

Oracle® Fusion Middleware

Developing Applications with Oracle ADF Desktop Integration

12c (12.1.3)

E41269-01

May 2014

Documentation for Oracle ADF Desktop Integration developers that describes how to extend the functionality provided by a Fusion web application to desktop applications.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Primary Author: Himanshu Marathe

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface	xv
Audience.....	xv
Documentation Accessibility	xv
Related Documents	xv
Conventions	xv
What's New in This Guide	xvii
New and Changed Features for 12c (12.1.3).....	xvii
Other Significant Changes in this Document for 12c (12.1.3)	xvii
1 Introduction to ADF Desktop Integration	
1.1 About ADF Desktop Integration	1-1
1.2 About ADF Desktop Integration with Microsoft Excel	1-2
1.2.1 Overview of Creating an Integrated Excel Workbook	1-2
1.2.2 Advantages of Integrating Excel with a Fusion Web Application	1-3
2 Introduction to the ADF Desktop Integration Sample Application	
2.1 About the Summit Sample Application for ADF Desktop Integration	2-1
2.2 Setting Up and Running the Summit Sample Application for ADF Desktop Integration	2-1
2.3 Overview of the Fusion Web Application in the Summit Sample Application for ADF Desktop Integration	2-3
2.3.1 About the Fusion Web Application in the Summit Sample Application for ADF Desktop Integration	2-3
2.3.2 Downloading Integrated Excel Workbooks	2-3
2.4 Overview of the Integrated Excel Workbooks in the Summit Sample Application for ADF Desktop Integration	2-4
2.4.1 Log on to the Fusion Web Application from an Integrated Excel Workbook	2-4
2.4.2 Downloading Data Rows	2-5
2.4.3 Modify Customers and Warehouses Information in the Workbooks	2-5
2.4.4 Upload Modified Information to the Fusion Web Application	2-6
3 Setting Up Your Development Environment	
3.1 About Setting Up Your Development Environment	3-1
3.2 Required Oracle ADF Modules and Third-Party Software	3-2
3.3 Configuring Excel to work with ADF Desktop Integration	3-3

3.4	Installing ADF Desktop Integration	3-4
3.4.1	How to Set Up ADF Desktop Integration	3-4
3.5	Removing ADF Desktop Integration	3-6
3.6	Upgrading ADF Desktop Integration	3-6
3.6.1	How to Migrate an Integrated Excel Workbook to the Current Version of ADF Desktop Integration	3-6
3.7	Using an Integrated Excel Workbook with Older Versions of ADF Desktop Integration	3-7
3.8	Using ADF Desktop Integration on a System with Multiple Instances of JDeveloper	3-8

4 Preparing Your Integrated Excel Workbook

4.1	About Preparing Your Integrated Excel Workbooks	4-1
4.2	Adding an Integrated Excel Workbook to a Fusion Web Application	4-1
4.2.1	How to Add an Integrated Excel Workbook to a Fusion Web Application	4-2
4.2.2	How to Configure a New Integrated Excel Workbook	4-3
4.2.3	How to Add Additional Worksheets to an Integrated Excel Workbook	4-7
4.2.4	What Happens When You Deploy an ADF Desktop Integration-Enabled Fusion Web Application from JDeveloper	4-8
4.2.4.1	Fusion Web Application is Deployed on Oracle WebLogic Server	4-8
4.3	Working with Page Definition Files for an Integrated Excel Workbook	4-8
4.3.1	How to Create ADF Desktop Integration Page Definition File	4-9
4.3.2	What Happens When You Create a Page Definition File	4-11
4.3.3	How to Reload a Page Definition File in an Excel Workbook	4-11
4.3.4	What You May Need to Know About Page Definition Files in an Integrated Excel Workbook	4-12
4.4	Enabling ADF Desktop Integration Manually	4-13
4.4.1	How to Manually Add ADF Desktop Integration In Fusion Web Application	4-13
4.4.2	How to Enable ADF Desktop Integration in an Existing Workbook	4-14
4.4.3	How to Manually Configure a New Integrated Excel Workbook	4-15
4.4.4	What Happens When You Add ADF Desktop Integration to Your JDeveloper Project . 4-17	
4.4.5	Adding ADF Library Web Application Support	4-18

