Oracle® Fusion Middleware

User's Guide for Oracle Team Productivity Center

11*g* Release 2 (11.1.2.1.0)

E17914-02

September 2011

ORACLE

Oracle Fusion Middleware User's Guide for Oracle Team Productivity Center 11*g* Release 2 (11.1.2.1.0)

E17914-02

Primary Author: Scott Fisher

Contents

Preface ix

Audience ix
Documentation Accessibility ix
Related Documents ix
Conventions x

What's New in This Guide in Release 11.1.2.1.0 xi

1 Introduction to Oracle Team Productivity Center

1.1 About Oracle Team Productivity Center 1-1
1.1.1 How to Migrate from a Previous Version 1-2
1.2 Connecting to Oracle Team Productivity Center 1-2
1.2.1 How to Log Into Team Productivity Center 1-2
1.2.2 How to Disconnect From Oracle Team Productivity Center 1-3
1.2.3 How to Manage Your User Profile 1-3
1.2.4 How to Manage Your Repository Accounts 1-4
1.3 Navigating Oracle Team Productivity Center 1-4
1.3.1 How to Select Your Team Productivity Center Team 1-5
1.3.2 How to Use the Team Productivity Center Chat 1-5
1.3.2.1 Logging into the Chat Server 1-6
1.4 Understanding the Build Dashboard and Build Summary 1-8
1.4.1 How to Open and Use the Build Dashboard 1-8
1.4.2 How to Use the Build Dashboard and Build Summary 1-9

2 Working with Oracle Team Productivity Center

2.1 About Working with Oracle Team Productivity Center 2-1
2.1.1 How to Support an Information Repository for Oracle Team Productivity Center 2-2
2.2 Working With Work Items 2-2
2.2.1 How to Create and Use Work Items 2-3
2.2.2 How to Use Work Items with the Change List 2-3
2.2.3 How to Set and Display the Active Work Item 2-4
2.2.4 How to Connect to a Repository from Team Productivity Center 2-4
2.2.5 How to Access Work Items through a Team Query 2-5
2.2.6 How to Associate Work Items with Project Files 2-5
2.2.7 How to Save and Restore Context 2-6

2.3 Working with the Task Repository ... 2-7
2.3.1 How to Find Tasks in the Task Repository ... 2-8
2.3.2 How to Create a Task Work Item ... 2-8
2.3.3 How to Edit a Task Work Item ... 2-8
2.4 Working with Attachments ... 2-8
2.4.1 How to Add Attachments ... 2-9
2.4.2 How to Download Attachments ... 2-10
2.4.3 How to Update Attachments ... 2-10
2.4.4 How to Rename Attachments ... 2-10
2.5 Working with Queries ... 2-11
2.5.1 How to Create Queries ... 2-11
2.5.2 How to Run a Query ... 2-12
2.5.3 How to Customize a Query ... 2-12
2.5.4 How to Rename a Query ... 2-13
2.5.5 How to Save a Modified Query ... 2-13
2.5.6 How to Create a Team Query ... 2-14
2.5.7 How to Delete a Query ... 2-14
2.6 Working with Tags ... 2-15
2.6.1 How to Create, Modify, and Delete Tags ... 2-15
2.6.2 How to Use Tags ... 2-16
2.7 Working with Relationships ... 2-17
2.7.1 How to Add a Relationship to a Work Item ... 2-18
2.7.2 How to Delete a Relationship ... 2-18
2.7.3 How to Associate a Relationship with a Work Item ... 2-18

