for the task in the fields. For more information about any of the fields in the Task work item, press F1 at any time or click the **Help** icon.
4. When you are finished, click **Submit**.

The next time you view all available Task work items, the updated Task will be visible in the Work Item pane. For more information, see Section 2.2, "Working with Work Items."

2.5 Working with Attachments

Team Productivity Center lets you add attachments to work items, as long as the work item's connector supports it. For example, if you build a jar file on your local system as part of testing a bug fix, you can attach the local jar to the bug report. Your QA staff can then download the attachment and test your fixes. Your user-experience team can attach graphics showing their design for the layout of an application's screens and dialogs, making it simpler to design the interface in JDeveloper. And because these attachments are available to any kind of work item, you can tag the work items or define relationships to make it easier to keep track of your work.

Some repository connectors support attaching any type of document (such as test cases, snippets of code, screenshots, specifications) or Web link to a work item. It is up to the connector writer to indicate whether attachments are supported and if so, to store the documents in a document repository and return them to Team Productivity Center. For more information, see Chapter 4, "Working with Oracle Team Productivity Center Connectors."

If attachments are supported by a work item repository, the work item displays an Attachments tab. Clicking on that tab displays all attachments that have been attached to the work item already. Double-clicking on an attachment either launches the default application associated with it, or saves the file locally. Users also have the ability to add new attachments to the work item by selecting a local file on the system to upload. Depending on the document store's permission settings and the privileges set for your role, you may also be able to delete documents attached to work items. Because these documents are attached to the work item, this means that all Team Productivity Center users who access the work item can view the attachments.

Attachments can be files from your local file system or your local network, in which case the file contents are uploaded to a document repository specified by the connector author. Because these references are attached to the work item, this means that all Team Productivity Center users accessing the work item can view and download the attachments. The details of uploading and downloading depend on how the connector has been defined; connector authors are encouraged to use the standard file browser interface for uploading and downloading attachments.

In addition, work item attachments can be links to content on the Web. Adding an attachment of this type adds the URL to the work item; to view the destination of the URL in a work item you are viewing, click on **Download Attachment**.

2.5.1 How to Add Attachments

Adding attachments to a selected work item lets you share additional information (such as a customer email, a UI mock-up, a bug report, or any other file attachment) with your colleagues. This can help ensure that all team members are working from the same source material or background information while resolving their individual areas as represented by the work item.

To add an attachment to a work item

- Select **Work Item view > Attachments tab > Add Attachment icon**

This opens the Add Attachment dialog, with which you can save a file or link to the work item.

Attachments can be one of the following:

Document File
Select the file icon to open a file system browser. Browse to the desired file, then select **Save**.

Web link
Attach a URL to this work item. Enter the following information for the URL you wish to save as an attachment:

- **Name**: The name that you wish to appear in the Team Productivity Center work item.
- **Link**: The URL you wish to associate with this name.

When you have specified the file or URL to use as the attachment to this work item, select **Add**.

2.5.2 How to Download Attachments

You can download attachments from a work item in Team Productivity Center to your local workstation. You can then save it or open it with the associated program (for example, a text editor or PDF viewer).

To download an attachment:

- Select **Work Item view > Attachments tab > Download Attachment icon**.

This opens the Download Attachments dialog, from which you can make the attachment available for viewing or editing locally. Select one of the options from the dialog:

Open Attachment
Opens the attachment in the appropriate editor or application. For example, if you download a text file, JDeveloper opens the current default text editor (for example, WordPad).

Save Attachment
Copies the attachment to your local file system. Select this option, then select the file icon to browse to the location in your local file system where you wish to save the attachment.

2.5.3 How to Update Attachments

You can update an existing attachment when there is new content available to share with the team.

To update an attachment with new content:

- Select **Work Item view > Attachments tab > Update Attachment icon**.

This opens the Update Attachment dialog, which you can use when the content of the attachment has changed since the last time you accessed it. For example, if the attachment is a text file containing minutes of a meeting on the issue, use Update Attachment to copy the newest minutes to the work item.

When you select Update Attachment, JDeveloper opens a file browser dialog from which you can select the file to update. Browse to the file, then select **Save**.

> **Note:** You can only update a file with the same name. For example, if the minutes of your design meeting are dated with the month and date (such as `minutes_jan_15.txt`), you will not be able to update the attachment with the minutes from the following week (in this example, `minutes_jan_22.txt`).

2.5.4 How to Rename Attachments

You can rename an attachment that has already been uploaded.

To rename an attachment:

- Select **Work Item view > Attachments tab > Rename Attachment icon**.

This opens the Rename Attachment dialog, from which you can change the name of the attachment associated with the work item.

This dialog contains the following fields:

Table 2–3 Fields in Rename Attachment dialog

Field	Description
Type	The type of attachment (File or URL) represented by the selected attachment.
Old Name	The existing name of the selected attachment.
New Name	The name to which you wish to change the attachment.

When you are finished, click **OK**.

2.6 Working with Queries

Queries are the way Team Productivity Center retrieves work items, based on the search criteria, from the repositories to which you are connected. As you work with Team Productivity Center, you will have the opportunity to work with team queries that your administrator (or another team member) has created to cover common situations. For example, your entire team would benefit from a team query that searches a bug database and returns all open items assigned to the project or product the team is working on.

In addition, you can create custom queries that apply specifically to your work. For example, a query that searches the same bug database and returns all open items assigned to you could help you manage your workload effectively.

In either event, the result of the queries in these examples will be a list of records extracted from the bug database, presented in Team Productivity Center as a list of work items. Other queries from different repositories might return records from a

feature tracking list, from a schedule management program, or from any other repository you and your team use. Once the query returns these work items, you can then interact with them in the various ways that Team Productivity Center offers.

2.6.1 How to Create Queries

Oracle Team Productivity Center lets you create two types of query: user queries and team queries. User queries are those you create to solve a specific problem that you are facing—for example, to find bugs in a specific date range, or filed by an individual team member if you are taking over another member's tasks. You can easily create specific user queries to solve these and other immediate needs.

If you find that you are regularly searching for the same kind of work item, you can save your user query rather than modifying an existing one. For example, if you are tracking bugs on different branches of the same product, you can save user queries for each branch individually and retrieve work items that match each query as needed.

Additionally, some queries may be general enough that they can be used by multiple team members. For example, the entire team might want to view bugs filed as a result of a usability session or beta program. Queries with a wide application such as these, which apply to more than one user, are saved and accessed as team queries. Depending on the Team Productivity Center administrative rights you have, you may be able to save team queries.

To create a query:

1. Select a repository and create a query for it, returning fields and information of interest to you.
2. Review the results of that query, editing the settings as required.
3. Save the query for later use. When you use the query again, it will return the same fields from the repository but with updated information.
4. If you have administrative privilege and the query is general enough to be of use to the entire team, save it as a Team Query.

2.6.2 How to Run a Query

Running a query from Team Productivity Center gives you access, within JDeveloper, to the resource for which the query returns results. In a bug database, for example, running an Oracle Team Productivity Center query can return all bugs matching the criteria of the query you run. This simplifies tracking bugs, fixes, and features compared to using separate applications in multiple windows.

To run an existing query:

1. Double-click on the Work Item Node that contains the query you want to run. (Alternatively, click on the + to the left of the node.)
2. Open either Team Queries or My Queries (if a "+" doesn't appear adjacent to the Queries icon, it means no queries are available).
3. Double-click on an available query (such as Open Bugs). (Alternatively, right-click on the Query and choose Run.) The query results should appear in a tabbed panel named for the query.

Note: For any returned results, you can double click on the returned row to view details and make modifications and add comments to the extent supported by the Oracle Team Productivity Center Connector for the particular repository you are accessing.

2.6.3 How to Customize a Query

In Oracle Team Productivity Center, queries are easy to customize, even if you start with a pre-existing team query. The key to this customization potential is that data fields that are made available from the underlying repository can be used in two ways:

- To apply additional fields to the underlying information.
- To tailor the resulting on-screen report.

For example, a team query might provide the following fields:

- **Product**
- **Type**
- **Status**

As the development process draws to a close, you may find it useful to add Priority as a filtering element.

Here is a simple example of how an existing query can be enhanced by adding an additional query field filter. In this example, we're creating a query on a defect repository, specifically a query that returns defects that do not have the priority of "Blocker."

To add an additional query field filter:

1. Click the green + icon to add a new criteria line.
2. With the Query Results tab open, click any dropdown on the far left of the query to see a list of available fields.

   Note: You can choose a field element that is already in use. For example, you could add a second Project field element to get a report on two projects.

3. In the leftmost field dropdown, select **Priority**.
4. Change the **Not Equals** dropdown to **Equals**.
5. Choose **Blocker** from the rightmost dropdown.
6. Click **Search**.

To clear the field option selections you have made before saving the query, select **Clear** from the More Actions dropdown menu. To clear all the entries, select Restore.

The query returns the results that match the criteria you entered.

Tip: You can sort the results by any column simply by clicking on the column name. In addition, the widths of the columns and the currently sorted columns will be saved when you save the query

Once you have created a custom query, you can save it.

2.6.4 How to Rename a Query

Just as a project matures and develops over time, you may need to change the ways that you interact with it. You may wish to change the name of a query to reflect other modifications or customization you have made to it, as well as to reflect changes in the project itself.

To rename a query:

1. Click the right mouse button on the query and select **Rename**.
2. Enter the new name for the query in the Name field.
3. Click **OK**.

The query will now be available with the name you entered.

2.6.5 How to Save a Modified Query

Often, an existing query may contain slightly different criteria than what your present needs require. For example, if you have a query which has all the right criteria for a regular search, you might wish to modify the query, for example by changing the date to return the latest work items. You can then save this modified query so that you can use it again in the future without having to re-modify the original query.

To save the new or changed query:

1. Click the **More Actions** button on the right side of the screen report.
2. Select **Save As**.
3. Pick a name for your query, such a Open Blocker Bugs.
4. If you have team administration privileges, you will have a choice of creating a Team Query or a User Query. If not, you will only be allowed to create a User Query. In either case, for this example, choose User Query and click **OK**.

If you have team administration privileges and the query you are saving would be useful to all the members of your team, you can also save it as a Team Query.

Note that all queries are saved against the current team. Even if you have access to this repository from multiple teams, queries are saved in the scope of the current team.

2.6.6 How to Create a Team Query

If you have team administration privileges, you will likely want to create team queries based on the projects your team is working on and the status of the projects. For example, towards the end of a project, creating a query around Priority 1 bugs with a description that includes the phrase "stop ship" might be a useful addition to the set of queries available to the team.

The option to save a team query only appears if you have team privileges. If you find that you are frequently creating queries that other members of your team would find useful, ask your Team Productivity Center administrator to grant you team privileges. For more information, see Chapter 3, "Working with Oracle Team Productivity Center Administration."

To save a new team query:

1. Click the **More Actions** button on the right side of the screen report.
2. Select **Save As**.

3. Pick a name for your query, such a Open Blocker Bugs.
4. If you have team administration privileges, choose Team Query and click **OK**.

2.6.7 How to Delete a Query

Only users with a role that allows them to modify team queries can delete team queries. In the default roles used by Team Productivity Center, this includes users with roles of Team Administrator and Group Administrator.

To delete a query you created:

1. In the Oracle Team Productivity Center panel right-click on the query you want to delete.
2. Select the **Delete** option, then answer **Yes** to the verification dialog.

2.7 Working with Tags

In addition to querying individual work item repositories, Oracle Team Productivity Center provides a tagging mechanism that allows you to tag work items from multiple repositories for additional integration. You can then view items from across your repositories by querying for tags you have set on work items; Team Productivity Center will display work items with those tags, regardless of which repository contains them.

A key distinction between tagged work items and work items returned by a query is that the query searches through the entire repository, finding and returning items that match the query terms (for example, a range of bug IDs or the owner of a feature enhancement request). Tagged work items, on the other hand, are work items you have already identified and specifically tagged for a particular purpose.

Team Productivity Center uses two kinds of tags:

- Team tags, set up by your team or group administrator, are available for use by all members of the team. Each team's tags are unique to that team, not shared among other teams. If you work on more than one team, the administrators of the various teams may use different tags (though they may also choose to use the same tags, particularly if your organization uses company-wide tags for bug tracking, feature requests, etc.)
- Private tags, which you define on your own, help you find, sort, and group work items by project, by urgency, or by a specific section of the code. For example, you might choose to set up a tag for use in a bug database for bugs where you have fixed the code and checked it in, but not yet verified the fix in a product build. You can tag the appropriate work items—entries in a bug-tracking database, in this example—with this private tag each time you've checked in the fixes for that work item's issue.

Once you tag a number of work items, you can return a list of work items that you have identified with a specific tag. Creating a tag, as described here, for items you've fixed but not verified could greatly simplify verification of your assigned bugs once the next build of the product is available, because you can query the work item repository for all items that use this tag.

2.7.1 How to Create, Modify, and Delete Tags

You create tags by using the Manage Tags dialog. If you are a team administrator, you have the choice of creating team or user tags; if you do not have administrative

privileges, you can only create user tags. Administrators and non-administrators alike can apply tags to work items.

2.7.1.1 Creating Tags

Creating a tag gives you an easy way to label work items across repositories, allowing you to take a quick look at previously tagged work items that all apply to a common issue.

To create a new tag:

1. Select the tag icon from the Work Item accordion toolbar in the Team Navigator.
2. Select **Manage Tags**.
3. In the Manage Tags dialog, click the green plus button.
4. Enter a name and description for the tag. The name should be short and easily distinguished; the description can include more detail.
5. If you are a Team or Group Administrator, select the visibility for this tag:
   - **Team**: all members of the team can view and query by this tag. (Option available to administrators, or to users with Manage Tag privileges.)
   - **User**: only you can view and query by this tag.
6. Click **OK**.

2.7.1.2 Modifying Tags

As your project evolves, you may need to modify an existing tag to more accurately reflect the tag's current usage, role, or importance. For example, if you have been using a tag to identify a series of related bugs which later testing determines to have a greater severity than initially detected, you can change the name of the tag applied to these bugs to reflect the increased importance.

To modify a tag:

You can modify any tag to which you have access, changing its name, description, and (if you have sufficient privileges) scope of visibility.

1. Locate the Work Items accordion in Team Productivity Center.
2. Click the Tag icon, selecting **Manage Tags**.
3. In the Manage Tags dialog, select the tag you want to change.
4. Double-click the field you wish to modify and make the necessary changes.
5. Click **OK**.

2.7.1.3 Deleting Tags

During the project lifecycle, you occasionally reach a point where you need to remove some of the "clutter" around your working environment. Deleting unused or obsolete tags is one way of cleaning up your JDeveloper environment, making the tags you are still using more effective because there is less distraction.