5 Getting Started with the Development Tools

5.1	About Development Tools	5-1
5.1.1	ADF Desktop Integration Development Tools Use Cases and Examples	5-2
5.1.2	Additional Functionality for ADF Desktop Integration Development Tools	5-3
5.2	Designer Ribbon Tab	5-3
5.3	ADF Desktop Integration Designer Task Pane	5-6
5.4	Using the Bindings Palette	5-7
5.5	Using the Components Palette	5-9
5.6	Using the Property Inspector	5-9
5.7	Using the Binding ID Picker	5-11
5.8	Using the Expression Builder	5-12
5.9	Using the Web Page Picker	5-13
5.10	Using the File System Folder Picker	5-14
5.11	Using the Page Definition Picker	5-14
5.12	Using the Collection Editors	5-15

5.13	Using the Cell Context Menu	5-16
5.14	Removing ADF Desktop Integration Components	5-17
5.15	Exporting and Importing Excel Workbook Integration Metadata	5-19
5.15.1	How to Export Workbook Integration Metadata	5-19
5.15.2	How to Import Workbook Integration Metadata	5-20
5.15.3	What You May Need to Know About Exporting and Importing Excel Workbook Integration Metadata	5-21

6 Working with ADF Desktop Integration Form-Type Components

6.1	About ADF Desktop Integration Form-Type Components	6-1
6.1.1	ADF Desktop Integration Form-Type Components Use Cases and Examples	6-2
6.1.2	Additional Functionality for ADF Desktop Integration Form-Type Components	6-3
6.2	Inserting an ADF Button Component	6-3
6.3	Inserting an ADF Label Component	6-5
6.4	Inserting an ADF Input Text Component	6-7
6.5	Inserting an ADF Output Text Component	6-9
6.6	Inserting an ADF List of Values Component	6-10
6.7	Inserting an ADF Image Component	6-12
6.8	Inserting Error Components	6-14
6.8.1	What You May Need to Know About Error Components	6-15
6.9	Displaying Tooltips in ADF Desktop Integration Form-Type Components	6-17
6.9.1	What You May Need to Know About Tooltips	6-19
6.10	Displaying Output from a Managed Bean in an ADF Component	6-19
6.10.1	How to Display Output from a Managed Bean	6-20
6.10.2	What Happens at Runtime: How an ADF Component Displays Output from a Managed Bean	6-21
6.11	Displaying Concatenated or Calculated Data in Components	6-21
6.11.1	How to Configure a Component to Display Calculated Data	6-21
6.12	Using Navigation Buttons	6-22

7 Working with ADF Desktop Integration Table-Type Components

7.1	About ADF Desktop Integration Table-Type Components	7-2
7.1.1	ADF Desktop Integration Table-Type Components Use Cases and Examples	7-3
7.1.2	Additional Functionality of Table-Type Components	7-3
7.2	Page Definition Requirements for an ADF Table Component	7-3
7.3	Inserting ADF Table Component into Excel Worksheet	7-4
7.3.1	How to Insert ADF Table Component	7-5
7.3.2	How to Add a Column in an ADF Table Component	7-7
7.4	Configuring Oracle ADF Component to Download Data to an ADF Table Component	7-8
7.4.1	How to Configure an ADF Component to Download Data to an ADF Table Component	7-8
7.4.2	What Happens at Runtime: How the ADF Table Component Downloads Data	7-10
7.5	Configuring a Worksheet to Download Data as Pending Insert Rows in an ADF Table component	7-10
7.5.1	What Happens at Runtime: Download Action is Invoked	7-11
7.5.2	Using STATUS_INITIALIZED Rows for Pending Inserts	7-11
7.5.3	What You May Need to Know About DownloadForInsert Action	7-12