3 Working with Oracle Team Productivity Center Administration

3.1 About Working with Oracle Team Productivity Center Administration ... 3-1
3.1.1 Understanding Repositories, Users, and Teams ... 3-1
3.1.2 Understanding User Permissions ... 3-2
3.1.3 Understanding User Roles ... 3-2
3.2 Adding Repositories to Oracle Team Productivity Center ... 3-3
3.2.1 How to Add a Repository ... 3-3
3.2.2 How to Remove a Repository ... 3-4
3.3 Making Repositories Available in Oracle Team Productivity Center ... 3-4
3.3.1 How to Access and Select Repositories in Team Productivity Center ... 3-4
3.3.2 How to Find Repositories in the Team Administration Dialog ... 3-5
3.3.3 How to Assign Repositories ... 3-5
3.3.4 How to Create and Use Versioning Repositories ... 3-6
3.4 Working with Users, Teams and Roles ... 3-7
3.4.1 How to Create Team Productivity Center User Accounts ... 3-7
3.4.2 How to Add Users to Team Productivity Center ... 3-7
3.4.3 How to Modify User Information ... 3-9
3.4.4 How to Remove Users from Team Productivity Center ... 3-9
3.4.5 How to Find Users or Teams in the Team Administration Dialog ... 3-10
3.4.6 How to Add and Remove Teams from Oracle Team Productivity Center ... 3-10
3.4.7 How to Set and Change Team Member Roles ... 3-11

4 Working with Oracle Team Productivity Center Connectors

4.1 About Working with Oracle Team Productivity Center Connectors 4-1
4.1.1 Fetching Data From the Repository 4-2
4.1.2 Generating the Component UI Tree 4-2
4.1.3 The Work Item Editor 4-2
4.1.4 The Dynamic Work Item Model and UI 4-2
4.1.5 How to Download Extension Connectors 4-2
4.1.6 Team Productivity Center Connector Execution Flow Example 4-3
4.1.7 How to Define Team Productivity Center Repository Data 4-3
4.1.8 How to Add Listeners to the Connector 4-7
4.2 Creating Oracle Team Productivity Center Connectors 4-7
4.2.1 How to Create an Oracle Team Productivity Center Connector 4-7
4.2.2 How to Create an Oracle Team Productivity Center Connector Configuration File. 4-8
4.2.3 How to Use Customized Listeners and Managed Beans 4-10
4.2.4 How to Access Connector Data Through a Firewall 4-14
4.2.5 How to Present Data Declaratively with the Team Productivity Center UI 4-15
4.2.6 How to Retrieve Data from an Oracle Team Productivity Center Repository 4-18
4.2.6.1 Implementing the WorkItemConnector Interface 4-19
4.2.7 How to Refresh Work Item Detail Based on Custom Actions 4-20
4.2.7.1 Decide UI control and listener attribute 4-22
4.2.7.2 Define New Regions in UI Metadata File 4-22
4.2.7.3 How to Implement a Corresponding Listener in a Managed Bean 4-23
4.2.7.4 Guideline for implementing the managed bean 4-25
4.2.7.5 Register the Bean class in tpc-config.xml 4-26
4.2.8 How to Open a New Editor Inside JDeveloper 4-26
4.2.9 How to Debug an Oracle Team Productivity Center Connector 4-28
4.3 Handling Oracle Team Productivity Center Connector Errors 4-29
4.4 Adding Help to an Oracle Team Productivity Center Connector 4-30
4.4.1 How to Add Help to the Team Productivity Center Connector 4-30
4.4.2 How to View Team Productivity Center Connector Help in an External Viewer .. 4-30
4.5 Packaging Connectors 4-31
4.5.1 How to Package Connectors 4-31
4.5.2 How to Generate the Connector JAR File 4-31
4.5.3 How to Generate the Connector Help JAR File 4-32
4.5.4 How to Generate the Connector Bundle File 4-32
4.5.5 How to Define Dependencies in the Connector Bundle File 4-33
4.6 Team Productivity Center Connector Internationalization 4-33

Preface

Welcome to the *User's Guide for Oracle Team Productivity Center*.

Audience

This document is intended for developers who 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

- *Oracle Fusion Middleware Installation Guide for Oracle Team Productivity Center Server*
- *Oracle Fusion Middleware Java API Reference for Oracle Team Productivity Center*
- *Oracle Fusion Middleware Tag Reference for Oracle Team Productivity Center Connectors*
- *Oracle Fusion Middleware Installation Guide for Oracle JDeveloper*
- *Oracle Fusion Middleware User Guide for Oracle JDeveloper*
- *Oracle Fusion Middleware Developer's Guide for Oracle JDeveloper Extensions*
- Oracle JDeveloper *11g Online Help*
- Oracle JDeveloper *11g Release Notes*, link included with your Oracle JDeveloper 11*g* 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 in Release 11.1.2.1.0

For Release 11.1.2.1.0, the *Oracle Fusion Middleware User Guide for Oracle Team Productivity Center* replaces the non-context sensitive online help that was available in previous releases of Oracle Team Productivity Center. It incorporates the information on working with Oracle Team Productivity Center, administering Oracle Team Productivity Center, and working with Oracle Team Productivity Center connectors.