To delete a tag:

1. Locate the Work Items accordion in Team Productivity Center.
2. Click the Tag icon, selecting **Manage Tags**.
3. In the Manage Tags dialog, select the tag you want to delete.

4. Click the red X to delete the selected tag.
5. Click **Yes** to confirm the deletion.

2.7.2 How to Use Tags

Work items artifacts can be tagged with existing keywords. For example, bugs in a product under development may be tagged by a project manager as Hot, Warm, EOD (end-of-day), or Deferrable.

> **Note:** These tag names are just examples. You and your team can devise a tag naming scheme that best matches your tasks and work style.

In addition, artifacts in other work items can also be tagged with the same terms. For example, a JIRA feature could be tagged Hot. In such a case both the bugs and the feature request would show up if you look at the list of work items tagged as Hot.

You use tags by applying them to work items.

2.7.2.1 Applying Tags to Work Items

You can apply a tag, or multiple tags, to work items in any of your existing work item repositories. This lets you group similar work items by whatever characteristic you are using the tags to sort or group your work items.

To apply a tag to a work item:

1. Open the repository containing the work item you wish to tag—for example, a bug database.
2. Double-click the work item to be tagged, and then select the **Tags** tab.
3. Click the green + icon to add a tag. This opens the Apply Tags dialog.
4. Select the tag or tags you wish to apply to this work item, and then click **OK**.

Once you have applied tags to work items, you can view work items by the tags you have applied to them.

If the tag you want to apply does not already exist in the Apply Tags dialog, click on Manage Tags to open the Manage Tags dialog and create a new tag. See Section 2.7.1, "How to Create, Modify, and Delete Tags."

2.7.2.2 Displaying Tagged Work Items

You can easily check your work items to see those that you have already tagged. This can be a quick and effective way to view a number of work items, in multiple repositories, which you (or your team) have tagged as being related in some way.

To view work items associated with a particular tag:

1. Select the tag icon from the Work Item accordion toolbar in the Team Navigator.
2. Select **Query by My Tag** to select a private tag, or select **Query by Team Tags** to select a team tag.
3. Select a tag name from the dropdown list.

A JDeveloper window will open, showing the items whose tags match the name you selected. This window shows the results of a cross-repository search for tagged work

items, essentially returning all work items to which you or another team member (in the case of team tags) has assigned the tag on which you are querying. To get a list of tagged items for just one repository, right-click on the repository and select **Query By My Tag** or **Query By Team Tag**.

When you search across repositories, JDeveloper only displays the Repository, ID, Type and Subject (generally the common fields). When you search within a repository, additional fields may be displayed.

You can remove a work item from a Tag Result by selecting the item and clicking the Delete icon.

2.7.2.3 Searching for Specific Tags

While one of the most useful aspects of tagging work items is the ability to show the contents of multiple repositories, you may occasionally find it necessary to search a specific repository for a tag. For example, if a colleague mentions a specific bug report that you think may have an impact on your work, you can search just the bug database for work items with this tag, and see if any of them relate to your portion of the project.

To search for tagged work items in a specific repository:

- Click the right mouse button on the repository you wish to search. This brings up a context menu with the following options.

Table 2–4 Options while searching for tagged work items

Option	Description
Query by My Tag	Displays a list of your tags, from which you can select the tag you wish to use for the query.
Query by Team Tag	Displays a list of team tags from which you can select the tag you wish to sue for the query.
Query by ID	Look through this repository and return the work item with the single ID you are looking for. The query will return the specified work item; to view tags associated with that work item, click the Tags tab of the work item display.
New Query	Opens the New Query pane in JDeveloper. See Section 2.6.1, "How to Create Queries."
New <work item>	Opens a pane in JDeveloper from which you can create a new work item of the type associated with the selected repository. For example, in the Task repository, this menu selection will say New Task. To obtain more information about the specific repository, click inside one of the fields in the New Workitem pane and press F1.
Refresh	Updates the display of the selected repository in the Work Items accordion.

Note that only the top two menu items deal with tags; the other items on the menu provide additional functionality in working with work items and queries.

2.8 Working with Relationships

A relationship is an association between work items, which can either be in the same repository or in different repositories. It provides a way to relate two work items together. You can navigate to what is related to the current work item, and to the work item from the elements related to it.

Because relationships cross repositories, you can establish a relationship between, for example, a defect-tracking system and a customer request database; the relationship means that the work items in the two repositories will refer to each other, so that team members who view the defect-tracking entry will be able to track down the customer request as well.

Relationships cross the entire team. For example, you could create a relationship between a bug and an item in the Atlassian JIRA feature request tracking repository. Once you create the relationship, anyone from any team who opens that bug will see the relationship with the JIRA work item. However, team members who don't have that JIRA repository in their connectors will not be able to open the work item.

2.8.1 How to Add a Relationship to a Work Item

You add a relationship to a work item through the Add Work Item Relationship dialog. This dialog displays a list of work items from which you can select the ones for which you wish to create a relationship with the work item you have selected.

To add a relationship to a work item:

1. Select the work item to which you wish to add a relationship, then select the Relationships tab.
2. Click on the **Add** drop-down (the green +). This displays the work item relationship context menu, which lets you determine how to sort and display the available work items for the relationship you are adding. See Section 2.8.3, "How to Associate a Relationship with a Work Item."
3. Select an option from the context menu. The Add Work Item Relationship dialog is displayed.
4. Click on the work item (or work items) for which you wish to create a relationship with the item you selected, and then click **OK**.

2.8.2 How to Delete a Relationship

You delete a relationship from a work item with the Delete button (the red X) from the work item tool bar.

To delete a relationship from a work item:

1. Select the work item from its repository.
2. Click on the **Relationship** tab.
3. Select the relationship that you wish to delete, and then click **Delete** (the red X).
4. When the Delete Tag from Work Item dialog asks you to confirm, click **Yes**.

The relationship is deleted.

Note that the work item referred to by the relationship is unchanged. You can add other relationships using that work item as required.

2.8.3 How to Associate a Relationship with a Work Item

When you add a relationship to a work item, the **Create** button (the green +) displays a context menu that lets you select how to sort, display and select other work items for the relationship. This context menu contains the following selections:

Table 2–5 Associate Relationships with a Work Item

Selection	Description
By My Tags	Displays a list of your user tags. When you select a tag from the list, the Add Work Item Relationship displays all work items that have been tagged with the selected user tag. To select one of these tagged work items for the relationship, click on the work item from the Add Work Item Relationship, and then click **OK**. To select more than one tagged work item, hold down the Control key and click any additional work items (**Ctrl+Click**).
By Team Tags	Displays a list of your available team tags. When you select a tag from the list, the Add Work Item Relationship displays all work items that have been tagged with the selected team tag. To select one of these tagged work items for the relationship, click on the work item from the Add Work Item Relationship, and then click **OK**. To select more than one tagged work item, hold down the Control key and click any additional work items (**Ctrl+Click**).
By Opened Work Items	Displays a list of all the work items you currently have open in JDeveloper. To select one of these open work items to use in the relationship, click on the work item from the Add Work Item Relationship, and then click **OK**. To select more than one open work item, hold down the Control key and click any additional work items (**Ctrl+Click**).
By Active Work Item	Creates a relationship between the selected work item and the active work item. See Section 2.2.6, "How to Set and Display the Active Work Item." If the work item you have selected is the active work item, this menu option will not be available.

2.9 Working with the News Panel

Oracle Team Productivity Center includes a News panel, in which you can select from available RSS feeds to have news of interest to you displayed in JDeveloper. The News panel is adjacent to the Messages and Chat panels in the main JDeveloper pane.

The News panel gives you a number of controls over what appears in JDeveloper:

- Important items. Important items are identified by the sender, at the RSS feed. You can choose to display only Important items, or to display all items.
- Quick sorting of news items by column. To change how news items are displayed, click the heading of each column (title, date, categories). Clicking toggles the display; for example, clicking once on the date column displays news items oldest first, while clicking again displays news items newest first.

To set up a news feed:

1. Select **Tools > Preferences > News**. This opens the News dialog.
2. Click the green + sign to add a news feed.
3. Enter the news feed URL, then click **Test** to verify the connection.

Additionally you can also configure other user preferences here, such as the frequency for refreshing news items.

Once you have set up the News feed, select **Team > News** to open the news window. By default, the news window is docked in the bottom pane (shared with the Log window and Pending Changes windows, if open.)

3

Working with Oracle Team Productivity Center Administration

This chapter describes how to administer users, teams, roles, and repositories with Oracle Team Productivity Center.

This chapter includes the following sections:

3.1 About Working with Oracle Team Productivity Center Administration

Oracle Team Productivity Center requires a certain amount of administration, both during initial setup and also on an ongoing basis. Oracle Team Productivity Center administration comprises the management of the following areas: repositories, users, teams, roles, and Lightweight Directory Access Protocol (LDAP).

Repositories store the information used by your team to track, manage, and follow up on issues, features, and other elements of the product throughout its lifecycle. Once you add repositories to Oracle Team Productivity Center, you need to make them available to your team members.

The way your team members access and interact with the information in these repositories depending on two separate elements:

- The nature of the repository and its information. A repository that requires SSL login will have different access requirements from a repository that permits anonymous login; furthermore, a repository that tracks bugs against the files in a change list will have different interactions than a task repository which has beginning and ending dates.
- the privileges you have assigned to each individual user, or to the role or team to which that user belongs. Users who have team query privileges will be able to create and modify queries accessible to the team; users who do not will only be able to create and modify user queries.

When you add a new user, you have the option of assigning them permissions which allow them to create a team, create a user, or act as a team administrator. For more information, see Section 3.4, "Working with Users, Teams and Roles".

When you assign users to a team, they acquire a role, which is a mechanism for grouping together the abilities to modify different aspects of Team Productivity Center content. The Team Administrator or Group Administrator, at the Team level, can modify individuals' roles, either on a person-by-person basis. Creating new roles (with specific permissions that serve the needs of your organization) is possible for administrators at the system level.

3.1.1 Understanding Repositories, Users, and Teams

Managing the repositories used in Team Productivity Center by your organization ensures that your users and teams have access to the records of features, bugs, progress, and other tracking mechanisms you use during the product lifecycle. These repositories store the work items that track individual issues during development.

To make a repository available to Oracle Team Productivity Center, you must first have access to the repository in your network, and then you must install the Oracle Team Productivity Center connector that links you to that repository.

For more information on installing connectors to repositories, see *Installing Oracle Team Productivity Center Server*.

Administrators typically perform the following tasks:

- Configure repositories for use with Team Productivity Center so that users and teams can access work items. For more information, see Section 3.2, "Adding Repositories to Oracle Team Productivity Center."
- Assigning repositories (including versioning repositories used by source control management software, such as Subversion) to teams. For more information, see Section 3.3, "Making Repositories Available in Oracle Team Productivity Center."
- Add users and manage teams as the product matures. For more information, see Section 3.4, "Working with Users, Teams and Roles."
- Manage the roles assigned to users and teams, to make sure the right team members have the ability to modify the appropriate elements of the repositories they access. For more information, see Section 3.4, "Working with Users, Teams and Roles."

While these tasks are roughly sequential (that is, you need to add repositories and make them available before you can assign users and teams to those repositories), you will most likely go back and perform these tasks at different times as the product matures and the team membership changes.

3.1.2 Understanding User Permissions

In Team Productivity Center, permissions define what controls a user has over the creation of new users and new teams. Each user can have different permissions that apply, no matter what teams they belong to. These permissions are as follows:

- **Create New Teams**: Users with this privilege can create new teams. Unless they also add themselves as members of the new team, they will not be able to edit that team after they close the administration dialog.
- **Create User Accounts**: This user can create new user accounts. These new user accounts cannot be edited once the administration dialog is dismissed. Only Administrators can further edit users.
- **Administrator**: All functions of Team Productivity Center are open to users with Administrator permissions. Only Administrators may edit users, add/edit repositories, and modify team members' roles.

3.1.3 Understanding User Roles

Roles are the mechanism Team Productivity Center uses to give users, at a Team level, the ability to modify the elements of Team Productivity Center, such as work items, queries, tags, and document. Each role defines what capabilities and privileges which users on that team can have. By default, Team Productivity Center roles include the following three roles:

- **Team Members** can access and edit work items, and can also create and manage queries, tags, and documents for their own use.
- **Team Administrators** have Team Member privileges, but in addition they can create and manage queries, tags, documents and sources for use by all members of their team.
- **Group Administrators** have all Team Administrator privileges, but can also create and manage queries, tags, documents and sources for their teams, plus any child teams that are contained within their own team.

The administrator can either use these default roles and privileges, or adjust them by changing the selected privileges for each role. The administrator can also create new roles with different combinations of privileges, selecting from an available list on the Roles tab of the Team Administration dialog. Those privileges are:

- Manage Team Documents - a user whose role includes this privilege can create and modify team documents
- Administer Teams - a user whose role includes this privilege can add. modify, and remove team members and repositories
- Manage Team Queries - a user whose role includes this privilege can create and modify team queries.
- Manage Team Sources - a user whose role includes this privilege can modify team sources, including versioning
- Gain Privilege over Child Teams - a user whose role includes this privilege can modify the content of child teams to the same degree that they modify their explicit (parent) team
- Manage Team Tags - a user whose role includes this privilege can create and modify team tags.

3.2 Adding Repositories to Oracle Team Productivity Center

Adding repositories to Team Productivity Center is an important administrative task, though typically you only add each repository once. Adding a repository makes it possible for team members to access work items, which are in turn artifacts of specific repositories. The team administrator adds the repository to the team's implementation of Team Productivity Center; the individual team members access those repositories while checking, tracking, and modifying work items as part of the product development lifecycle.

One crucial prerequisite for adding a repository: you (or any Team Productivity Center administrator) must have installed the connector for that repository on your Team Productivity Center server. You install connectors using the Team Productivity Center installation tool, available by download from Oracle Technology Network. Note that this is different from the connector extensions that all Team Productivity Center users download from **Help > Check for Updates**. You (or any administrator) can only add a repository to Team Productivity Center if that repository has a connector installed as a JDeveloper extension, through the Team Productivity Center installation tool.

For more information on installing connectors to repositories, see the *Installing Oracle Team Productivity Center Server*.

3.2.1 How to Add a Repository

Before adding a repository, make sure you have all necessary connection information, such as the server name or URL, port number, authentication credentials such as user name and password, or any other data necessary to make a connection to the repository you are adding.

Note that the terms used for these attributes may vary among connectors, and what one connector calls a server address may be called a server URL by another connector. Be sure to check the specifics of each connector as you add their repository to Team Productivity Center.