7.6	Configuring an ADF Table Component to Update Existing Data	7-12
7.6.1	How to Configure an ADF Table Component to Update Data	7-12
7.6.2	What Happens at Runtime: How the ADF Table Component Updates Data	7-13
7.7	Configuring an ADF Table Component to Insert Data	7-13
7.7.1	How to Configure an ADF Table Component to Insert Data Using a View Object's Operations	7-13
7.8	Configuring an ADF Component to Upload Changes from an ADF Table Component	7-15
7.8.1	How to Configure an ADF Component to Upload Data from an ADF Table Component	7-16
7.8.2	What Happens at Runtime: How the ADF Table Component Uploads Data	7-17
7.8.3	What Happens at Runtime: How the ReadOnly EL Expression Is Evaluated During Upload	7-19
7.8.4	What Happens at Runtime: Upload Failure	7-19
7.8.5	How to Create a Custom Upload Dialog	7-19
7.8.6	What Happens at Runtime: Custom Upload Dialog	7-20
7.9	Configuring an ADF Table Component to Upload Changes Using UploadAllOrNothing Action	7-21
7.9.1	How to Configure an ADF Component to use UploadAllOrNothing Action	7-21
7.9.2	What Happens at Runtime: UploadAllOrNothing Action is Invoked	7-21
7.9.3	Limiting the Amount of Changed Data That Can Be Uploaded With UploadAllOrNothing Action	7-22
7.10	Configuring an ADF Table Component to Delete Rows in the Fusion Web Application	7-22
7.10.1	How to Configure an ADF Table Component to Delete Rows in the Fusion Web Application	7-22
7.10.2	What Happens at Runtime: How the ADF Table Component Deletes Rows in a Fusion Web Application	7-24
7.11	Batch Processing in an ADF Table Component	7-25
7.11.1	How to Configure Batch Options for an ADF Table Component	7-25
7.11.2	Row Flagging in an ADF Table Component	7-26
7.11.3	Troubleshooting Errors While Uploading Data	7-27
7.12	Special Columns in the ADF Table Component	7-27
7.13	Configuring ADF Table Component Key Column	7-28
7.13.1	How to Configure the Key Column	7-29
7.13.2	How to Manually Add the Key Column At Design Time	7-29
7.14	Creating a List of Values in an ADF Table Component Column	7-30
7.14.1	How to Create a List of Values in an ADF Table Component Column	7-31
7.14.2	What Happens at Runtime: How the ADF Table Column Renders a List of Values	7-32
7.15	Adding a ModelDrivenColumnComponent Subcomponent to Your ADF Table Component	7-33
7.16	Adding a Dynamic Column to Your ADF Table Component	7-34
7.16.1	How to Configure a Dynamic Column	7-34
7.16.2	What Happens at Runtime: How Data Is Downloaded or Uploaded In a Dynamic Column	7-35
7.16.3	How to Specify Header Labels for Dynamic Columns	7-35
7.16.4	How to Specify Styles for Dynamic Columns According to Attribute Data Type ...	7-36
7.17	Configuring Column Widths in an ADF Table Component	7-37
7.17.1	How to Configure an ADF Table Component to Resize Columns at Runtime	7-37

7.17.2	How to Configure an Action Set to Resize Columns of an ADF Table Component at Runtime	7-40
7.17.3	What Happens at Runtime: How the ADF Table Columns are Resized	7-41
7.17.4	What You May Need to Know About Resizing Columns of an ADF Table Component at Runtime	7-41
7.18	Displaying Tooltips in ADF Desktop Integration Table-Type Components	7-42
7.18.1	What You May Need to Know About Tooltips for Table Columns	7-43
7.19	Creating an ADF Read-Only Table Component	7-44
7.19.1	How to Insert an ADF Read-only Table Component	7-45
7.19.2	How to Manually Add a Column to the ADF Read-only Table Component	7-46
7.20	Limiting the Number of Rows Your Table-Type Component Downloads	7-47
7.20.1	How to Limit the Number of Rows a Component Downloads	7-47
7.20.2	What Happens at Runtime: How the RowLimit Property Works	7-48
7.21	Clearing the Values of Cached Attributes in an ADF Table Component	7-49
7.21.1	How to Clear the Values of Cached Attributes in an ADF Table Component	7-49
7.21.2	What Happens at Runtime: How the ADF Table Component Clears Cached Values	7-49
7.22	Tracking Changes in an ADF Table Component	7-50
7.23	Evaluating EL Expression for ReadOnly Properties	7-50
7.23.1	What Happens at Runtime: Evaluating EL Expression While Downloading Data ..	7-50
7.23.2	What Happens at Runtime: Evaluating EL Expression While Uploading Data or Tracking Changes	7-50
7.23.3	What You May Need to Know About Evaluating EL Expression While Uploading Data or Tracking Changes	7-51
7.24	Using Explicit Worksheet Setup Action	7-51
7.24.1	How to Configure Explicit Worksheet Setup Action	7-51
7.24.2	What You May Need to Know About Explicit Worksheet Setup Action	7-52