For additional information on Oracle Team Productivity Center in this release, see the Oracle Team Productivity Center page on the Oracle Technology Network at http://www.oracle.com/technetwork/developer-tools/tpc/overview/index.html.

For changes made to Oracle JDeveloper and Oracle Application Development Framework (Oracle ADF) for this release, see the What's New page on the Oracle Technology Network at http://www.oracle.com/technetwork/developer-tools/jdev/documentation/index.html.

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:

- Section 1.1, "About Oracle Team Productivity Center"
- Section 1.2, "Connecting to Oracle Team Productivity Center"
- Section 1.3, "Navigating Oracle Team Productivity Center"
- Section 1.4, "Understanding the Build Dashboard and Build Summary"

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 and specifications, 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 three parts:

- The Oracle Team Productivity Center client software.
- 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.

To connect to the Oracle Team Productivity Center server:

- **View > Team Navigator**

This opens the Team Navigator. Click on **Connect to Team Server** to continue.

For more information on installing connectors from Check for Updates, see the "*How to Install Extensions with Check for Update*s" section in the Oracle Fusion Middleware User Guide for Oracle JDeveloper.

1.2.1 How to Log Into Team Productivity Center

When you initially log into Team Productivity Center, you will need some information provided by your administrator.

Table 1–1 Productivity Center Login 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 Connection Information	With this option selected, you will only need to enter your password when logging in the future. Team Productivity Center will retain all other information about your connection (server, port, and username).
Connect Automatically	With this option selected, you will be connected to Oracle Team Productivity Center every time you open JDeveloper.

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.

1.2.3 How to Manage Your User Profile

Your Oracle Team Productivity Center user profile contains your first and last name, your email, and your Oracle Team Productivity Center password. On the Manage User Profile dialog, you can also set Oracle Team Productivity Center to remember your password on future connections.

To set values in your user profile:

1. From the Team Navigator menu, select **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**.
5. Click **OK**.

1.2.4 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**.
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 and viewing or querying work items. You can also select your team and chat with team members.

You navigate Team Productivity Center through the Team Navigator.

To use the Team Navigator:

- Select **View > Team > Team Navigator**.

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

At the very top of the Team Navigator accordion, Oracle Team Productivity Center displays a drop-down list of teams. Once you select a team from this list, the three accordion panels below display data for the selected team and its members.

Team Members
Displays the members when you select a given team.

Work Items
Displays work item repositories connected for the selected team. When 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

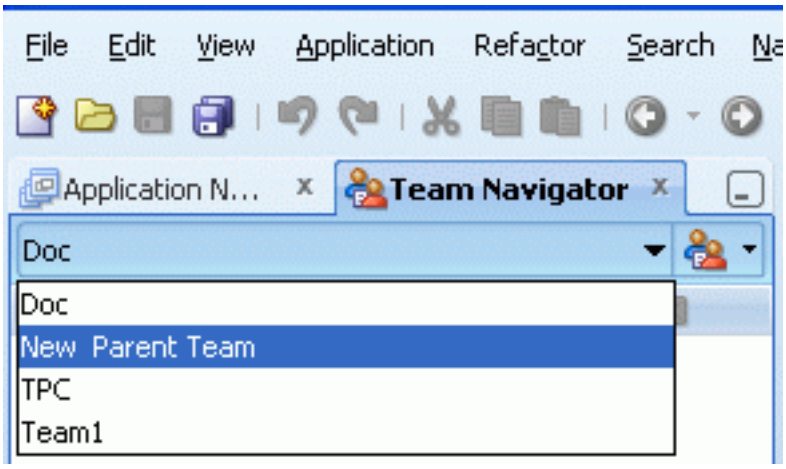
When you connect to your Team Productivity Center server, Oracle Team Productivity Center displays a list of the teams to which your administrator has added you.