To add a repository to Team Productivity Center:

1. Select **Team > Team Administration > Repositories** tab.
2. In the Repository area, click the green + icon.
3. From the Connectors drop-down, select the appropriate connector for the repository you are adding. The available connectors depend on which connectors you have installed.
4. Enter a name and description. This is the name that will appear in the list of repositories available to a particular team.
5. Enter a local alias name for the repository server and a description in the Repository Servers fields.
6. Enter the repository parameters, which are a set of name/value pairs. These will depend on the data required by the connector to access the repository. For some connectors, such as Atlassian JIRA ®, the Name field should be Server URL while the value field is the URL of the JIRA server that serves this repository. For other connectors, you may be required to enter additional names. Check with the connector author or with the connector's online Help for more details about the specific data you are required to enter when adding this repository. In any case, the parameter name is provided when the connector is installed, so as administrator, you only have to enter the value for the name parameter.

Note also that newly created repositories still need to be assigned to teams before they can be used. For more information, see Section 3.3, "Making Repositories Available in Oracle Team Productivity Center".

3.2.2 How to Remove a Repository

Before you remove a repository, write down the various URLs and parameters associated with it, in case you need to add it back to the system at some later point.

Note that you cannot remove a repository that is being used by a team. This prevents you from removing any team's associations with a repository.

To remove a repository:

1. Select **Team > Team Administration > Repositories**
2. Click the repository you wish to delete.
3. Click the red X button.

The repository and any associations to it from teams will be removed.

3.3 Making Repositories Available in Oracle Team Productivity Center

Once you have added a repository, you need to make it available to users and teams of Oracle Team Productivity Center. This makes it possible for teams and users to access work items in the repositories you have added. In addition, as the administrator, you can create versioning repositories, giving your team members a quick way of accessing your selected source control system through the Repositories tab of Oracle Team Productivity Center.

3.3.1 How to Access and Select Repositories in Team Productivity Center

Oracle Team Productivity Center links JDeveloper users to external repositories. These repositories can include defect-tracking databases, customer-request tools, and other collections of data such as versioning systems.

The Administrator has the ability to add repositories to Team Productivity Center. Before doing so, it is necessary to install the Oracle Team Productivity Center Connector for the specific repository on the Oracle Team Productivity Center server. For more information, see the *Installing Oracle Team Productivity Center Server*.

Administrative tasks involving repositories generally begin from the Team Administration dialog.

To access the Repositories tab of the Team Administration dialog:

- **Team Navigator > Team Administration > Repositories**

The Team Administration dialog lets you add, remove, and modify repositories, based on the installed connectors. You may find it useful to add a different server that points to the same repository, a common situation if your teams is distributed throughout the world. Adding a local server for different branches of your organization can improve performance and efficiency for your organization.

Once the Team Administrator has set up repositories, users with administrative privilege can select the repositories used by the team.

3.3.2 How to Find Repositories in the Team Administration Dialog

If you have a large number of repositories, the quick filter feature of the Team Administration menu can let you narrow down the repositories on display, helping you identify the correct repository quickly and easily.

To find a repository with the quick filter feature:

1. Click on the **Repositories** tab of the Team Administration dialog.
2. Type the first few characters of the repository's name into the Filter field at the top of the Repositories column.

   JDeveloper lists all repositories whose names include the characters you type.

The filter searches progressively as you type each character. The filter is not case-sensitive; you can type in lower-case and JDeveloper will display repository names regardless of capitalization.

3.3.3 How to Assign Repositories

You assign repositories by selecting them from the list of added repositories, which in turn is based on the installed connectors on the server. The team administrator adds the repository to the team's implementation of Team Productivity Center; individual

team members access those repositories while checking, tracking, and modifying work items as part of the product development lifecycle.

To determine which repositories a team will be able to access:

- **Team Navigator > Team Administration > Teams > Team Repositories** tab

This displays a list of all available repositories in your installation of Team Productivity Center. Some repositories might require you to enter a parameter (such as a project name for the Microsoft Project connector, or a URL or a port number for versioning repositories).

To make a repository accessible or inaccessible to your team:

1. Check the box beside each repository to make your new team able to access that repository.
2. Remove the check to make the repository inaccessible to your team.
3. To set or clear all boxes, click the topmost box (beside the Name field).

3.3.4 How to Create and Use Versioning Repositories

Creating a versioning repository gives your team a quick way of navigating to your versioning system, through the Repository tab.

To create a versioning repository:

1. Select **Team Administration > Repositories**., then select the Versioning node.
2. Click on the green + sign to create a new repository.

> **Note:** To make sure you do not have an existing repository for the specific version control system, click on the + sign beside the Repositories node. This will display the list of available version control repositories currently available in Oracle Team Productivity Center. If you need to create a new repository for the same version control system, you can give it a new name.

3. Click on the **Connector** drop-down to select the connector for the desired version control system.
4. In the **Name** field, type a name for the repository you are creating. If this is a new repository for a version control system that already has a repository (for example, for a different branch or version of your project), be sure the name is easily identifiable.
5. In the Description field, describe the versioning repository you are creating.
6. Click **OK**.

3.3.4.1 Giving Team Access to Versioning Repositories

Once you have created the versioning repository, you can control which teams have access to it from the **Teams > Team Repositories** tab of the Team Administration dialog.

To give team access to a versioning repository:

1. Select **Team Administration > Teams > Team Repositories** tab.

2. Scroll through the list of teams in the left-hand pane of the dialog to find the team to which you wish to give access to your new repository.
3. Scroll through the list of repositories in the right-hand pane of the dialog to find the repositories you wish to make available to the selected team.
4. Check the box beside each repository you wish to be accessible to the selected team.
5. Click **OK**.

Note that you need to select each team directly, but you can specify all appropriate repositories for that team in a single operation, by selecting all desired repositories and then clicking **OK**.

3.4 Working with Users, Teams and Roles

A key part of the administrator's job concerns managing the team members: maintaining user accounts, keeping users on the right teams, and ensuring they all have the capabilities required to do their jobs. This is especially important in a dynamic environment where new members are joining the product development group as the product matures. Team Productivity Center's administration gives a number of ways of adding and removing users, modifying user information, and managing teams.

Oracle Team Productivity Center uses roles to define the capabilities of each individual team member. As the administrator, you can define role which have a unique combination of capabilities; you then assign different roles to individual members based on their needs in the organization. Each role has its own combination of privileges, which permit team members to create and modify different elements (queries, tags, documents, and more) within Oracle Team Productivity Center.

For more information about managing roles, see Section 3.4.7, "How to Set and Change Team Member Roles."

3.4.1 How to Create Team Productivity Center User Accounts

As an Oracle Team Productivity Center administrator, one of your ongoing tasks will be to create accounts for team members. This involves adding individuals as they join your team, but also making sure that they have the correct access to various features and functions of Team Productivity Center.

An Oracle Team Productivity Center administrator's tasks include the following:

- Adding users: This can be either when you are first setting up your team, or when others join your team later.
- Modifying information for an existing user: This can include contact information, permissions, and other information. You access this through the Users tab of the Team Administration menu (**Team > Team Administration > Users**).
- Changing team member roles: Team members are assigned roles within the team. These roles play an important part in what privileges the team members have when it comes to modifying queries and other team-wide interactions. You access this through the Roles tab of the Team Administration menu (**Team > Team Administration > Roles**).
- Managing repositories: Because the user accounts include granting permission for individual team members to access the various repositories your team uses, you may also be called upon to manage some repository information, such as making

sure that your team members have any accounts and permissions that an individual repository requires. You access this through the Repositories tab of the Team Administration menu (**Team > Team Administration > Repositories**).

These tasks might be performed at various times, either at the beginning of the product lifecycle or during development, or as new members join the team.

3.4.2 How to Add Users to Team Productivity Center

In a typical production environment, your team will grow and develop as the product matures. As an Oracle Team Productivity Center administrator, you can add new users in one of the following ways:

- through the Team Productivity Center Administration dialog
- by importing users from a comma-separated values (CSV) file
- by synchronizing users from an LDAP source.

Each of these is discussed individually.

3.4.2.1 Adding Users through the Team Productivity Center Administration Dialog

You can add users to Team Productivity Center through the Administration dialog. This is a simple method best suited for adding an individual user.

To add a user to Team Productivity Center through the Team Productivity Center Administration Dialog:

1. Open the Team Server Administration by clicking on its icon in the Team Navigator menu (or by selecting **Team > Team Administration**).
2. Click the **Users** tab.
3. Click the green + sign and to open the new user dialog.
4. Complete the user registration information include assigning a password and setting status to Active or Inactive.
5. Set the following user permission options:
   - **Create new team?** Select this option if you want the user to be able to create new teams.
   - **Create new users?** Select this option if you want the user to be able to add additional users to the system.
   - **Is an administrator?** Select this option if the user is to be an Oracle Team Productivity Center administrator. Users who are administrators automatically receive the ability to create users and teams.
6. Click **OK**.

You will notice that your user is now represented in the Users roster. This also means that your new user can be assigned to teams.

3.4.2.1.1 Adding Members to an Existing Team You can add and remove team members to and from existing teams through the Team Productivity Center Administration dialog.

To add a team member:

1. Open the Team Server Administration dialog by clicking on its icon in the Team Navigator menu.
2. Click the **Teams** tab.

3. Click on the team name whose membership you wish to adjust.
4. Click the green + associated with the Team Members panel.
5. Select one or several members from available users.
6. Use Ctrl-Shift to select multiple users; use Shift to select a range of users. To select all users, use the >> symbol.
7. When you have added a user to the team, click on the new member's role and select the appropriate role. This step is crucial in determining what additional privileges this user is to have on the team.
8. Click **OK**.

3.4.2.1.2 Removing Members from Existing Teams As your membership changes and develops over time, you may need to remove members from teams. The Team Navigator menu lets you do this.

To remove a team member:

1. Open the Team Server Administration dialog by clicking on its icon in the Team Navigator menu.
2. Click on the **Teams** tab. A list of teams in the system appears.
3. Click on the name of the team from which you want to remove a user.
4. Click on the user's name in the Team Member's area.
5. Click the red X adjacent to the Team Members panel.

The user will be removed from the team roster.

Note: An alternative to removing a user from a team is to set the user's status to Inactive. For more information, see Section 3.4.4, "How to Remove Users from Team Productivity Center."

3.4.2.2 Adding Users Through a CSV File

If you have a number of users to add, you can save time by importing a comma-separated values (CSV) file with the information for multiple users in a single operation.

You can download a CSV template to ensure that your file is formatted correctly for the import., but essentially the format is:

```
login,email,first name,last name
```

Repeat this structure for each member of the team you plan to add via CSV.

To add users through a CSV file:

1. Open the Team Server Administration by clicking on its icon in the Team Navigator menu (or by selecting **Team > Team Administration**).
2. Click the **Users** tab.
3. Click the dropdown arrow next to the green + sign, then select **Add From CSV**.
4. Click the magnifying glass icon to open a file browser, then browse to the file containing your team members' identification and click **Open**.
5. Enter a default password for the users you are adding, then click **OK**.

The team members whose IDs are contained in the CSV file will be added to Team Productivity Center.

3.4.2.3 Adding Users Through LDAP

If your team uses the Lightweight Directory Access Protocol (LDAP) for coordinating team members, you can sync team member information using LDAP.

When you check the **Enable LDAP** box on the LDAP tab of the Team Administration dialog, the title bar above the users list changes from the Add Member button to a sync button, Add from LDAP. Note that the first time you check the Enable LDAP box, you will need to click OK to close the Team Administration dialog and then reopen it for the sync button to be enabled.

To add users through LDAP:

1. From the Users tab of the Team Administration dialog, click the Add from LDAP button at the top of the Users pane.
2. In the Filter field, enter a string that identifies the user or users you wish to add.
3. Click **OK** to add the selected user or users to Team Productivity Center.

3.4.3 How to Modify User Information

If you have administrative privileges, or Create User permission, you can modify information related to any user in the system.

To modify user information:

1. Open the Team Server Administration dialog by clicking on its icon in the Oracle Team Productivity Center masthead.
2. Click on the **Users** tab.
3. Click the name in the Users roster.
4. Modify the information as required.
5. Click **OK**.

> **Tip:** Once a user is assigned to one or more teams, his or her teams will be listed below the user configuration information in read-only form. You can modify team assignments through the Teams administrative tab.

3.4.4 How to Remove Users from Team Productivity Center

Users can only be removed from Oracle Team Productivity Center during the initial session in which they are added. Once a user has been added to the database, it is no longer possible to remove that user. This is because there is user information in Oracle Team Productivity Center that cannot be removed (such as team membership, tags, and queries) associated with the user's account.

If you have a user who no longer participates in Team Productivity Center, the solution is to change that user's status from Active to Inactive.

To change a user's status to Inactive:

1. Select **Team Navigator > Team Navigator menu > Team Administration**.
2. Click on the **Users** tab.

3. Click on the name of the user in the user roster.
4. Click on the Status drop-down list and select **Inactive**.
5. Click **OK**.

You must have Administrator privilege to perform this procedure.

You can use the same procedure to change a user's status back to Active. Once you have selected the user's name from the Team Administration menu, select Active from the Status drop-down list and then click **OK**.

3.4.4.1 Showing Inactive Users

You can also change the setting to view active or inactive users by selecting the **Show Inactive Users** check box.

To show inactive users:

- Click the check box labeled **Show Inactive Users**.

3.4.5 How to Find Users or Teams in the Team Administration Dialog

If you have a large team with many members, you can use the quick filter feature of the Team Administration menu to narrow down your search to simplify finding an individual member.

To find a user with the quick filter feature:

1. Click on the **Users** tab of the Team Administration dialog.
2. Type the first few characters of the user's name into the Filter field at the top of the Users column.

   JDeveloper displays the users with matching characters in first name or last name.

The filter searches progressively as you type each character. The filter is not case-sensitive; you can type in lower-case and JDeveloper will display user names regardless of capitalization. The quick filter also works to search through the entire list of teams.

3.4.5.1 Finding Teams

Finding teams also uses the quick filter feature to let you browse among your existing teams for a specific name, or to select from teams with similar names.

To find a team with the quick filter feature:

1. Click on the **Teams** tab of the Team Administration dialog.
2. Type the first few characters of the team's name into the Filter field at the top of the Teams column.

   JDeveloper displays all teams whose names begin with the characters you type.

As with the Users filter, the Teams filter is progressive. For example, typing the characters T, E and S will display a list of all teams with the word "Test" in their names.

The filter is not case-sensitive; continuing with the "Test" example, JDeveloper will return teams with the words TEST, Test and test in their names.