8 Adding Interactivity to Your Integrated Excel Workbook

8.1	About Adding Interactivity to an Integrated Excel Workbook	8-1
8.1.1	Adding Interactivity to Integrated Excel Workbook Use Cases and Examples	8-2
8.1.2	Additional Functionality for Adding Interactivity to an Integrated Excel Workbook	8-3
8.2	Using Action Sets	8-3
8.2.1	How to Invoke a Method Action Binding in an Action Set	8-5
8.2.2	How to Invoke Component Actions in an Action Set	8-5
8.2.3	What You May Need to Know About an Action Set Invoking a Component Action	8-7
8.2.3.1	Verifying an Action Set Invokes the Correct Component Action	8-7
8.2.3.2	Invoking Action Sets in a Disconnected Workbook	8-8
8.2.4	How to Invoke an Action Set from a Worksheet Event	8-8
8.2.5	How to Display a Status Message While an Action Set Runs	8-9
8.2.6	What Happens at Runtime: How the Action Set Displays a Status Message	8-11
8.2.7	What You May Need to Know About Progress Bars	8-12
8.2.8	What You May Need to Know About Canceling an Action	8-13
8.2.9	How to Provide an Alert After the Invocation of an Action Set	8-14
8.2.10	What Happens at Runtime: How the Action Set Provides an Alert	8-16
8.2.11	How to Configure Error Handling for an Action Set	8-16
8.2.12	How to Invoke a Confirmation Action in an Action Set	8-17

8.2.13	What Happens at Runtime: How the Action Set Provides a Confirmation	8-18
8.3	Configuring the Runtime Ribbon Tab	8-19
8.3.1	How to Define a Workbook Command Button for the Runtime Ribbon Tab	8-20
8.3.2	How to Configure a Worksheet Command for the Runtime Ribbon Tab	8-21
8.4	Displaying Web Pages from a Fusion Web Application	8-23
8.4.1	How to Display a Web Page in a Popup Dialog	8-23
8.4.2	How to Display a Web Page in ADF Desktop Integration Runtime Task Pane	8-25
8.4.3	What You May Need to Know About Displaying Pages from a Fusion Web Application	8-26
8.4.3.1	Keeping an Integrated Excel Workbook and a Fusion Web Application Synchronized	8-26
8.4.3.2	Sharing Data Control Frames Between Integrated Excel Worksheets and Fusion Web Application Pages	8-27
8.4.3.3	Configuring a Fusion Web Application for ADF Desktop Integration Frame Sharing	8-27
8.5	Adding a Custom Popup Picker Dialog to an ADF Table Column	8-28
8.6	Creating ADF Databound Search Forms in an Integrated Excel Workbook	8-29
8.6.1	How to Create a Search Form in an Integrated Excel Workbook	8-30
8.6.2	How to Create a Search Form using a Web Page in an Integrated Excel Workbook	8-32
8.7	Creating a Form in an Integrated Excel Workbook	8-34
8.8	Creating Dependent Lists of Values in an Integrated Excel Workbook	8-35
8.8.1	How to Create a Dependent List of Values in an Excel Worksheet	8-38
8.8.2	What Happens at Runtime: How the Excel Worksheet Renders a Dependent List of Values	8-39
8.8.3	How to Create a Dependent List of Values in an ADF Table Component's Columns	8-39
8.8.4	What Happens at Runtime: How the ADF Table Component Column Renders a Dependent List of Values	8-41
8.8.5	Creating a Dependent List of Values in an Excel Worksheet and an ADF Table Component Column	8-42
8.8.6	What Happens at Runtime: How the Excel Worksheet and the ADF Table Component Column Render a Dependent List of Values	8-44
8.9	Using EL Expression to Generate an Excel Formula	8-45
8.9.1	How to Configure a Cell to Display a Hyperlink Using EL Expression	8-46
8.9.2	What Happens at Runtime: How a Cell Displays a Hyperlink using an EL Expression	8-46
8.10	Using Calculated Cells in an Integrated Excel Workbook	8-47
8.10.1	How to Calculate the Sum of a Table-Type Component Column	8-48
8.10.2	What Happens at Runtime: How Excel Calculates the Sum of a Table-Type Component Column	8-48
8.11	Using Macros in an Integrated Excel Workbook	8-49

9 Configuring the Appearance of an Integrated Excel Workbook

9.1	About Configuring the Appearance of an Integrated Excel Workbook	9-1
9.1.1	Integrated Excel Workbook Configuration Use Cases and Examples	9-2
9.1.2	Additional Functionality for Configuring the Appearance of an Integrated Excel Workbook	9-2
9.2	Working with Styles	9-2
9.2.1	Predefined Styles in ADF Desktop Integration	9-2