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 Versioning and Work Items accordions. Each of these will provide access to information repositories needed by your team.

Oracle Team Productivity Center displays the online and chat status of team members. 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.

1.3.2 How to Use the Team Productivity Center Chat

Once you have installed Team Productivity Center, you can chat with members of your team.

To connect to Oracle Team Productivity Center chat:

- Select **View > Team > Chat**.

You can initiate a chat session in two ways:

- Click the chat symbol adjacent to the Team Members accordion header.
- Double-click on the name of any team member.

When you initiate a chat in Oracle Team Productivity Center you will have two groups available for chat:

- Buddies from your active XMPP/Jabber protocol.
- Teammates from the currently selected team, visible in the members accordion only.

Although you can only connect to one chat server at a time, you can also create a new connection to a different chat server and select it as your active server.

Once you have installed Team Productivity Center, you can use Chat by selecting **View > 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 create a new chat server connection:

1. Disconnect from your current server. You can either use the Disconnect button, or select the connection from the list, then click the right mouse button and select **Disconnect**.
2. Right-click on the empty Buddies node, then select **Connect**.

   You can also 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.
3. Enter the settings for your new chat server.

1.3.2.1 Logging into the Chat Server

Oracle Team Productivity Center only supports XMPP/Jabber protocols, and does not support multiple 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 1–2 contains the information that is typically included.

Table 1–2 Configuration Options

Option	Description
Host	The name of the host chat server URL.
Port	The assigned port number.
Domain	The assigned domain.
Resource	JDeveloper
User Name	Your chat server user name.
Password	Your chat server password.
Remember Password	(dependent on chat server configuration)

Table 1–2 (Cont.) Configuration Options

Option	Description
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.
Edit Connection Properties	Displays Connect to Chat Server dialog, which allows you to specify the following properties for your chat connection and connect to the server.
Host	The URL for your selected chat server. Disconnect from any previously connected server before choosing this one, either for Connect or for Edit Connection Properties
Port	The port on the chat server dedicated to chat.
Domain	The top-level domain (for example, mycompany.com) used to host the chat server.
Resource	JDeveloper
Username	Your username in the chat server you are selecting or specifying.
Password	The password associated with the username in the preceding field.
Remember Password	Save this password for future chat sessions. Selected automatically if you also select Connect When JDeveloper Starts.
Connect When JDeveloper Starts	Automatically connect to this chat server each time you start JDeveloper. Use this to specify the default chat server.

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

Table 1–3 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.
Show	Displays a drop-down menu with the following options: Show Offline Buddies Show Empty Groups Show Details Add Group Add Buddy

Table 1–3 (Cont.) Chat window control buttons

Button	Description
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.

1.4 Understanding the Build Dashboard and Build Summary

The Build Dashboard and Build Summary summarize the builds of projects your organization is tracking with Team Productivity Center. When you first log in to Team Productivity Center, JDeveloper displays a temporary snapshot of the Build Summary, containing information about the development/build branches your team is following, with 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 work items 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 View menu.

To open the Build Dashboard:

- **View > Team > 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.

To use the Build Dashboard:

Clicking the branch name or the Total Transactions links of a row in the Build Dashboard displays the build summary window. 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 1–4.

Table 1–4 Build Dashboard Options

Option	Description
Branch	Click the dropdown to select the development branch you are interested in.

Table 1–4 (Cont.) Build Dashboard Options

Option	Description
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 associated with the change set being displayed. Select the work item to view and modify it.

1.4.2 How to Use the Build Dashboard and Build Summary

All build/test system configuration takes place when the plug-in is installed in the build/test environment, typically when Oracle Team Productivity Center 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 **Branch** drop-down from the Build Dashboard, then select **Customize**.

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

From the build dashboard, you select one of the available builds about which you wish to find out more information. This displays the Build Summary for the selected build

To select a build from the Build Dashboard:

- Click on the **Branch** drop-down from the Build Dashboard, then select the build of interest.

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. Table 1–5 contains Build Summary options, including detailed information about a given build with work item and owner information.