3.4.6 How to Add and Remove Teams from Oracle Team Productivity Center

In Team Productivity Center, users can be grouped into any arbitrary arrangement of teams and sub-teams; members of each team can be assigned to one of the predefined roles, or you can create new roles if your work environment requires it.

Teams can only be removed from Oracle Team Productivity Center during the initial session in which they are added. Once the team has been added to the database, it is no longer possible to remove them.

If you have administrative privileges, you can add teams through the Team Productivity Center Administration Dialog. With Team Administrator privileges, you can adjust team hierarchies. Team Administrators can make changes to teams they have created. If the Group Administrator has Create Team or Team Administrator privileges, then it is possible to modify the team to which the administrator belongs, and any child team of the Group Administrator's team.

> **Tip:** While a team roster can be duplicated under another team name, a team cannot have more than one parent team.

To add a team to Oracle Team Productivity Center:

1. Select **Team Navigator menu > Team Administration**.
2. Click the **Teams** tab. A list of teams and subteams, if any, appears.
3. Click on a team name. If you choose the top entry in the hierarchy, your team will be a top-level team. Otherwise, click on the name of any existing team to create the new team as its child. (After a team has been created, you can easily adjust its location in the team hierarchy.)
4. Click the green + symbol. A team named "untitled" appears.
5. Provide the following information:
   - Team name. (Use a name consistent with the naming style of other teams in the system.)
   - Parent. If the team you are creating is a sub-team of an existing team, select a parent. Otherwise select the blank option to create a team that is a peer of the primary teams in the system.
   - Status (active or inactive).
   - Optional description.
6. Click **OK** to create the team.

Once you add members to the team, their names will appear in the Team Members panel. Be sure to select roles for each team member as you add them to the team.

3.4.6.1 Making Teams Inactive

Teams in Oracle Team Productivity Center cannot be removed, because there is data (such as team members, tags, and queries) associated with their account. If you have a team which no longer participates in Oracle Team Productivity Center, the solution is to change that team's status from Active to Inactive.

To make a team inactive:

1. Select **Team Navigator menu > Team Administration**.
2. Click on the Status drop-down list and select **Inactive**.
3. To verify the change in status for a team made inactive, click on another team.

4. Click **OK**.

The team made inactive will only disappear for the display when the Team Administration Dialog is closed and re-opened, or when the Show Inactive team is checked and unchecked once.

3.4.6.2 Showing Inactive Teams

To see all teams that you have already made inactive, use the Team Navigator menu.

To show inactive teams:

- Click on the box labeled **Show Inactive Teams**.

This displays all teams that you have made inactive.

3.4.7 How to Set and Change Team Member Roles

Roles are the mechanisms that Oracle Team Productivity Center uses to manage users' privileges to create and modify various elements (work items, queries, tags, and documents such as attachments) within the team working environment. While Oracle Team Productivity Center has default roles, the administrator can also create and define new roles, choosing from the privileges listed in Table 3–1.

The three roles with which Oracle Team Productivity Center is delivered are:

- **Team Member** (the default for all users). Team Members can access and edit work items, and can also create and manage queries, tags, and documents for their own use.
- **Team Administrator.** Team Administrators have all Team Member privileges, but in addition they can create and manage queries, tags, documents and sources for use by all members of their team.
- **Group Administrator**. Group Administrators have all the privileges of Team Members and Team Administrators, but also have the same privileges on any child teams that are contained within their own team.

In addition to these roles with which Oracle Team Productivity Center, administrators can create new roles which contain a combination of the available privileges. This gives you great flexibility to create specific roles that match the desired capabilities, privileges and responsibilities of team members.

Table 3–1 lists the privileges available when creating a new role:

Table 3–1 Available Privileges for Role Creation

Privilege	Description
Administer Teams	Perform all administrative duties on their own team. Available by default to the Team Administrator and Group Administrator roles.
Manage Team Queries	Create, manage and delete queries into data repositories for use by all members of their own team. Available by default to the Team Administrator and Group Administrator roles.
Manage Team Tags	Create, manage and delete tags for use by members of their own team. Available by default to the Team Administrator and Group Administrator roles.
Manage Team Documents	Create, manage and delete documents for use by members of their own team. Available by default to the Team Administrator and Group Administrator roles.

Table 3–1 (Cont.) Available Privileges for Role Creation

Privilege	Description
Manage Team Sources	Create, manage and delete sources for use by members of their own team. Available by default to the Team Administrator and Group Administrator roles.
Gain Privilege Over Child Teams	Users assigned a role with this privilege will have this same role on every team in the hierarchy below the current one, whether or not they are explicitly a member of those teams. This privilege is intended to allow certain users to administer all teams within a hierarchy without being a member of every single team. The Team Productivity Center administrator can change the allocation of privileges between these three roles, and can also create a new role with a different mix of privileges if required.

To create a new role:

1. Select **Team Navigator menu > Team Administration**.
2. Click on the **Roles** tab, and then click on the green plus sign. This creates a new role.
3. Type a name and a description for the new role you are creating.
4. Select the combination of privileges that you wish to associate with the new role, and then click **OK**.

Once you have created a new role, you can assign it to individual team members by modifying their role on a team of which they are a member.

3.4.7.1 Changing a Team Member's Role

In addition to creating a new role (which you can subsequently assign to users as they join your team), you can also change the role of any member of your team, selecting from any of the default roles or any new roles you have created.

To change the role of a team member:

1. Select **Team Navigator menu > Team Administration**.
2. Click on the **Teams** tab. A list of teams in the system appears.
3. Click on the name of the team that contains the user or users whose roles you wish to change.
4. Click on the user's name.
5. Click on the user's current role and select a new role from the dropdown list, and then click **OK**.

You can change the role of a team member at any time. After the change, the member will have the capabilities of the new role you have assigned them.

4

Working with Oracle Team Productivity Center Connectors

This chapter describes how to develop connectors between Oracle Team Productivity Center and external data repositories, such as bug databases, feature tracking systems, and any other repository of data used by a team for development. It also describes how to work with Oracle Team Productivity Center connectors after you have created them. This includes error handling, adding help, packaging the connectors for distribution, and internationalization.

Oracle Team Productivity Center connectors provide a framework to integrate third-party repositories with JDeveloper. Standard work item interfaces allow third-party connectors to fetch data from their backend repositories and present them in a standard format. The declarative user interface approach gives connector writers the freedom to lay out UI controls in many flexible ways, without writing an extensive amount of code. JDeveloper then combines the data and UI elements at runtime, and presents repository objects in a coherent way that JDeveloper users already are familiar with.

This chapter includes the following sections:

4.1 About Working with Oracle Team Productivity Center Connectors

Developing your own connectors for Team Productivity Center begins with an understanding of connector architecture. The architecture of an Oracle Team Productivity Center connector is organized to provide three primary functions:

- **Data fetching from the work item repository served by the connector.** Each work item connector implements the `WorkitemConnector` interface, and optionally the `WorkitemAttachment` interface if the repository for this connector supports attachments. The `WorkitemConnector` interface specifies methods for the end user to log into and out of the repository, basic Create, Read, Update and Delete (CRUD) operations on work items, and retrieving query results. Based on the capabilities of the backend repository, the connector writer determines what `WorkitemConnector` methods to implement to expose the backend repository

functionalities. When implementing these selected methods, connector writers use the appropriate APIs (or Web services) available from the backend repository. After data is collected through the APIs, it needs to be converted to the work item data structure that the connector writer defines.

- **Generating the runtime UI component tree inside the Team Productivity Center client.** After the connector obtains the work item data from the repository, Oracle Team Productivity Center's declarative UI framework reads the work item UI definition defined by the connector writer and binds the data with the corresponding UI controls. The Oracle Productivity Center provides a metadata-driven UI framework in which connector writers can develop a work item UI layout XML file by using the pre-defined UI tags. Preferably, the connector writer authors a UI layout that conforms to JDeveloper look and feel, while maintaining the behaviors of its native application as much as possible. This way the end users will have the same experience when navigating the UI in JDeveloper as well as in its native application.
- **Displaying UI controls and data in the work item editor that ties the repository data to UI components.** When both work item data and its UI component tree are available, the WorkitemEditor, a special editor inside JDeveloper, binds the data to the UI controls and displays them inside the editor.

Oracle Team Productivity Center supports a dynamic work item model and UI at runtime. For certain repositories, the work item model and/or UI are determined by some work item instance field value(s) or a pre-defined condition. Oracle Team Productivity Center provides the flexibility to show a different UI layout at runtime by exposing methods in the `WorkitemConnector` interface.

4.1.1 Team Productivity Center Connector Execution Flow Example

To understand the way the preceding information fits together in practice, consider the following example using the JIRA repository:

You double-click on a work item that represents a query titled "My P1 Issues." JDeveloper opens the query, displays the query criteria and executes the query to show the results in a list. You can then double-click on a particular issue in the result list; JDeveloper opens a new work item editor and shows the selected issue's details. Here is the call sequence represented by this example:

1. UI framework gets the query information from the TPC database for "My P1 Issues."
2. UI framework finds the JIRA connector instance and calls the method `getQueryResults` on the JIRA connector and sends in the query criteria.
3. JIRA connector returns a list of Work Item objects that match the specified criteria.
4. TPC framework shows the list of work items in the results list of the query.
5. When the user double clicks on particular work item, UI framework calls the `getWorkitem()` method on the JIRA connector and passes the issue ID.
6. Inside the JIRA connector instance, `getWorkitem()` retrieves the issue data.
7. When the SOAP interface returns the data, `getWorkitem()` converts it to a JIRA Work item object.
8. UI framework creates a work item editor.
9. The work item editor now loads the issue UI page by calling `getUIRegionName()` and then generating the UI component tree.

10. The work item editor then binds the data to their UI controls.
11. The work item editor, through different renderers, displays the issue data and their associated UI controls.

4.2 Creating Oracle Team Productivity Center Connectors

The Team Productivity Center connector begins as a JDeveloper extension project. It uses a configuration file (in XML) to store and manage parameters for creating the connector instance. The connector also defines the data used by the repository so that JDeveloper can interpret it, retrieves data from the repository, and uses the Team Productivity Center UI for presenting data within JDeveloper. The remaining sections provide more details on each task and show sample code for implementing the `WorkItemConnector` interface.

4.2.1 How to Create an Oracle Team Productivity Center Connector

Team Productivity Center connectors are created as JDeveloper extensions. This allows them to integrate with JDeveloper and also provides a framework for packaging and distributing the connector through the JDeveloper Check for Updates feature, once the connector development is completed. This provides a well-known mechanism for distributing connectors to your development team.

The following steps provide an overview of the process of creating an Oracle Team Productivity Center connector. Details of many of the procedures listed here are available in the Oracle® Fusion Middleware User Guide for Oracle JDeveloper.

To create an Oracle Team Productivity Center connector:

1. Create a JDeveloper Extension Project to build the connector: select **File > New > All Features > Client Tier > Extension Development**.
2. Write the connector configuration XML file. You need to provide connector configuration parameters to be used by the Team Productivity Center installer. The parameters will be treated as seed data and entered into the Team Productivity Center database. They will be used at runtime when creating an instance of the connector.

   For more information, see Section 4.2.3, "How to Create an Oracle Team Productivity Center Connector Configuration File."
3. Determine what object types will be exposed to Oracle Team Productivity Center. For example, the writer of an MS Project Server connector may decide to show task, project, and resource data, so there will be three object types.
4. Write the repository object definition XML file. This file provides field details on each object type, default label, default UI control type to use, whether it is required, etc.

   For more information, see Section 4.2.2, "How to Define Team Productivity Center Repository Data."
5. If you are using specific listeners, add them to the `workitem.properties` file and make sure that file is included in the connector bundle.
6. Write the connector UI layout XML file. This lets you lay out the UI by using UI tags provided by the Team Productivity Center declarative UI framework.
7. Create a class that implements `WorkItemConnector` and (optionally) `WorkItemAttachment` interface methods. This class will use the appropriate APIs available from the backend repository to establish a connection, send and

receive repository data, and convert the data to work item format that Team Productivity Center uses.

8. Provide context-sensitive help files. The help file is loaded by the JDeveloper Help Center when the user presses F1 with the work item detail UI in focus.

   For more information, see Section 4.4.1, "How to Add Help to the Team Productivity Center Connector."

9. Provide resource bundle files for multi-language support.

   For more information, see Section 4.5.4, "How to Generate the Connector Bundle File."

10. Package the connector into the appropriate ZIP format for deployment.

The remaining sections provide more details on each task and show sample code for implementing the `WorkItemConnector` interface.

4.2.2 How to Define Team Productivity Center Repository Data

In Example 4–1, you define the XML data objects to hold the various types of objects inside your repository. The name of this file must be specified in the `connector.xml` file as the `modelFileName`. You create this model file and put it in under your project in the `src/META-INF` folder.

The model definition file specifies the fields to expose in the Team Productivity Center, default columns to show in a query result list, and list of value definitions and the data source to use for getting data.

Example 4–1 is an example work item definition file (`SampleDef.xml`).

Example 4–1 Sample Work Definition File

```
<?xml version="1.0" encoding="windows-1252"?>
<RepositoryModel resName="res" resFile="/META-INF/res/modelresource.xml">
        <WorkItem data-source="rpthead" id-def="TASKID" id-label="${res.TASK_ID}"
label-def="TASKID"
                  name="${res.TASK_NAME}" type="Task" subject-def="DESC"

webURLHandler="oracle.sampleconnector.model.TaskWebReferenceImpl"
                  supportSearchByID = "true"
                  xmlns="http://www.oracle.com/alm" version="1.1.1.1">
                <Fields>
                        <Field name="TASKID" label="${res.TASK_ID}" type="number"
                               readOnly="true"/>
                        <Field name="DESC" label="${res.TASK_DESC}"
required="true"
                               maxLength="80" type="string"/>
                        <Field name="OWNER" label="${res.TASK_OWNER}"
                               defaultValue="" controlType="lov" type="string"
                               lovDef="owner_lov"/>
                        <Field name="STATUS" label="${res.TASK_STATUS}"
type="number"
                               required="true" controlType="choice"
                               lovDef="statusLookUp"/>
                        <Field name="DUEDATE" label="${res.TASK_DUEDATE}"
type="date"
                               readOnly="true"/>
                        <Field name="URL" label="${res.TASK_URL}" maxLength="255"
                               type="string"/>
                </Fields>
```

```
                <QueryListColumns>
                        <FieldRef name="TASKID"/>
                        <FieldRef name="DESC"/>
                        <FieldRef name="OWNER"/>
                        <FieldRef name="STATUS"/>

                </QueryListColumns>
                <WebResource>
                        <URLDef name="URL" anchorOn="label"/>
                        <URLDef name="TASKID" anchorOn="field"/>
                </WebResource>
                <LovDefs>
                        <LovDef name="owner_lov" list-source="users">
                                <CriteriaMap>
                                        <Map listFieldRef="USERID"
                                             fieldRef="OWNER"/>
                                </CriteriaMap>
                                <FieldMap>
                                        <Map listFieldRef="USERNAME"
                                             fieldRef="OWNER"/>
                                </FieldMap>
                                <DisplayList>
                                        <FieldRef name="USERID"/>
                                        <FieldRef name="USERNAME"/>
                                </DisplayList>
                        </LovDef>
                        <LovDef name="statusLookUp" list-source="status">
                                <FieldMap>
                                        <Map listFieldRef="SID"
                                             fieldRef="STATUS"/>
                                </FieldMap>
                                <DisplayList>
                                        <FieldRef name="SID"/>
                                        <FieldRef name="SDESC"/>
                                </DisplayList>
                        </LovDef>
                </LovDefs>
                <DataSources>
                        <DataSource name="users" id-def="USERID">
                                <Field name="USERID" label="${res.USER_ID}"
                                       type="string"/>
                                <Field name="USERNAME" label="${res.USER_NAME}"
                                       type="string"/>
                        </DataSource>
                        <DataSource name="priority" id-def="PID">
                                <Field name="PID" type="number"/>
                                <Field name="PDESC" type="string"/>
                        </DataSource>
                        <DataSource name="status" id-def="SID">
                                <Field name="SID" type="number"/>
                                <Field name="SDESC" type="string"/>
                        </DataSource>
                </DataSources>
        </WorkItem>
</RepositoryModel>
```

Table 4–1 lists the tags used and their descriptions.