9.2.2	Excel's Date Formats and Microsoft Windows' Regional and Language Options	9-3
9.2.3	How to Apply a Style to an Oracle ADF Component	9-4
9.2.4	What Happens at Runtime: How Style Is Applied to an ADF Component	9-5
9.3	Applying Styles Dynamically Using EL Expressions	9-6
9.3.1	What Happens at Runtime: How an EL Expression Is Evaluated	9-6
9.3.2	How to Write an EL Expression That Applies a Style at Runtime	9-7
9.3.3	What You May Need to Know About EL Expressions That Apply Styles	9-8
9.4	Using Labels in an Integrated Excel Workbook	9-8
9.4.1	Retrieving the Values of String Keys from a Resource Bundle	9-9
9.4.2	Retrieving the Values of Attribute Control Hints	9-9
9.4.3	How an Integrated Excel Workbook Evaluates a Label Property	9-10
9.5	Using Styles to Improve the User Experience	9-10
9.5.1	Using ADF Label Components to improve the User Experience	9-11
9.5.2	What You May Need to Know About the Read-Only Property in an Integrated Excel Workbook	9-11
9.6	Branding Your Integrated Excel Workbook	9-12
9.6.1	How to Brand an Integrated Excel Workbook	9-12
9.6.2	What Happens at Runtime: the BrandingItems Group of Properties	9-13
9.7	Using Worksheet Protection	9-14
9.7.1	How to Enable Worksheet Protection	9-14
9.7.2	What Happens at Runtime: How the Locked Property Works	9-15
9.7.3	What You May Need to Know About Worksheet Protection	9-16

10 Internationalizing Your Integrated Excel Workbook

10.1	About Internationalizing Your Integrated Excel Workbook	10-1
10.1.1	Internationalizing Integrated Excel Workbook Use Cases and Examples	10-2
10.1.2	Additional Functionality for Internationalizing Integrated Excel Workbook	10-2
10.2	Using Resource Bundles in an Integrated Excel Workbook	10-3
10.2.1	How to Register a Resource Bundle in an Integrated Excel Workbook	10-3
10.2.2	How to Override Resources That Are Not Configurable	10-4
10.2.3	What Happens at Runtime: Override Resources That Are Not Configurable	10-5
10.2.4	What You May Need to Know About Resource Bundles	10-5
10.2.4.1	Resource Bundle Types	10-5
10.2.4.2	Caching of Resource Bundles in an Integrated Excel Workbook	10-5
10.2.4.3	EL Expression Syntax for Resource Bundles	10-6
10.3	Localization in ADF Desktop Integration	10-6
10.3.1	Configuring Fusion Web Application to Override Server-Side Locale Settings	10-7
10.3.1.1	How to Create a User Preference Handler	10-7
10.3.1.2	How to Register the User Preference Handler	10-8

11 Securing Your Integrated Excel Workbook

11.1	About Security In Your Integrated Excel Workbook	11-1
11.1.1	Integrated Excel Workbook Security Use Cases and Examples	11-2
11.1.2	Additional Functionality for Integrated Excel Workbook in a Secure Fusion Web Application	11-2
11.2	Authenticating the Excel Workbook User	11-3

11.2.1	What Happens at Runtime: How the Login Method Is Invoked	11-3
11.2.2	What Happens at Runtime: How the Logout Method Is Invoked	11-4
11.3	Checking the Integrity of an Integrated Excel Workbook's Metadata	11-4
11.3.1	How to Reset the Workbook ID	11-5
11.3.2	What Happens When the Metadata Tamper-Check Is Performed	11-5
11.4	What You May Need to Know About Securing an Integrated Excel Workbook	11-6
11.5	Authorizing the Excel Workbook User	11-6
11.5.1	What You May Need to Know About ADF Desktop Integration-Disabled Worksheet .	11-7

12 Adding Validation to an Integrated Excel Workbook

12.1	About Adding Validation to an Integrated Excel Workbook	12-1
12.1.1	Integrated Excel Workbook Validation Use Cases and Examples	12-1
12.1.2	Additional Functionality for Adding Validation to an Integrated Excel Workbook	12-2
12.2	Providing Server-Side Validation for an Integrated Excel Workbook	12-2
12.3	Providing Client-Side Validation for an Integrated Excel Workbook	12-2
12.4	Error Reporting in an Integrated Excel Workbook	12-3
12.4.1	Error Reporting Using EL Expressions	12-3
12.4.2	Error Reporting Using Component Actions	12-4
12.5	Providing a Row-by-Row Status on an ADF Table Component	12-6
12.6	Adding Detail to Error Messages in an Integrated Excel Workbook	12-7
12.7	Handling Data Conflicts When Uploading Data from a Workbook	12-7
12.7.1	How to Configure a Workbook to Handle Data Conflicts When Uploading Data ..	12-8
12.7.2	What Happens at Runtime: How Data Conflicts Are Handled	12-8