Table 1–5 Build Summary Options

Option	Description
Show	Select the statistics that you want the Build Summary to display:
All Tests	Show the results of every test, including pass and fail results.
Errors Only	Show only the results of tests that contain errors.
Failures Only	Show only the results of tests that fail.
Errors and Failures	Show the results both of tests with errors and also tests with failures.
Successes Only	Show only 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.
Unassigned Only	If you check this box, the Build Summary will display only selected results (as determined by the statistics selected in the Show box) that have no assigned owner. Leave the box unchecked to display all results.
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, 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:
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.

Table 1–5 (Cont.) Build Summary Options

Option	Description
% 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.

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:

- Section 2.1, "About Working with Oracle Team Productivity Center"
- Section 2.2, "Working With Work Items"
- Section 2.3, "Working with the Task Repository"
- Section 2.4, "Working with Attachments"
- Section 2.5, "Working with Queries"
- Section 2.6, "Working with Tags"
- Section 2.7, "Working with Relationships"

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®.
- 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 permission 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.

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 Create and 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 2.5.2, "How to Run a Query."

After selecting one or more work items 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 2.2.2, "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 2.2.3, "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 2.2.7, "How to Save and Restore Context."
5. Add a work item to a tag list. For more information, see Section 2.6, "Working with Tags."
6. Create a relationship to another work item. Section 2.7, "Working with Relationships."

2.2.2 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. Alternatively, you can choose one or more other work items to associate the files with. 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

2.2.3 How to Set and Display the Active Work Item

The active work item in JDeveloper is a work item that you have selected by clicking the Make Active command in the tool bar for the work item. You can also right-click on a work item in a list of work items returned by a query. You can select any work item as active at any time; only one work item can be active at a time, so making a new work item active replaces any previously selected active work item.

The Work Items accordion includes a link to the work item you have selected, above the list of repositories. This link remains after closing JDeveloper. If you shut down 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.

2.2.4 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. 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.

To connect to a repository:

1. Choose **Team Navigator menu > Manage Accounts**.
2. Select one of the repositories under the Accounts column.
3. From the Repository Server drop-down, select the preferred server for the repository you selected in the preceding step.
4. 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**. This will allow you to verify the connection data and make changes if required.

2.2.5 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 2.5, "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.

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, you can refine the results of this query to restrict (or expand) the number of work items returned. Additionally, 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.

2.2.6 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 the following optional capabilities 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. For example: **Team > Subversion > Commit**.

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 Worktems panel of the commit or checkin dialog displays four columns of information about each work item you add:

Table 2–1 Commit dialog options for Team Productivity Center work items

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 U**p icon (double upward arrows).

   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**.

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 dialog immediately. If you delete a work item in error, add it back.

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

2.2.7 How to Save and Restore 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.

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.

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**.

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.

2.3 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. The Task repository is installed when you install the Team Productivity Center server. 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
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.

You specify or change the values that appear in each column when you create or edit a Task work item. See Section 2.3.2, "How to Create a Task Work Item."

2.3.1 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 2.5, "Working with Queries."

2.3.2 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, select **File > Save**.

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."

2.3.3 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, select **File > Save**.

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."

2.4 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, 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**.

2.4.1 How to Add Attachments

You add attachments to a selected 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**.

2.4.2 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.

2.4.3 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`).

2.4.4 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:

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**.

2.5 Working with Queries

Queries are the way Team Productivity Center retrieves work items 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.

2.5.1 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.

2.5.2 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.

2.5.3 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.

2.5.4 How to Rename a Query

You may wish the change the name of a query to reflect other modifications or customization you have made to it.

To rename a query:

1. Click on **More Actions > Save As**.
2. Enter the new name for the query in the Name field.
3. If desired, change the visibility of the query:

   To make the query visible to the entire team, select **Team**.

   To make the query visible only to you, select **User**.
4. Click **OK**.

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

2.5.5 How to Save a Modified Query

Some Team Productivity Center queries have a life span of a single use. For example, you may wish to query your repository for a single bug record that had been filed in error, so that you can correct it. Once the bug entry is fixed, you may never need to run that query again.

Other queries, however, may be useful on a regular basis, either to you or to your team members. For example, you may have a query that searches the bug database for all bugs that have your name and are not closed, sorted in reverse chronological order, for a specific combination of product areas. This is a query you might conceivably use every day. In such cases, saving the query lets you choose it again in the future.