Table 4–1 Repository Data Configuration Tags

Configuration Tag	Attributes	Description
RepositoryModel		Encloses all types of work items used for this connector.
	resName	Name of the resource bundle. It is used as a name space to identify runtime labels based on the symbols specified.For example, `label="${res.ISSUE_PROJECT}"` will be resolved in runtime to "Project" if `EN` is the current JDeveloper locale, if `ISSUE_PROJECT` is defined as "Project" in `EN`.
	resFile	Path of the resource bundle file for the connector.
Workitem	data-source	Refers to backend repository source. It could be a database table.
	id-def	Field use to uniquely identify a work item. In most cases, it is the work item's `ID` or `NO`.
	subject-def	Field used as a Subject field during tagging a work item or creating relationship between work items.
	Type	The work item's type or category. Since you can specify multiple work items with different type in the connector model definition file, the work item's type is used to construct a URL editor for the work item at runtime. The type is also showed in certain UI elements, like right mouse menu items in various work item's pages.
	xmlns	XML name space.
	version	Version of the model definition file.
Fields		Encloses all fields used for the repository object
Field	name	Name of the field.
	label	Display label used for the field in various work item's pages, such as detail UI and query UI.
	type	Data type of the field. Valid values are "`number`", "`string`", "`date`". Default = `string`.
	required	If set to true, an asterisk (*) will show up before the field label to indicate the field is a required field. Default = false.
	controlType	For information on which UI controls to use., see Section 4.2.6, "How to Present Data Declaratively with the Team Productivity Center UI."
	lovDef	Name of an LOV definition to use if the `controlType` is "`lov`". See `LovDefs`.
	readonly	If set to true, the corresponding control is grayed out and the field value is in read only mode. Default = false.
	queryable	If set to `true` this field will appear inside the field combo box on the query form. Default = `true`.
	maxLength	This sets the maximum characters that can be entered into the field.
LovDefs		Encloses all List of Value (LOV) definitions used for the repository fields.
LovDef		Definition of an LOV.
	name	Name of the LOV.

Table 4–1 (Cont.) Repository Data Configuration Tags

Configuration Tag	Attributes	Description
	list-source	Name of a data source to use for retrieving its data. See DataSource.
DataSources		Encloses all data sources used by LOVs.
DataSource		Definition of a data source.
	name	Name of a data source.
	id-def	Object type or table name in the backend repository.
QueryListColumns		List of default columns to show in the query result list.
WebResource		Section that describes the fields that are hyperlinked.
URLDef		A child of `WebResource`. Describes each field that can be a hyperlink.
	name	The name of the field from the field list.
	anchorOn	Whether the underlined hyperlink should be on the label or the actual value. Valid values are "`label`" or "`field`". Default = `field`.

4.2.3 How to Create an Oracle Team Productivity Center Connector Configuration File

The connector configuration XML file contains the parameters used when creating a runtime connector instance. The parameters can be set at the team level or at the server level. Oracle Team Productivity Center Installer reads in this configuration file during installation and populates seed data to Team Productivity Center database tables.

The connector.xml file will need to be created so that you can add parameters for connector that the team leader or administrator can define values for.

To create the connector.xml file:

1. Right-click on your extension project in the Application Navigator.
2. Select **New > XML > XML Document**.
3. Name this file connector.xml and save it under your project in the src/META-INF folder.

This configuration file uses tags defined the connector.xsd file, which is located in the `almcommon-api.jar` file. You can find this JAR file under your JDeveloper install folder:
`\jdeveloper\jdev\extensions\oracle.teamproductivitycenter\lib`

Example 4–2 contains an example of a configuration file:

Example 4–2 Configuration file sample code

```
<?xml version="1.0" encoding="UTF-8"?>
<ConnectorDefinition xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xsi:schemaLocation="http://www.oracle.com/alm connector.xsd"
                     xmlns="http://www.oracle.com/alm">
  <configuration id="Sample"
                 className="oracle.sampleconnector.model.SampleConnectorService"
                 repositoryType="Work Item"
                 uiFileName="/META-INF/SampleUI.xml"
                 modelFileName="/META-INF/SampleDef.xml"
```

```
                version="1.0">
    <parameter>
      <name>SampleParam1</name>
      <defaultValue>Default value 1</defaultValue>
      <label>
        <locale>US</locale>
        <value>Sample Param 1</value>
      </label>
      <description>This is the sample admin param one</description>
      <accessLevel>ALM-Admin</accessLevel>
    </parameter>
    <parameter>
      <name>SampleParam2</name>
      <defaultValue>Default value 2</defaultValue>
      <label>
        <locale>US</locale>
        <value>Sample Param 2</value>
      </label>
      <description>This is the sample admin param two</description>
      <accessLevel>ALM-Admin</accessLevel>
    </parameter>
    <parameter>
      <name>SampleTeamParam</name>
      <defaultValue>Sample Team Param</defaultValue>
      <label>
        <locale>US</locale>
        <value>Sample Team Param</value>
      </label>
      <description>This is a sample team parameter</description>
      <accessLevel>Team-Admin</accessLevel>
    </parameter>
  </configuration>
</ConnectorDefinition>
```

Table 4–2 lists the tags used and their descriptions.

Table 4–2 Tags used in connector.xml

Configuration Tag	Attributes	Description
configuration	id	Name of the connector
	className	Connector class to load when creating a connector instance during runtime.
	uiFileName	Path to a file inside the connector jar file that contains the connector UI definitions
	modelFileName	Path to a file inside the connector jar file that contains the connector model definitions
	version	Current connector version. Any new/upgrade connector package should use a new version.
parameter		Specifies each of configurable parameters
name		Parameter name
label		Display text to use in the Team Administration UI (or Admin UI)
locale		Localization support
value		Value for the specified locale

Table 4–2 (Cont.) Tags used in connector.xml

Configuration Tag	Attributes	Description
description		A brief description of what this parameter does. It will be displayed in the Admin UI
accessLevel		ALM-Admin - indicates that this parameter can be configured only by a TPC Administrator in the **Admin UI > Repositories** tab
		Team-Admin - indicates that this parameter can be configured by a Team Administrator in the **Admin UI > Teams > Repositories** tab

4.2.4 How to Use Customized Listeners and Managed Beans

Developers of connectors for Team Productivity Center can provide connector-specific functionalities by customizing the work item UI control behavior.

Team Productivity Center provides a connector level configuration file, `tpc-config.xml`, for registering all metadata resources. This file should be put under `src/META-INF/` in the connector source code directory.

For details on the tags and attributes supported in the configuration file, see the *Tag Reference for Oracle Team Productivity Center Connectors*.

To write a custom managed bean:

When you write a connector for Team Productivity Center, you can write a customized managed bean for any value binding support, as shown in the following steps.

1. Register a managed bean entry in the file `tpc-config.xml`, as shown in Example 4–3:

Example 4–3 Register a Managed Bean Entry

```
<managed-bean>
    <name>labelBean</name>
<impl-class>oracle.alm.sample.resbean.LabelBean</impl-class>
<lifecycle>page</lifecycle>
</managed-bean>
```

2. Implement the `LabelBean`, as shown in Example 4–4:

Example 4–4 LabelBean Implementation

```
public class LabelBean
{
   public LabelBean(){}
   public String getLabel()
      {
   return _label;
  }
   public  void setLabel(String label)
    {
  _label = label;
}
   String _label = "test label";
}
```

3. Use the managed bean in either the work item definition metadata file (for a work item field, for example), or a label attribute for a UI control in the UI metadata file:

```
label="${labelBean.label}"
```

4.2.4.1 Writing a Customized UI Listener

You can also write a customized UI listener and register it for a control in the work item UI metadata by performing the following steps:

1. Register the listener in `tpc-config.xml` as shown in Example 4–5:

Example 4–5 Registering the Listener

```
<managed-bean>
  <name>opentask </name>
  <impl-class>oracle.alm.sample.view.OpenTaskListener </impl-class>
  <lifecycle>page </lifecycle>
</managed-bean>
```

2. Implement `OpenTaskListener` class:

Example 4–6 Implementing Class OpenTaskListener

```
package oracle.alm.sample.view;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import java.util.List;

import oracle.alm.connector.data.WorkItem;
import oracle.alm.view.application.ViewManager;
import oracle.alm.view.context.AlmELContext;
import oracle.alm.view.model.AlmDataTableModel;
import oracle.alm.view.uicomponents.AlmComponent;
import oracle.alm.view.uicomponents.AlmContextMenuComponent;
import oracle.alm.view.uicomponents.AlmTableComponent;
import oracle.alm.view.uicomponents.RenderingContext;

public class OpenTaskListener
  implements ActionListener, AlmScope
{
  public OpenTaskListener(RenderingContext rcontext, AlmComponent component)
  {
   super();
   _rcontext = rcontext;
   _component = component;
  }

  public void actionPerformed(ActionEvent e)
  {
    if (_component instanceof AlmContextMenuComponent)
    {
      AlmComponent parent = _component.getParent();
      if (parent instanceof AlmTableComponent)
      {
        AlmTableComponent tableComp = (AlmTableComponent) parent;
        List<Integer> rows = tableComp.getSelectedRows();
        if (rows != null)
        {
```

```
          ViewManager vmanger = ViewManager.getInstance();
          for (int row: rows)
          {
            AlmELContext elcontext = vmanger.getELContext(_rcontext);
            AlmDataTableModel dataModel = tableComp.getValue(elcontext);
            Object selectedValue = dataModel.getSelectedItem(row);
            if (selectedValue instanceof WorkItem)
            {
              String wiType = _rcontext.getCurrentWorkItemType();
              String reposName = _rcontext.getCurrentReposName();
              String reposId = _rcontext.getCurrentReposId();
              vmanger.OpenWorkItemInEditor(((WorkItem) selectedValue),
                                           wiType, reposName, reposId);
            }
          }
        }
      }
    }
  }
  public void setRenderingContext(RenderingContext rc){_rcontext = rc};
  public RenderingContext getRenderingContext(){return _rcontext};
  public void setSourceComponent(AlmComponent component) {_component = component};
  public AlmComponent getSourceComponent() {return _component};
  private RenderingContext _rcontext;
  private AlmComponent _component;}
```

3. Use the customized listener in the work item UI metadata:

Example 4–7 Using customized listener in a work item

```
<?xml version="1.0" encoding="windows-1252" ?>
<regions resFile="/META-INF/res/uiresources.xml">
<region id="Default" helpTopicId="f1_connector_sample_htm">
    <formLayout columns="1" blockSize="10" fieldWidth="400">
      <inputText label="#{workitemmodel.labels.TASKID}"
             value="#{workitemmodel.values.TASKID}" readOnly="true"/>
      <inputText label="#{workitemmodel.labels.DESC}"
             value="#{workitemmodel.values.DESC}"/>
      <listOfValues label="#{workitemmodel.labels.OWNER}"
             value="#{workitemmodel.values.OWNER}"
             source="OWNER" pprTargets="statusID"/>
      <comboBox label="#{workitemmodel.labels.STATUS}"
             value="#{workitemmodel.values.STATUS}"
             valueSet="#{workitemmodel.listItems.STATUS}"
             readOnly="true" id="statusID"/>
      <inputDate label="#{workitemmodel.labels.DUEDATE}"
             value="#{workitemmodel.values.DUEDATE}"/>
      <inputText label="#{workitemmodel.labels.URL}"
             value="#{workitemmodel.values.URL}"/>
    </formLayout>
    <panelLayout>
      <action text="Open Task" actionListener="${opentask}"/>
    </panelLayout>
  </region>
</regions>
```

4.2.4.2 Customizing Context Menu Actions

You can add customized menu items when the user clicks the right mouse button on the query result table. Perform the following steps:

1. Register the listener in `tpc-config.xml` and specify the menu item details:

Example 4–8 Register the Listener, With Details

```
<managed-bean>
<name>savetask</name>
<impl-class>oracle.alm.sample.model.SaveTaskListener</impl-class>
<lifecycle>page</lifecycle>
</managed-bean>
<contextMenuDef>
<menuItemDef label="Save" listenerRef="savetask"/>
</contextMenuDef>
```

2. Implement `SaveTaskListener` class:

Example 4–9 Implementing class SaveTaskListener

```
package oracle.alm.sample.model;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;
import javax.swing.JOptionPane;
import oracle.alm.connector.data.WorkItem;
import oracle.alm.view.application.ViewManager;
import oracle.alm.view.context.AlmELContext;
import oracle.alm.view.model.AlmDataTableModel;
import oracle.alm.view.uicomponents.AlmComponent;
import oracle.alm.view.uicomponents.AlmContextMenuComponent;
import oracle.alm.view.uicomponents.AlmTableComponent;
import oracle.alm.view.uicomponents.RenderingContext;

public class SaveTaskListener implements ActionListener, AlmScope
{
  public SaveTaskListener(RenderingContext rcontext, AlmComponent component)
  {
   super();
   if (component instanceof AlmContextMenuComponent)
         _almComponent = (AlmContextMenuComponent) component;
 _renderingContext = rcontext;
   }

   public void actionPerformed(ActionEvent e)
   {
    ViewManager vmgr = ViewManager.getInstance();

//First get the corresponding AlmTableComponent
AlmComponent parent = _almComponent.getParent();
if (parent instanceof AlmTableComponent)
{
   AlmTableComponent tableComp = (AlmTableComponent) parent;
//Then call public API on tableComp to get selected rows
List&l;Integer> rows = tableComp.getSelectedRows();

if (rows != null)
 {
  int row = rows.size();
  String cs = "";
  if (row == 1)
  {
```

```
    //show the ID and subject of the selected item
//Then get the selected value object throw the tableComp's UI model.
ViewManager vmanger = ViewManager.getInstance();
AlmELContext elcontext = vmanger.getELContext(_renderingContext);
AlmDataTableModel dataModel = tableComp.getValue(elcontext);
Object selectedValue = dataModel.getSelectedItem(row);
if (selectedValue instanceof WorkItem)
{
     cs = "Save Task listener: work item ";
 WorkItem wi = (WorkItem)selectedValue;
 cs += wi.getRowKey() + " with subject \'";
 cs += wi.getSubject() + "\' has been selected.";
}
  }
  else if (row > 1)
  {
    //show the number of rows selected
cs = "Save Task listener: " + Integer.toString(row) + " rows selected";
}
JOptionPane.showMessageDialog(vmgr.getIDEMain(), cs);
  }
}
  }
public void setRenderingContext(RenderingContext rc){_ renderingContext = rc};
public RenderingContext getRenderingContext(){return _ renderingContext };
public void setSourceComponent(AlmComponent component) {_almComponent =
component};
public AlmComponent getSourceComponent() {return _almComponent };

AlmContextMenuComponent _almComponent;
RenderingContext _renderingContext;
}
```

4.2.5 How to Access Connector Data Through a Firewall

If the connector needs to use the HTTP protocol to fetch data through a firewall that JDeveloper is running behind, the following special modifications are required.