13 Testing Your Integrated Excel Workbook

13.1	About Testing Your Integrated Excel Workbook	13-1
13.1.1	Integrated Excel Workbook Testing Use Cases and Examples	13-1
13.1.2	Additional Functionality of Testing an Integrated Excel Workbook	13-2
13.2	Testing Your Fusion Web Application	13-2
13.3	Validating the Integrated Excel Workbook Configuration	13-3
13.3.1	How to Validate the Integrated Excel Workbook Configuration	13-3
13.3.2	What Happens When You Validate the Integrated Excel Workbook Configuration	13-4
13.3.3	How to Fix Validation Failures	13-4
13.3.4	How to Log the Integrated Excel Workbook Configuration Validation Failures at Runtime	13-5
13.4	Testing Your Integrated Excel Workbook	13-6
13.5	Running a Server Ping Test	13-7

14 Deploying Your Integrated Excel Workbook

14.1	About Deploying Your Integrated Excel Workbook	14-1
14.1.1	Integrated Excel Workbook Deployment Use Cases and Examples	14-2
14.1.2	Additional Functionality for Deploying Your Integrated Excel Workbook	14-2
14.2	Making ADF Desktop Integration Available to End Users	14-2
14.3	Publishing Your Integrated Excel Workbook	14-3
14.3.1	How to Publish an Integrated Excel Workbook from Excel	14-3

14.3.2	How to Publish an Integrated Excel Workbook Using the Command Line Publish Tool	14-4
14.3.3	What Happens When You Publish an Integrated Excel Workbook	14-5
14.4	Deploying a Published Workbook with Your Fusion Web Application	14-6
14.4.1	What Happens at Runtime: Deploying a Published Workbook	14-7
14.5	Passing Parameter Values from a Fusion Web Application Page to a Workbook	14-8
14.5.1	How to Configure the Fusion Web Application's Page to Pass Parameters	14-9
14.5.2	How to Configure Parameters Properties in the Integrated Excel Workbook	14-11
14.5.3	How to Configure the Page Definition File for the Worksheet to Receive Parameters ...	14-14
14.5.4	What Happens at Runtime: How Parameters Are Passed from a Fusion Web Application to the Integrated Excel Workbook	14-15
14.6	Customizing Workbook Integration Metadata at Runtime	14-15
14.6.1	How to Enable Workbook Customization at Runtime	14-15
14.6.2	What Happens at Runtime: Workbook Integration Metadata is Customized	14-16
14.6.3	What You May Need to Know About Customizing Workbook Integration Metadata ...	14-16

15 Using an Integrated Excel Workbook Across Multiple Web Sessions and in Disconnected Mode

15.1	About Disconnected Workbooks	15-1
15.1.1	Disconnected Workbooks Use Cases and Examples	15-2
15.1.2	Additional Functionality for Disconnected Workbooks	15-2
15.2	Restore Server Data Context Between Sessions	15-2
15.2.1	How to Configure an Integrated Excel Workbook to Restore Server Data Context .	15-3
15.2.2	What Happens at Runtime: How the Integrated Excel Workbook Restores Server Data Context	15-4
15.3	Caching of Static Information in an Integrated Excel Workbook	15-4
15.4	Caching Lists of Values for Use in Disconnected Mode	15-5

A ADF Desktop Integration Component Properties and Actions

A.1	Frequently Used Properties in the ADF Desktop Integration	A-1
A.2	ADF Input Text Component Properties	A-3
A.3	ADF Output Text Component Properties	A-3
A.4	ADF Output Text (Worksheet Errors)	A-4
A.5	ADF Output Text (Table Errors)	A-4
A.6	ADF Label Component Properties	A-5
A.7	ADF List of Values Component Properties	A-5
A.8	ADF Image Component Properties	A-5
A.9	TreeNodeList Subcomponent Properties	A-6
A.10	ModelDrivenColumnComponent Subcomponent Properties	A-6
A.11	ADF Button Component Properties	A-7
A.12	ADF Table Component Properties and Actions	A-7
A.12.1	ADF Table Component Properties	A-7
A.12.2	ADF Table Component Column Properties	A-12
A.12.3	ADF Table Component Actions	A-14
A.13	ADF Read-only Table Component Properties and Actions	A-16

A.14	Action Set Properties	A-17
A.14.1	Confirmation Action Properties	A-19
A.14.2	Dialog Action Properties	A-20
A.15	Workbook Actions and Properties	A-20
A.16	Worksheet Actions and Properties	A-25