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.

2.5.6 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**.

2.5.7 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.

2.6 Working with Tags

In addition to connecting to 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 the repositories 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. Or considered functionally: you use a query to return a list of work items that match a particular search term; once you have that list, you can tag the individual work items that you are specifically interested in.

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.

2.6.1 How to Create, Modify, and Delete Tags

You can 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.

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
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 only.)
- User: only you can view and query by this tag.

6. Click **OK**.

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**.

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.

2.6.2 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.

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 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 2.6.1, "How to Create, Modify, and Delete Tags."

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.

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 2–3 Options while searching for tagged work items

Option	Description
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.
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.
New Query	Opens the New Query pane in JDeveloper. See Section 2.5.1, "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.

2.7 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.

2.7.1 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.
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 2.7.3, "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**.

2.7.2 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.

2.7.3 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:

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 2.2.3, "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.

3

Working with Oracle Team Productivity Center Administration

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

This chapter includes the following sections:

- Section 3.1, "About Working with Oracle Team Productivity Center Administration"
- Section 3.2, "Adding Repositories to Oracle Team Productivity Center"
- Section 3.3, "Making Repositories Available in Oracle Team Productivity Center"
- Section 3.4, "Working with Users, Teams and Roles"

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 four areas: repositories, users, teams, and roles.

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. Your team members can access and interact with the information in these repositories depending on two sets of circumstances: the nature of the repository and its information (for example, a bug tracking database or a task repository); and the privileges you have assigned to each individual user, or to the team to which that user belongs.

When you add a new user, you assign them permissions allowing them to modify different content elements in Team Productivity Center, and you assign them to one or more teams. 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. You can also modify individuals' roles, either on a person-by-person basis or by creating new roles with specific permissions that serve the needs of your organization.

3.1.1 Understanding Repositories, Users, and Teams

Managing the repositories used 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 the *Oracle Fusion Middleware Installation Guide for Oracle Team Productivity Center Server*.

Administrators typically perform the following tasks:

- Connect repositories to 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 and users. 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 permission to modify the elements of Team Productivity Center, such as work items, queries, tags, and document. All users have defined roles, either by default or by the administrator, and each user's role defines what capabilities and privileges that user has. 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 those 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 those 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 accept 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.

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 *Oracle Fusion Middleware Installation Guide for 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.

To add a repository to Team Productivity Center:

1. Select **Team Navigator menu > Team Administration > Repositories** tab.
2. In the Repository Servers 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 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 Navigator menu > 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, 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 *Oracle Fusion Middleware Installation Guide for 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.

To select repositories for team use:

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

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.

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.

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 available repositories, which in turn is based on the installed connectors. 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 this 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 URL or a port number).

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**.
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**.

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

To give team access to a versioning repository:

1. Select **Team Administration > Repositories > 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 individually, 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 roles 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, which may be performed in any order as team members and repositories are added, either at the beginning of the product lifecycle or during development:

- 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.
- Changing team member roles. Team members are assigned roles within the team. These roles play an important part in what permissions the team members have when it comes to modifying queries and other team-wide interactions.
- 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.

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 through the Team Productivity Center Administration Dialog.

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.
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.

You can add and remove team members to and from existing teams through the Team Productivity Center Administration dialog.

To add a team member to an existing team:

1. Open the Team Server Administration 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 double right-angle 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 administrative rights this user is to have on the team.
8. Click **OK**.

To remove a team member from an existing team:

1. Open the Team Server Administration 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."

3.4.3 How to Modify User Information

If you have administrative privileges you can modify information related to any user in the system.

To modify user information:

1. Open Team Server Administration 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**.

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 all users 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 user names regardless of capitalization. The quick filter also works to search through the entire list of teams.

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. An Oracle Team Productivity Center administrator can adjust team hierarchies. Team Administrators will be restricted to teams on which they have been assigned the team administrative role that includes any team they have created. The Group Administrator can 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.

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. Click **OK**.

To show inactive teams:

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

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) 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 can also create and manage queries, tags,

documents and sources for their teams, plus 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.
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 user settings. For more information, see Section 3.4.3, "How to Modify User Information."