1. In JDeveloper, select **Tools > Preferences**.
2. In the Preferences dialog, open the Web Browser and Proxy node.
3. Check the **Use Proxy** check box and enter the values for **Host Name**, **Port**, and **Exceptions**.
4. The connector writer needs to set the proxy configuration for its `HttpClient` inside the connector code where it needs to fetch data (for example, inside the `getQueryResult()` API).

Example 4–10 Set Proxy Configuration Inside Connector Code

```
HttpClient httpclient = new HttpClient();
String proxyHost = System.getProperty("http.proxyHost");
String proxyPort = System.getProperty("http.proxyPort");
Boolean useProxy = null;

if (proxyHost != null && !proxyHost.isEmpty()&&
    proxyPort != null && !proxyPort.isEmpty())
{
  String proxyExceptions = System.getProperty("http.nonProxyHosts");
  useProxy = (proxyExceptions == null);
```

```
    if (!useProxy)
    {
      String host = "";
      try
      {
        URL url = new URL(getServerURL());
        host = url.getHost();
        useProxy = true;
      }
      catch (Exception e)
      {
        e.printStackTrace();
      }

      proxyExceptions = proxyExceptions.replace("*", ".*");
      useProxy = (Pattern.matches(proxyExceptions, host) == false);
    }
  }

  if (useProxy != null && useProxy.equals(Boolean.TRUE))
        httpclient.getHostConfiguration().setProxy(proxyHost,
Integer.parseInt(proxyPort));
```

4.2.6 How to Present Data Declaratively with the Team Productivity Center UI

Oracle Team Productivity Center has a framework that supports declarative UI for any connector. Connector writers can use the UI tags defined in `workitem-ui.xsd` to lay out the work item UI. Basic UI elements like input text, radio buttons, combo box, checkbox, pick list or List-Of-Values (LOV), are all supported.

This step defines the XML data objects to hold the UI definition in order to show your repository objects to the user. The name of this file must be specified in the connector.xml file as the `uiFileName`. You can create this UI file and put it in under your project in the `src/META-INF` folder.

`workitem-ui.xsd` is located in the file `almcommon-api.jar`.

By using the pre-defined `controlType` defaults in the model definition, the UI can be laid out easily by simply using a `formLayout` component that automatically picks up the control tags from the model. This section shows several examples of how this can be done.

Example 4–11 Simple Method for Displaying Work Item UI Page

```
<?xml version="1.0" encoding="windows-1252" ?>
<regions resFile="/META-INF/res/uiresources.xml">
  <region name="Easy" helpTopicId="f1_connector_sample_htm">
    <formLayout value="${workitemmodel}" columns="1" blockSize="10"/>
  </region>
</regions>
```

Table 4–3 UI tags used in preceding example

UI Tag	Attributes	Description
Regions		Encapsulation of all UI regions used
	resFile	Specifies the resource bundle file to use for multi language support
Region		One page layout identified by its name attribute

Table 4–3 (Cont.) UI tags used in preceding example

UI Tag	Attributes	Description
	helpTopicId	Tells JDeveloper which help file name to use (from within the connector jar file)
formLayout	value	It has value of "`${workitemmodel}`", which will be resolved at runtime to an appropriate connector model.
	columns	This sets the number of columns to lay out the form into.
	blockSize	This means after every blockSize rows a line separator will be added
	rowToSpan	Tells which row(s) from the model will span to the entire row of the page. If it is "1, 3, 5" then the 1st, 3rd, and 5th rows will span the entire row of the page.

4.2.6.1 Using an Explicit Layout to Show a UI Page

This example shows the same UI as in Example 4–11, but uses an explicit layout instead. This is necessary if you have multiple UI views of the same data model. For example, you would like only a subset of the fields to be visible when creating a new instance of an object but as soon as it is saved, all the fields become visible.

Example 4–12 Explicit Layout for UI page

```
<?xml version="1.0" encoding="windows-1252" ?>
<regions resFile="/META-INF/res/uiresources.xml">
  <region name="Default" helpTopicId="f1_connector_sample_htm">
    <formLayout columns="1" blockSize="10" >
        <inputText label="#{workitemmodel.labels.TASKID}"
                   value="#{workitemmodel.values.TASKID}"
                   readOnly="true"/>
        <inputText label="#{workitemmodel.labels.DESC}"
                   value="#{workitemmodel.values.DESC}"/>
        <listOfValues label="#{workitemmodel.labels.OWNER}"
                   value="#{workitemmodel.values.OWNER}"
                   srcAttr="OWNER"
                   lovDef="#{workitemmodel.lovDefs.OWNER}"/>
  <comboxBox label="#{workitemmodel.labels.STATUS}"
                   value="#{workitemmodel.values.STATUS}"
                   valueSet="#{workitemmodel.listItems.STATUS}"/>
  <inputDate label="#{workitemmodel.labels.DUEDATE}"
                   value="#{workitemmodel.values.DUEDATE}"/>
        <inputText label="#{workitemmodel.labels.URL}"
                   value="#{workitemmodel.values.URL}"/>
    </formLayout>
  </region>
</regions>
```

Table 4–4 Control types

UI Tag	Attributes	Description
inputText	label	Field label to show in the UI
	value	Runtime field value
	required	Determines if it is required.

Table 4–4 (Cont.) Control types

UI Tag	Attributes	Description
	readOnly	Determines if the data can be edited
inputDate	label	Field label to show in the UI
	value	Runtime field value
	required	Determines if it is required.
	readOnly	Determines if the data can be edited
comboBox	label	Field label to show in the UI
	value	Runtime field value
	required	Determines if it is required.
	readOnly	Determines if the data can be edited
checkBox	label	Field label to show in the UI
	value	Runtime field value
	required	Determines if it is required.
	readOnly	Determines if the data can be edited
radio	label	Field label to show in the UI
	value	Runtime field value
	required	Determines if it is required.
	readOnly	Determines if the data can be edited
	valueSet	Defines where to get the list of values.
listOfValues	label	Field label to show in the UI
	value	Runtime field value
	required	Determines if it is required.
	readOnly	Determines if the data can be edited
	srcAttr	Defines the source field to use
	lovDef	Defines the lovDef of the source field
textEditor	label	Field label to show in the UI
	value	Runtime field value
	required	Determines if it is required.
	readOnly	Determines if the data can be edited
	rows	Number of rows to show
	required	Determines if it is required.
list	label	Field label to show in the UI
	value	Runtime field value
	valueSet	Defines where to get the list of values.

4.2.6.2 Using Panels for More Complicated UI

You can also lay out controls using panels for more complicated UI layouts. Panels have the property of being horizontal or vertical. Panels can also be nested.

Example 4–13 Control layout using panels

```
<?xml version="1.0" encoding="windows-1252" ?>
<regions resFile="/META-INF/res/uiresources.xml">
  <region name="Default" helpTopicId="f1_connector_sample_htm">
<panelLayout layout="horizontal">
    <formLayout columns="1" blockSize="10" >
        <inputText label="#{workitemmodel.labels.TASKID}"
                   value="#{workitemmodel.values.TASKID}"
                   readOnly="true"/>
        <inputText label="#{workitemmodel.labels.DESC}"
                   value="#{workitemmodel.values.DESC}"/>
        <listOfValues label="#{workitemmodel.labels.OWNER}"
                   value="#{workitemmodel.values.OWNER}"
                   srcAttr="OWNER"
                   lovDef="#{workitemmodel.lovDefs.OWNER}"/>
  <comboxBox label="#{workitemmodel.labels.STATUS}"
                   value="#{workitemmodel.values.STATUS}"
                   valueSet="#{workitemmodel.listItems.STATUS}"/>
  <inputDate label="#{workitemmodel.labels.DUEDATE}"
                   value="#{workitemmodel.values.DUEDATE}"/>
        <inputText label="#{workitemmodel.labels.URL}"
                   value="#{workitemmodel.values.URL}"/>
    </formLayout>
 <textEditor value="#{workitemmodel.values.BIGDESC}"/>
  </region>
</regions>
```

Table 4–5 UI tags used in panel example

UI Tag	Attribute	Description
panelLayout	layout	Specifies the layout direction of the grouping controls. Valid values are vertical and horizontal.

4.2.7 How to Retrieve Data from an Oracle Team Productivity Center Repository

After the model and UI layouts for different work item types are defined, the next step is to implement the methods on the `WorkItemConnector` interface, and the methods on `WorkItemAttachment` interface (if the connector supports attachments). This topic provides details on these methods.

Before you can implement these methods, you need to add the location of the Team Productivity Center JAR files to your project so you can pick up the appropriate classes for the `WorkItemConnector` and `WorkItemAttachment` interfaces.

To add the location of Team Productivity Center JAR files to your project:

1. Double-click on your extension project, and then select **Project Properties > Libraries and Classpath**.
2. On the right side, select **Add JAR/Directory**.
3. Navigate to where you installed JDeveloper. Under the `JDev/Extensions/oracle.teamproductivitycenter/lib` folder, locate the file `almcommon-ip.jar` and include that.

4.2.7.1 Implementing the WorkItemConnector Interface

Now that the JAR is added, you can implement the interfaces for the `WorkitemConnector`.

To implement the interface:

1. Select **File > New**.
2. Select **General > Java > Java Class**.
3. In the Create Java Class wizard, enter your class and package name.
4. In the Optional Attributes section, click the plus to implement an interface.
5. In the Class and Package Browser, select the Hierarchy tab and then find the following class: `oracle/alm/connector/WorkItemConnector`. If you intend to implement the `WorkItemAttachment` interface, select that also using the **Ctrl** key.
6. Click **OK**.

This will provide you with a skeleton file that has all the methods that you must implement.

Table 4–6 Method Summary

API		Description
void	init(Map sessionContext)	Initializes the connector instance on the client side, and builds the connector's physical connection parameters for its corresponding repository
void	login(Map sessionContext, String userID, String password)	Establishes a user connection between the connector instance on the client side and the backend repository
void	logout(Map sessionContext)	Disconnect the user from the backend repository. State can be saved using the data hash structure.
Void	setWorkItemDefs(Map session, Map<String, WorkItemDef> wiDefs)	Framework constructs the runtime data structures for the work item types defined in the connector model definition
List<WorkItem>	getQueryResult(Map sessionContext, String wiType, QueryInfo query)	Retrieves the query result set that satisfies the query criteria defined by QueryInfo
WorkItem	getWorkItem(Map sessionContext, String wiType, WorkItem workItem)	Retrieves a specific work item by its type and unique identifier
void	updateWorkItem(Map sessionContext, String wiType, WorkItem workItem)	Updates an existing work item by its type and unique identifier
void	createWorkItem(Map sessionContext, String wiType, WorkItem workItem)	Creates a new work item on its backend repository for the specified type
	deleteWorkItem(Map sessionContext, String wiType, WorkItem workItem)	Deletes a work item from its backend repository
List<Row>	getLOVQueryResult(Map sessionContext, QueryInfo query, String fieldName)	Retrieves the value set for a predefined LOV (List of Value)

Table 4–6 (Cont.) Method Summary

API		Description
boolean	IsAttachmentSupported (Map sessionContext, WorkItem wi)	Determines if attachment is supported for the specified work item. If yes, the work item model needs to implement the `WorkItemAttachment` interface.
String	getUIRegionName(Map sessionContext, String wiType, WorkItem wi)	Retrieves the region name defined in the connector UI XML
boolean	hasDynamicUI(Map session)	If the connector supports dynamic UI, returns true
String	getDynamicUI(Map session, String currentUI)	Retrieves the new work item UI definition XML string
boolean	hasDynamicModel(Map session)	If the connector supports dynamic model, returns true
String	getDynamicModel (Map session, String currentModel)	Retrieves the new work item model definition XML string

4.2.8 How to Refresh Work Item Detail Based on Custom Actions

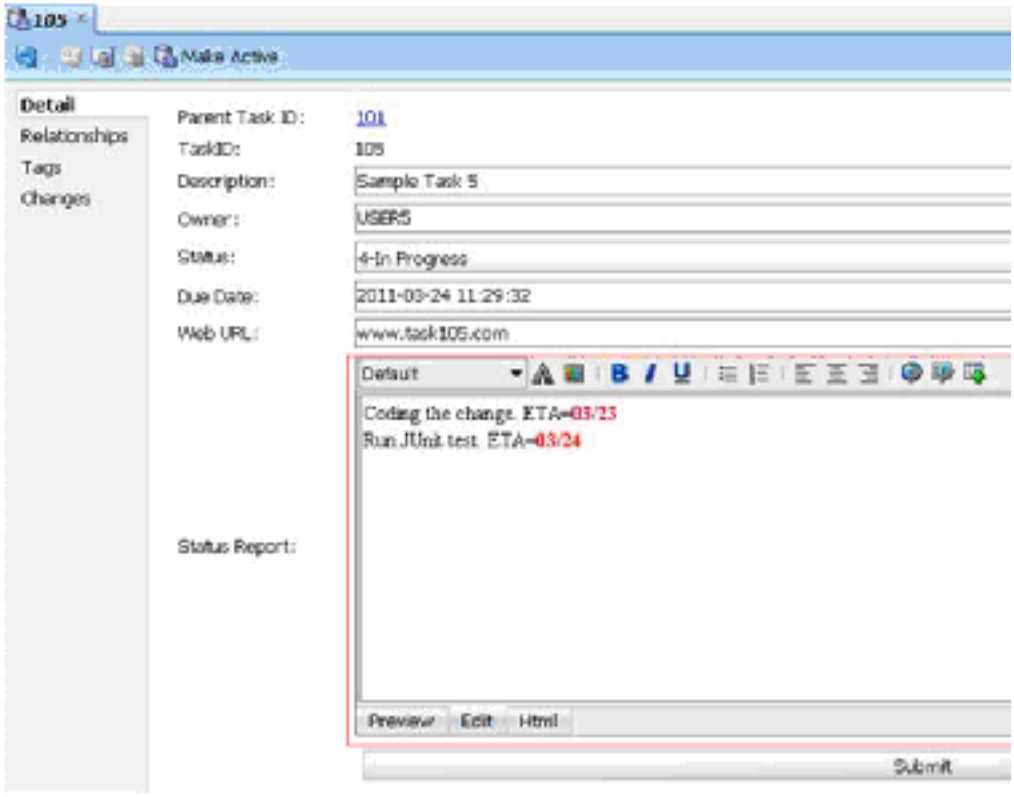
In certain scenarios, the work item detail page needs to be refreshed to a new layout based on some custom action in the existing layout. For example, users may make a new value selection from a combo box, or a date change from a date time picker, or just simply clicking a button in the page.