B ADF Desktop Integration EL Expressions

B.1	Guidelines for Creating EL Expressions	B-1
B.2	EL Syntax for ADF Desktop Integration Components	B-2
B.3	Attribute Control Hints in ADF Desktop Integration	B-4

C Troubleshooting an Integrated Excel Workbook

C.1	Verifying That Your Fusion Web Application Supports ADF Desktop Integration	C-1
C.2	Verifying End-User Authentication for Integrated Excel Workbooks	C-2
C.3	Generating Log Files for an Integrated Excel Workbook	C-2
C.3.1	About Server-Side Logging	C-2
C.3.2	About Client-Side Logging	C-3
C.3.2.1	How to Configure ADF Desktop Integration to Save Logs	C-3
C.3.2.2	About the ADF Desktop Integration Configuration File	C-5
C.3.2.3	How to Configure Logging Using User Environment Variables	C-6
C.3.2.4	What You May Need to Know About the adfdi-common Object	C-6
C.4	Common ADF Desktop Integration Error Messages and Problems	C-6

D ADF Desktop Integration Settings in the Web Application Deployment Descriptor

D.1	Configuring the ADF Desktop Integration Servlet	D-1
D.2	Configuring the ADF Desktop Integration Excel Download Filter	D-3
D.3	Examples in a Deployment Descriptor File	D-6

E String Keys in the Overridable Resources

F Java Data Types Supported By ADF Desktop Integration

G Using the ADF Desktop Integration Model API

G.1	About the Temporary Row Object	G-1
G.2	About ADF Desktop Integration Model API	G-2
G.2.1	How to Add ADF Desktop Integration Model API Library to Your JDeveloper Project G-2	
G.3	ADF Desktop Integration Model API Classes and Methods	G-3
G.3.1	The oracle.adf.desktopintegration.model.ModelHelper Class	G-3
G.3.1.1	The getAdfdiTempChildRow Method	G-3
G.3.1.2	The getAdfdiTempRowForView Method	G-4
G.3.1.3	The getChildViewDef Method	G-4

H End User Actions

H.1	Installing, Upgrading, and Removing the Runtime Edition of ADF Desktop Integration	H-1
H.1.1	How to Install Runtime Edition of ADF Desktop Integration	H-2
H.1.2	How to Remove the Runtime Edition of ADF Desktop Integration	H-3
H.1.3	How to Upgrade the Runtime Edition of ADF Desktop Integration On a Local System	H-3
H.2	Importing Data from a Non-Integrated Excel Worksheet	H-3
H.3	Removing Personal Information	H-4
H.4	Editing an Integrated Excel Workbook at Runtime	H-5
H.5	Limitations of an Integrated Excel Workbook at Runtime	H-5
H.6	Using An Integrated Excel Workbook	H-6
H.6.1	How to Insert a Row in an ADF Table Component of an Integrated Excel Workbook ...	H-6
H.6.2	How to Sort ADF Table Data in an Integrated Excel Workbook	H-7
H.6.3	How to Delete a Row in ADF Table of an Integrated Excel Workbook	H-7
H.7	Handling Time Zone Conversion	H-8

Preface

Welcome to the *Developing Applications with Oracle ADF Desktop Integration*.

Audience

This manual is intended for enterprise developers who configure desktop applications to integrate with the Oracle Application Development Framework (Oracle ADF).

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

For more information, see the following:

- *Developing Fusion Web Applications with Oracle Application Development Framework*
- *Developing Web User Interfaces with Oracle ADF Faces*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements (for example, menus and menu items, buttons, tabs, dialog controls), including options that you select.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.

Convention	Meaning
<code>monospace</code>	Monospace type indicates language and syntax elements, directory and file names, URLs, text that appears on the screen, or text that you enter.

What's New in This Guide

The following topics introduce the new and changed features of ADF Desktop Integration and other significant changes that are described in this guide, and provides pointers to additional information.

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>.

New and Changed Features for 12c (12.1.3)

Oracle ADF Desktop Integration 12c (12.1.3) includes the following new and changed development features for this document.