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:

- Section 4.1, "About Working with Oracle Team Productivity Center Connectors"
- Section 4.2, "Creating Oracle Team Productivity Center Connectors"
- Section 4.3, "Handling Oracle Team Productivity Center Connector Errors"
- Section 4.4, "Adding Help to an Oracle Team Productivity Center Connector"
- Section 4.5, "Packaging Connectors"
- Section 4.6, "Team Productivity Center Connector Internationalization"

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.
- Runtime UI component tree generation inside the Team Productivity Center client.
- Display of UI controls and data in work item editor that ties the repository data to UI components.

4.1.1 Fetching Data From the Repository

Each work item connector implements the `WorkitemConnector` interface, and optionally `WorkitemAttachment` interface if the repository this connector uses supports attachments. The `WorkitemConnector` interface specifies methods for end user logging 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, 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.

4.1.2 Generating the Component UI Tree

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 their 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 the 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.

4.1.3 The Work Item Editor

When both work item data and its UI components tree are available, the `WorkitemEditor`, a special editor inside JDeveloper, binds the data to the UI controls and displays them inside the editor.

4.1.4 The Dynamic Work Item Model and UI

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.

4.1.5 How to Download Extension Connectors

The Oracle Team Productivity Center is provided by default with the Task Repository connector. All other connectors require that you download them from the JDeveloper Update Center, using the Check for Updates wizard.

To download a connector extension:

1. Select **Help** > **Check for Updates**.
2. In the Check for Updates wizard, select **Search Update Centers**.
3. Make sure that the Open Source and Partner Extensions center is selected, and click **Next**.
4. On the Updates page, scroll through the list and select the Oracle Team Productivity Center connectors that you wish to download, and then click **Next**.

5. Click **Finish**.

Online help for each of these connectors will be available after you download and install them. You can also refer to the contents and structure of these connectors as samples while developing your own connector.

For more information, see the "How to Install Extensions with Check for Updates" section in the *Oracle Fusion Middleware User Guide for Oracle JDeveloper*.

4.1.6 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 JIRA work item object definition.
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()` uses the JIRA SOAP interface to retrieve the issue data.
7. When the SOAP interface returns the data, `getWorkitem()` converts it to a JIRA Workitem 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.

4.1.7 How to Define Team Productivity Center Repository Data

In this example, 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.

You can use tags in `workitem-object.xsd` to describe work item data in this definition file. This 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 their data.

workitem-object.xsd is located in the file almcommon-api.jar. This file, along with the Oracle Team Productivity CenterConnector Tag Guide tpctagdoc.zip, is distributed under the directory jdeveloper\jdev\extensions\oracle.teamproductivitycenter\doc.

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" archorOn="label"/>
                        <URLDef name="TASKID" archorOn="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.

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

Configuration Tag	Attributes	Description
	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 4.2.5, "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.
	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.

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

Configuration Tag	Attributes	Description
	archorOn	Whether the underlined hyperlink should be on the label or the actual value. Valid values are "`label`" or "`field`". Default = `field`.

4.1.8 How to Add Listeners to the Connector

The `workitem.properties` file is what Team Productivity Center uses to store the listeners (java class to invoke when any event happens regarding work item). Edit this file to include specific listeners that your connector relies on for user input.

When building your connector extension, make sure the `workitem.properties` file is included in your connector bundle.

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 an 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 4.2.2, "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 4.1.7, "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.

   For more information, see Section 4.1.8, "How to Add Listeners to the Connector."

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 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

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

Configuration Tag	Attributes	Description
	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
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

4.2.3 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 *Oracle Fusion Middleware 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 inExample 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}"
```

To write 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>
```

To customize context menu action:

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;
}
```

4.2.4 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));
```

4.2.5 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
	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.

To show a UI page using an explicit layout:

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.
	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.

To lay out more complicated UI using panels:

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.

4.2.6 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.