When a task is shown in its detail editor, there is a "Status" combination box in its base UI:

Figure 4–1 Status combination box in UI

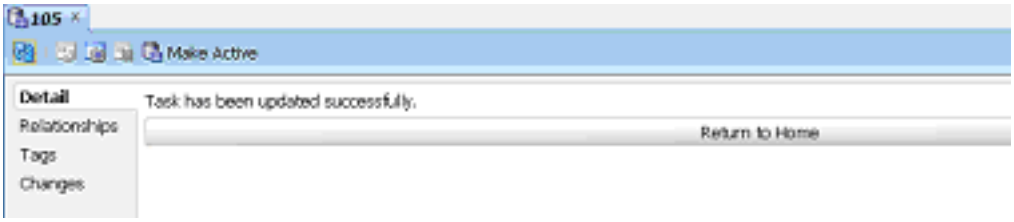
When the value of the status is changed from New to In-Progress, a new layout is shown with a rich text control that lets the user provide information on the status progress.

Figure 4–2 New rich text control for status UI



When the user finishes editing the work item and clicks **Submit**, the confirmation page shows with success or failure message.

Figure 4–3 Confirmation page after using rich text control



To show the updated task detail page, click **Return to Home**.

To implement the functionality:

1. Decide which UI control to use as the action source to trigger dynamic UI change, and which listener to control the UI interaction.
2. Define new UI regions in the work item UI layout meta data file.
3. Implement the corresponding listener in managed bean.
4. Implement API `getUIRegionName()` to return different UI region based on different condition.
5. Register the managed bean in the connector's `tpc-config.xml` file.

6. Use the bean in the UI metadata through EL expression.

The following sections provide more detail using the sample connector.

4.2.8.1 Decide UI Control and Listener Attribute

The sample connector uses the combination box control for the "status" field as the action source. It also uses the `PopupMenuListener` on the combo box control to listen to the selection change and to do dynamic update UI work. It also uses the Submit button to navigate between the task detail page and the confirmation page layout.

The configuration detail is in the `SampleDef.xml` file. Note in the following example how the status combination box and submit action are defined with their corresponding attribute name and value pairs.

Example 4–14 UI control and listener attribute selection

```
<?xml version="1.0" encoding="windows-1252" ?>
    <region id="Homepage">
        <formLayout columns="1" blockSize="10" fieldWidth="400" labelWidth="60">
            <inputText label="#{workitemmodel.labels.PARENT_TASKID}"
                               value="#{workitemmodel.values.PARENT_TASKID}"
readOnly="true"/>
            <inputText label="#{workitemmodel.labels.TASKID}"
                               value="#{workitemmodel.values.TASKID}"
readOnly="true"/>
            <inputText label="#{workitemmodel.labels.DESC}"
                                value="#{workitemmodel.values.DESC}"/>
            <listOfValues label="#{workitemmodel.labels.OWNER}"
                                    value="#{workitemmodel.values.OWNER}"
source="OWNER"
                                    pprTargets="statusID"/>
            <comboBox label="#{workitemmodel.labels.STATUS}"
                   value="#{workitemmodel.values.STATUS}" id="statusID"
          valueSet="#{workitemmodel.listItems.STATUS}" readOnly="true"
                   popupMenuListener="#{mainBean.statusListener}"/>
            <inputDate label="#{workitemmodel.labels.DUEDATE}"
                                value="#{workitemmodel.values.DUEDATE}"/>
            <inputText label="#{workitemmodel.labels.URL}"
                               value="#{workitemmodel.values.URL}"/>
           <panelLayout>
   <action text="Submit" actionCommand="gotoc
actionListener="${dynamicBean.actionListener}"/>onfirmpage"
          </panelLayout>
        </formLayout>
        <separator/>
    </region>
```

4.2.8.2 Define New Regions in UI Metadata File

In addition to the default `HomePage` region, this code creates two new regions: region "Demo" (shown when status is changed to "in-progress") and region "confirmation" (shown when the user clicks **Submit**).

Example 4–15 New region definition

```
<region id="Demo" helpTopicId="f1_connector_sample_htm">
     <formLayout columns="1" blockSize="10" fieldWidth="400" labelWidth="60">
            <inputText label="#{workitemmodel.labels.PARENT_TASKID}"
```

```
                              value="#{workitemmodel.values.PARENT_TASKID}"
readOnly="true"/>
            <inputText label="#{workitemmodel.labels.TASKID}"
                              value="#{workitemmodel.values.TASKID}"
readOnly="true"/>
            <inputText label="#{workitemmodel.labels.DESC}"
value="#{workitemmodel.values.DESC}"/>
            <listOfValues label="#{workitemmodel.labels.OWNER}"
                                 pprTargets="statusID"
value="#{workitemmodel.values.OWNER}" source="OWNER"/>
            <comboBox label="#{workitemmodel.labels.STATUS}"
                                  value="#{workitemmodel.values.STATUS}"
                                  popupMenuListener="#{mainBean.statusListener}"
                                  valueSet="#{workitemmodel.listItems.STATUS}"
readOnly="true"
                                  id="statusID"/>
            <inputDate label="#{workitemmodel.labels.DUEDATE}"
                                 value="#{workitemmodel.values.DUEDATE}"/>
            <inputText label="#{workitemmodel.labels.URL}"
value="#{workitemmodel.values.URL}"/>
            <textEditor label="#{workitemmodel.labels.STATUS_REPORT}"
                                  value="#{workitemmodel.values.STATUS_REPORT}"
contentType="html"/>
            <panelLayout>
                  <action text="Submit" actionCommand="gotoconfirmpage"
                              actionListener="${dynamicBean.actionListener}"/>
            </panelLayout>
        </formLayout>
</region>

<region id="confirmation">
     <panelLayout layout="vertical">
        <inputText value="Task has been updated successfully." readOnly="true"/>
        <action text="Return to Home" actionCommand="gotohomepage"
                                 actionListener="${dynamicBean.actionListener}"/>
      </panelLayout>
    </region>
```

4.2.8.3 How to Implement a Corresponding Listener in a Managed Bean

The listener object on the status combo box (`popupMenuListener`) is implemented in `HomePageBean.java` as follows:

Example 4–16 Implementing a listener in a managed bean

```
public PopupMenuListener getStatusListener()
{
  if (statusListener == null)
  {
    statusListener = new PopupMenuListener()
      {
        public void popupMenuWillBecomeInvisible(PopupMenuEvent e)
        {
          JComboBox src = (JComboBox) e.getSource();
          Object[] selected = (Object[])src.getSelectedItem();
          if(selected == null)
            return;

          String value = (String)selected[1];
```

```
            if (value != null && value.contains("Progress"))
            {
               _rcontext.getConnectorParams().put("navigatorpage", "D");
            }
            else
              _rcontext.getConnectorParams().put("navigatorpage", "H");
            //we just need to refresh the current page;
            ViewManager vm = ViewManager.getInstance();
            if(vm != null)
              vm.refreshWorkItemDetailUI(_rcontext);
          }
          public void popupMenuCanceled(PopupMenuEvent e)
          {
          }
          public void popupMenuWillBecomeVisible(PopupMenuEvent e)
          {
          }
        };
    }
    return statusListener;
  }
```

The listener object on the Submit button (`actionListener`) is implemented in the `DynamicRegionBean` as follows:

Example 4–17 Implementing listener on Submit button

```
  public ActionListener getActionListener()
  {
    if (actionListener == null)
    {
      actionListener = new ActionListener()
        {
          public void actionPerformed(ActionEvent event)
          {
            String command = event.getActionCommand();
            ViewManager vm = ViewManager.getInstance();
            if (command != null && command.equals("gotohomepage"))
            {
              _rcontext.getConnectorParams().put("navigatorpage", "H");
            }
            else if (command != null && command.equals("gotowelcomepage"))
            {
              _rcontext.getConnectorParams().put("navigatorpage", "W");
            }
            else if (command != null && command.equals("gotodemopage"))
            {
              _rcontext.getConnectorParams().put("navigatorpage", "D");
            }

            vm.refreshWorkItemDetailUI(_rcontext);
          }
        };
    }
```

4.2.8.4 Guideline for Implementing the Managed Bean

All managed beans should implement the `AlmScope` interface. Oracle Team Productivity Center uses `AlmScope` to synchronize the `renderingContext` and the corresponding `AlmComponent` for the UI control that uses the managed bean.

There are two ways to pass information from connector Managed Bean layer so it can be used in other connector classes: through `RenderingContext`, and through the current work item object.

- Through `RenderingContext`

  Parameter name value pairs can be put at the connector session level. To review the code shown in the previous example:

Example 4–18 Parameter name value pairs at connector session level

```
_rcontext.getConnectorParams().put("navigatorpage", "H");
```

  This information can be retrieved inside other connector classes as shown here:

Example 4–19 Retrieving information inside other connector classes

```
Map params = (Map) session.get(WorkItemConnector.SESSION_PARAMS_KEY);
```

- Through the current work item object

  The current work item object can be accessed through the `ViewManager` class:

Example 4–20 Accessing work item through ViewManager class

```
public   WorkItem getCurrentWorkItem(RenderingContext rcontext);
```

  Inside the managed bean class, you can modify the work item's submit values. Then other connector classes can use it to process logic based on different field values.

Use `ViewManager` APIs to refresh the work item detail page.

Example 4–21 Refreshing the work item detail page

```
public void refreshWorkItemDetailUI(RenderingContext rcontext);
```

This API triggers the call to `getUIRegionName()`, which returns new region name based on connector parameter or work item field values. For example, the sample connector uses a parameter to return a new region name:

Example 4–22 Using a parameter to return a new region name

```
  public String getUIRegionName(Map session, String wiType, WorkItem workItem)
    throws ALMException
  {
    String region = "";

    Map params = (Map)
session.get(WorkItemConnector.SESSION_PARAMS_KEY);
    if (params != null)
    {
      Object val = params.get("navigatorpage");
      if (val == null || val.equals("H"))
        region = "Homepage";
      else if (val != null && val.equals("W"))
```

```
        region = "Confirmation";
      else if (val != null && val.equals("D"))
        region = "Demo";
    }

    return region;
  }
```

4.2.8.5 Register the Bean Class in tpc-config.xml

Register the managed bean `DynamicRegionBean` with the name `UIBean` inside the file `tpc-config.xml` as follows:

Example 4–23 Registering the bean class in tpc-config.xml

```
<?xml version="1.0" encoding="windows-1252" ?>
<!-- TPC Configuration file -->
<tpc-config version="11.1.1.1.0" xmlns="http://fusion.oracle.com/tpc">
  <managed-bean>
    <name>mainBean</name>
    <impl-class>oracle.sampleconnector.view.mbean.HomePageBean</impl-class>
    <lifecycle>page</lifecycle>
  </managed-bean>
  <managed-bean>
    <name>dynamicBean</name>
    <impl-class>oracle.sampleconnector.view.mbean.DynamicRegionBean</impl-class>
    <lifecycle>page</lifecycle>
  </managed-bean>
  <managed-bean>
    <name>compBean</name>
    <impl-class>oracle.sampleconnector.view.mbean.ComponentBean</impl-class>
    <lifecycle>page</lifecycle>
  </managed-bean>
</tpc-config>
```

4.2.9 How to Open a New Editor Inside JDeveloper

On a work item detail page, the connector writer can create a link that, when clicked, opens a new Team Productivity Center-specific editor. For example, a Bugzilla bug has a field "Depends On," which may hold a bug number for another bug. The Bugzilla connector writer can create a link on the label. When clicked, a new editor opens up inside JDeveloper showing the details of that bug.

Team Productivity Center uses `alm` as the resource protocol and supports these URL formats:

- Work item

  `alm:/reposName/workitemType/workitemId.wid`

- Team query

  `alm:/reposName/workitemType/Team Queries/queryName.wiq`

- User query

  `alm:/reposName/workitemType/My Queries/queryName.wiq`

- Generic page

  `alm:/path/filename.tpcx?repository=reposName&…..`

Note that for a generic page, the connector writer needs to do the following:

1. Put UI page files directly under the `/META-INF/pages` folder in the connector source code tree. This is because `/META-INF/pages` is set as the default `DOC_HOME` in the TPC framework.
2. Attach `repository=reposName` as a parameter to the URL.

For example, if the URL used in the sample connector is `alm:/component/component.tpcx?repository=SampleConnector&component=inputDate`, then there should be a `component.xml` file under the `/META-INF/pages/component` folder.

To open a new editor:

Using these URLs, there are two different ways to open a Team Productivity Center-specific editor:

1. Use `OpenTPCPageInEditor ()` API on `ViewManager`:

Example 4–24 Using the OpenTPCPageInEditor() API

```
ViewManager vm = ViewManager.getInstance();
vm.OpenTPCPageInEditor("/component/component.tpcx?
repository=SampleConnector&component=inputDate");
```

Similarly, this will open a specific work item editor:

Example 4–25 Opening a specific work item editor

```
ViewManager.getInstance().OpenTPCPageInEditor
 ("reposName/workitemType/workitemId.wid");
```

2. Use the `WebResource` tag

   Using the sample connector as an example, there is a field “Parent ID” where its value portion is a link. When clicked, the work item editor for the parent task opens up. To achieve this, follow these steps.

   - Use the `WebResource` tag to define the field that can generate the URL in connector definition file:

Example 4–26

```
<RepositoryModel resName="res" resFile="/META-INF/res/modelresource.xml">
  <WorkItem type="Task"
                   webURLHandler="oracle.sampleconnector.SampleConnectorService">
    <Fields>
      <Field name="PARENT_TASKID" label="${res.PARENT_TASK_ID}" type="number"
                readOnly="true"/>
      <Field name="TASKID" label="${res.TASK_ID}" type="number" readOnly="true"/>
      <Field name="DESC" label="${res.TASK_DESC}" required="true"
          maxLength="80" type="string"/>
 ……
    <Fields/>
    <WebResource>
<URLDef name="PARENT_TASKID" anchorOn="label"/>
……
    </WebResource>
……
  <WorkItem/>
<RepositoryModel/>
```

The valid values for the attribute `anchorOn` are "label" and "value". If the attribute value is "label" the field label becomes a hyperlink. Otherwise the value acts as a hyperlink.

- Implement the `getURL()` API on `WorkItemFieldFeature` interface

  Class `SampleConnectorService` implements the method on this interface. The `getURL()` method constructs the URL for the parent task field.

Example 4–27

```
public URL getURL(Map session, String attrName, Object attrValue, WorkItem wi)
{
  String urlString = "";
  if (attrName.equalsIgnoreCase("PARENT_TASKID"))
  {
    Integer taskid = (Integer) attrValue;
    String reposName = (String)session.get("repositoryName");
    String wiType = (String)session.get("wiType");

    if(reposName != null && wiType != null)
      urlString = "alm:/" + reposName + "/" + wiType + "/" + taskid + ".wid";
  }

  try
  {
    return new URL(urlString);
  }
  catch (MalformedURLException e)
  {
    System.out.println(e);
  }
  return null;
}
```

- In the connector definition file, find the work item section and put the class name as the value for attribute `webURLHandler`. See the sample code in the first bullet.

  When the value for `anchorOn` is set to "label", the UI metadata for the corresponding work item field should be set to an "input" field, such as `inputText`, `inputDate`, or `ListOfValues`.

  When the value for `anchorOn` is set to "value", the UI metadata for corresponding work item field should be set to read-only `inputText`.

4.2.10 How to Debug an Oracle Team Productivity Center Connector

Once you have developed your connector, you will want to be able to run your connector inside of the debugger during execution. This capability is built into your project already.

To run your connector inside the JDeveloper debugger:

- Right-click on your built/deployed project, and then select **Debug Extension**.

This launches another copy of JDeveloper from which you can open up Team Productivity Center and connect to an Oracle Team Productivity Center Server that has your connector defined. Then you can set breakpoints in your Java code and step through it while using watch windows for variables and use standard debugging utilities.

4.3 Handling Oracle Team Productivity Center Connector Errors

Once you have developed a connector for use with Oracle Team Productivity Center, there are a number of tasks left to perform. One important task is to handle errors that occur while using the connector. Oracle Team Productivity Center has two ways of propagating errors and messages to the client; they are described here.

You may choose to include help files with your connector, to provide assistance for your team members in using the connector you have developed. You can then view your help in the JDeveloper online help browser or in an external help viewer.

In addition, you will need to package your Oracle Team Productivity Center connector so that it can be downloaded by other team members. This includes compressing the connector into a JAR file and a bundle file. Furthermore, any help files you have created for the connector need to be included in the JAR file.

Oracle Team Productivity Center provides two ways to propagate errors and messages to the Team Productivity Center client:

- through `ALMException` used on interface methods
- through `ALMMessage` used to queue to `ALMMessageFactory`

All interface methods throw `ALMException` when something serious happens inside a connector. You can put an appropriate warning or message with the runtime instance of `ALMException`. The client framework will show to it the end user.

Example 4–28 Error handling

```
try
{
   connector.hasDynamicUI(session);
   }
catch(ALMException e)
{
   setLastError(e);
}
```

You can also use the classes provided in package oracle.alm.common.message to propagate error messages to the client. Two classes are defined in this package.

Table 4–7 Class Summary

AlmMessage	
AlmMessageFactory	Manages the connection to the backend repository.

`AlmMessageFactory` is the class for message registration and access through Team Productivity Center. Use this class and `AlmMessage` to log any messages to be reported to the client through the UI framework.

`AlmMessageFactory` is implemented as a singleton. To get an `AlmMessageFactory`, call:

```
AlmMessageFactory msgFactory = AlmMessageFactory.getInstance()
```

To add a message, call `AlmMessageFactory.addMessage()` by passing the `messageID` and the associated message string.

4.4 Adding Help to an Oracle Team Productivity Center Connector

You can add custom help files, including context-sensitive help, to an Oracle Team Productivity Center connector. Your users can then access any information they need to help with the operation of setup of your connector.

There are two ways to display connector specific help pages to the users of your connector. The first approach is to show these help pages inside JDeveloper Help Center. The second approach is to show these pages in their own frame on the client's desktop or in a Web browser.

Extensive help is available for the writer of help for a JDeveloper extension, such as the Team Productivity Center connector, in the Developing Help Extensions topic.

4.4.1 How to Add Help to the Team Productivity Center Connector

You can write and display connector-specific help inside the JDeveloper Help Center for your users to refer to. If you implement JDeveloper help as an JDeveloper Help extension, your users will be able to access this help by pressing the F1 key while viewing a dialog in your connector.

To include help for a Team Productivity Center connector:

1. Provide F1 help files in HTML format for each page that shows in JDeveloper. They should be located in a folder `/Help` inside the packaged connector JAR file.
2. Create a new JDeveloper extension that only contains these help files.
3. Add the connector help ID to the connector UI definition XML file.

Example 4–29 Connector help ID

```
<region name=”Edit” helpTopicId="f1_connector_htm" >
  <formLayout value="${workitemmodel}"
              columns="2"
              blockSize="6"
              id="wiFormId"
              rowToSpan="1"/>
</region>
```

4.4.2 How to View Team Productivity Center Connector Help in an External Viewer

You may choose to present the help for your connector in an external viewer, such as the user's default Web browser or a PDF viewer.

To present connector help in an external viewer:

1. Provide F1 help files in HTML format for each page that shows. They should be located in a folder called `/Help` in the packaged connector JAR file.
2. Create a new JDeveloper extension that only contains these help files.
3. Add a Help button in the work item detail UI. This can be done by adding a toolbar and one help button on this toolbar in the connector UI definition XML file:

Example 4–30 Adding a Help button to a toolbar

```
<region name=”Edit”>
  <toolbar title="Actions">
    <action text="Help"
       actionCommand="showHelp"
```

```
      icon="HELP"
      actionListener="${showHelp}"/>
  <toolbar/>
  <formLayout value="${workitemmodel}"
              columns="2"
              blockSize="6"
             id="wiFormId"
              rowToSpan="1"/>
</region>
```

4. Implement the listener object class.

4.5 Packaging Connectors

The final phase of preparing an Oracle Team Productivity Center connector is to package it in JAR files for installation as a JDeveloper extension. This allows you to install the connector on the Team Productivity Center server so that your team members can connect to the repository to which this connector applies.

4.5.1 How to Package Connectors

In order to install your connector onto the server, and so it can be installed as an extension into JDeveloper, it must be packaged correctly. There are three pieces to the packaging:

1. The JAR file that will contain your connector and all of its required files.
2. A separate JAR file containing your help files and their required files.
3. A bundle in ZIP format that will contain the two previous JAR files plus a `bundle.xml` file to describe how JDeveloper can install this as an extension.

You can create deployment profiles inside of your project to generate all of these files for you.

Before you create any deployment profiles, ensure you have built the project successfully and all output files have been created in the correct directories.

4.5.2 How to Generate the Connector JAR File

The connector JAR file contains all the components of your connector that will be included in the extension dedicated to your connector.

To generate the connector JAR file:

1. Double-click on your project in the Application Navigator and select **Project Properties > Deployment**.
2. Click **New**.
3. Select JAR File as your archive type, and then enter a name for your JAR file. Typically this is `Connector<your connector name>.JAR`
4. In the Edit JAR Deployment Profile Properties dialog, select **Filters**. Select all files in the META-INF directory (except `Bundle.xml`, to be defined soon) and all the files in your package directory.
5. Click **OK**.

To test your connector JAR deployment, right-click on your project and select **Deploy > <your deploy profile> > to JAR File.**

4.5.3 How to Generate the Connector Help JAR File

The Help JAR file will contain everything needed to include Help with your connector extension. JDeveloper extension tools provide an integrated process for generating connector Help JAR files.

To generate a connector Help JAR file:

1. Double-click on your project in the Application Navigator and select **Project Properties > Deployment**.
2. Click **New**.
3. Select JAR File as your archive type, and then enter a name for your help JAR file. Typically this is `Connector<your connector name>Help.JAR`
4. In the Edit JAR Deployment Profile Properties dialog, select Filters. Select all the help-related files or the directory those are stored in. Deselect all other files.
5. Click **OK**.

To test your connector JAR deployment, right-click on your project and select **Deploy > <your help deploy profile> > to JAR File.**

4.5.4 How to Generate the Connector Bundle File

When the connector JAR file is constructed, you can then prepare to deploy it by generating a bundle file. This is a ZIP file, which you can use to install the connector extension.

To generate a connector bundle file:

1. Double-click on your project and select **Project Properties > Deployment**.
2. Click **New**.
3. Select **JAR File** as your archive type.
4. Uncheck **Include Manifest**.
5. Enter a name for your bundle ZIP file.

   This is typically located in a Bundle folder at the same level as your Deploy folder, and called `Connector<your connector name>Bundle.ZIP`. Be sure to name it .ZIP instead of .JAR.
6. In the Edit JAR Deployment Profile Properties dialog, expand the Project Output node, then select **Filters**.
7. Rename this to file group to JAR Output. Change the Target Directory in Archive to: `oracle.teamproductivitycenter/connectors/<your connector name>`. There should be no .JAR at the end. This is the path inside of the ZIP file to extract this file to

   Note: The installers are case-sensitive, so make sure the case is correct for your file paths.
8. Go to the Contributors node under the JAR Output file group. Click **Add** and add the location where your connector JAR and help JAR files are generated.
9. Go to the Filters node under the JAR Output file group. Select the connector and connector help JAR files you already generated

10. Select the File Groups node and click **New**. Create a new file group called Bundle Output.
11. Go to the Filters node under the Bundle Output file group and deselect everything except for your bundle.xml file. Make sure the Target Directory in Archive edit box is blank.
12. Go to the Profile Dependencies node and add the connector and the help deploy profiles as dependencies.
13. Click **OK**.

To test your connector bundle ZIP deployment, right-click on your project and select **Deploy > <your bundle profile> > to JAR File**. This creates a new ZIP file that contains your connector JAR, the help JAR and the `bundle.xml` file.

4.5.5 How to Define Dependencies in the Connector Bundle File

The bundle.xml file describes what is in an extension and what its dependencies are. It also holds the version information and info about the author.

Example 4–31 Connector bundle file bundle.xml

```
<update-bundle version="1.0"
    xmlns="http://xmlns.oracle.com/jdeveloper/updatebundle"
    xmlns:u="http://xmlns.oracle.com/jdeveloper/update">

  <u:update id="oracle.teamproductivitycenter.sampleconnector">
    <u:name>Oracle Sample Connector</u:name>
    <u:version>1.0</u:version>
    <u:author>Oracle Corporation</u:author>
    <u:author-url>http://www.oracle.com</u:author-url>

    <u:description>
      Sample connector for Oracle Team Productivity Center
    </u:description>

    <u:requirements>
      <u:requires-extension id="oracle.jdeveloper"
        minVersion="11.1.1.1.00"
        maxVersion="11.1.1.1.99" />
      <u:requires-extension id="oracle.teamproductivitycenter"
        minVersion="11.1.1.1.00"
        maxVersion="11.1.1.1.99" />
    </u:requirements>
  </u:update>
</update-bundle>
```

All Team Productivity Center connectors have dependencies on both JDeveloper and Team Productivity Center. These should always be included.

4.6 Team Productivity Center Connector Internationalization

Oracle Team Productivity Center provides support for connectors to show appropriate UI based on the running JDeveloper locale. As seen in the work item UI and model definition XML files, a resource file can be used to store localized strings. For example, if JDeveloper is running in JPN locale, the user may want to see connector detail page in the same locale. To support this, you can create language-specific resource bundle files. Example 4–32 is an example for the `modelresource.xml`:

Example 4–32 Internationalization file modelresource.xml

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<resources xmlns="http://xmlns.oracle.com/bali/rts/tpc"
package="company.bugzilla.res">
  <resource key="MSG_SAVEBUG_TITLE">BugDB: Save A Bug</resource>
  <resource key="MSG_NEWBUG_TITLE">BugDB: Create A Bug</resource>
  <resource key="MSG_BUGAPI_ERROR">Bug API Error: </resource>
  <resource key="DESCRIPTION">Description</resource>
  <resource key="BUG_NAME">Bug </resource>
  <resource key="BUG_NO">Bug Number</resource>
  <resource key="BUG_BASEBUG_NO">Base Bug No</resource>
  <resource key="BUG_STATUS">Status</resource>
  <resource key="BUG_SUBJECT">Subject</resource>
  <resource key="BUG_PRODUCT">Product</resource>
  <resource key="BUG_PRIORITY">Severity</resource>
  <resource key="BUG_COMPONENT">Component</resource>
  <resource key="BUG_SUBCOMPONENT">Sub Component</resource>
  <resource key="BUG_ASSIGNEE">Assigned</resource>
</resources>
```

And here is an example of the file `uiresource.xml`:

Example 4–33 Internationalization file uiresource.xml

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<resources xmlns="http://xmlns.oracle.com/bali/rts/tpc"
           package="company.bugzilla.res">
  <resource key="COMMMENTS">Comments</resource>
  <resource key="ADD_COMMENTS">Add Comments</resource>
  <resource key="UPDATE_COMMENTS">Update Comments</resource>
</resources>
```

If JDeveloper is running in Japanese, the framework looks for resource bundle files named `modelresource_jp.xml` and `uiresource_jp.xml` to find and display labels in Japanese. You can create these files and put them in a central location. Store these resource bundle files under the folder `META_INF\res` in the connector jar file.