4.2.6.1 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 sesion, 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

Table 4–6 (Cont.) Method Summary

API		Description
	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)
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

4.2.7 How to Refresh Work Item Detail Based on Custom Actions

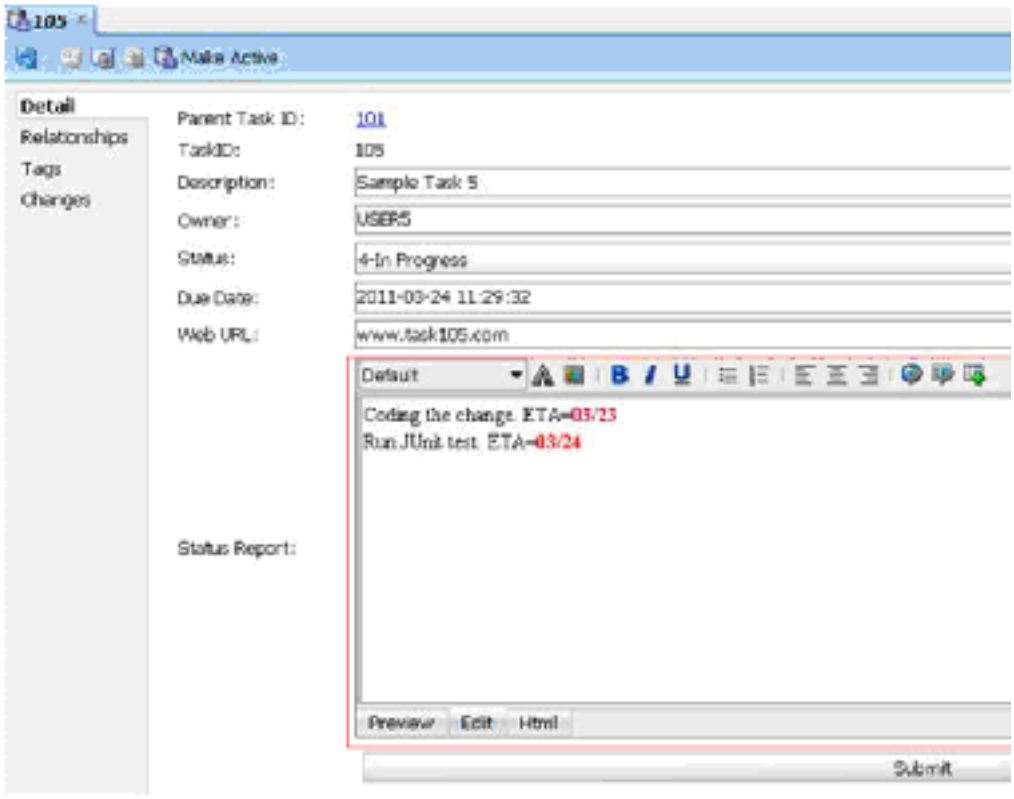
In certain scenarios, the work item detail page needs to be refreshed to a new layout based on some custom action in the existing layout. For example, users may make a new value selection from a combo box, or a date change from a date time picker, or just simply clicking a button in the page.

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

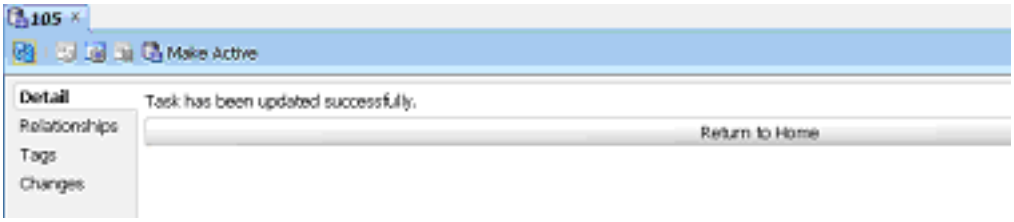
When the value of the status is changed from New to In-Progress, a new layout is shown with a rich text control that lets the user provide information on the status progress.

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.

4.2.7.1 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>
```

4.2.7.2 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>
```

4.2.7.3 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);
        }
      };
  }
```

4.2.7.4 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`

  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;
  }
```

4.2.7.5 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>
```

4.2.8 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" archorOn="label"/>
……
    </WebResource>
……
  <WorkItem/>
<RepositoryModel/>
```

The valid values for the attribute `archorOn` 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`.

4.2.9 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 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.