- New property, `AllowCancel`, for Status message. For more information, see [Section 8.2.5, "How to Display a Status Message While an Action Set Runs."](#)
- New component, ADF Image component, to add images to integrated Excel workbooks. For more information, see [Section 6.7, "Inserting an ADF Image Component"](#) and [Section A.8, "ADF Image Component Properties."](#)
- New properties (`ResizeMode` and `ResizeColumnsMode`) and a new action (`ResizeColumns`) for ADF Table components. For more information, see [Section 7.17, "Configuring Column Widths in an ADF Table Component."](#)
- New components, ADF Output Text (Worksheet Errors) and ADF Output Text (Table Errors), to display worksheet and table error messages. For more information, see [Section 6.8, "Inserting Error Components."](#)
- New property, `Tooltip`, to display tooltips for Form-type and Table-type components. For more information, see [Section 6.9, "Displaying Tooltips in ADF Desktop Integration Form-Type Components"](#) and [Section 7.18, "Displaying Tooltips in ADF Desktop Integration Table-Type Components."](#)
- Customize workbook integration metadata at runtime. For more information, see [Section 14.6, "Customizing Workbook Integration Metadata at Runtime."](#)

Other Significant Changes in this Document for 12c (12.1.3)

For 12c (12.1.3), this document has been updated in several ways, to include corrections and clarifications. In addition, the following significant changes or additions have occurred in these sections:

- Updated [Section 11.3, "Checking the Integrity of an Integrated Excel Workbook's Metadata"](#) to add a note that tamper-check is not performed for customization-enabled workbooks.
- Updated [Section 7.8.2, "What Happens at Runtime: How the ADF Table Component Uploads Data"](#) and [Section 7.10.2, "What Happens at Runtime: How the ADF Table Component Deletes Rows in a Fusion Web Application"](#) to add a note that when the `DeleteFlaggedRows` action is invoked on an ADF Table that has an Excel filter applied, the filter's criteria is cleared to show any hidden Excel worksheet rows.
- Updated [Section 14.3, "Publishing Your Integrated Excel Workbook"](#) to add a note that customization-enabled workbooks can only be published to a target location that is under the `public_html` directory.
- Updated [Section 14.5.2, "How to Configure Parameters Properties in the Integrated Excel Workbook"](#) to add information about Workbook Parameter dialog that prompts you to enter test parameter values while entering test mode.

Introduction to ADF Desktop Integration

This chapter introduces ADF Desktop Integration and provides an overview of the framework. The chapter also describes the advantages of integrating Microsoft Excel with a Fusion web application.

This chapter includes the following sections:

- [Section 1.1, "About ADF Desktop Integration"](#)
- [Section 1.2, "About ADF Desktop Integration with Microsoft Excel"](#)

1.1 About ADF Desktop Integration

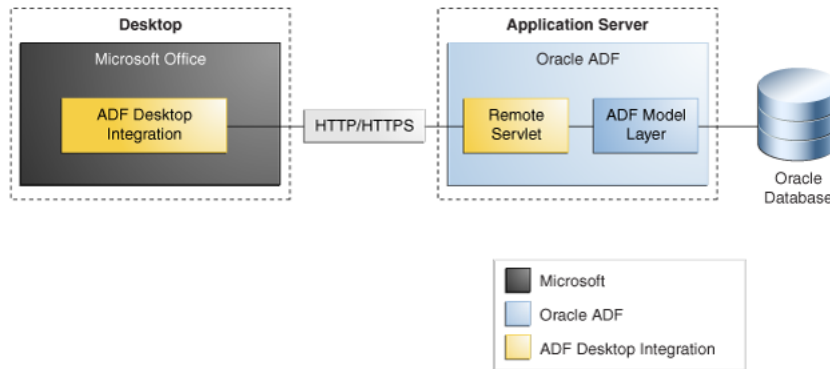
Many end users of Fusion web applications use desktop applications, such as Microsoft Excel, to manage information also used by their web application. ADF Desktop Integration provides a framework for Oracle Application Development Framework (Oracle ADF) developers to extend the functionality provided by a Fusion web application to desktop applications. It allows end users to avail themselves of Oracle ADF functionality when they are disconnected from their company network. End users may also prefer ADF Desktop Integration because it provides Excel's familiar user interface to undertake information management tasks, such as performing complex calculations or uploading a large amount of data, easily and seamlessly.

ADF Desktop Integration is a part of the Oracle ADF architecture. More information about the Oracle ADF architecture can be found in the "Oracle ADF Architecture" section of *Developing Fusion Web Applications with Oracle Application Development Framework*.

[Figure 1-1](#) illustrates the architecture of ADF Desktop Integration, which comprises of the following components:

- ADF Desktop Integration
- ADF Desktop Integration remote servlet
- ADF Model layer

Figure 1–1 ADF Desktop Integration Architecture



For more information about ADF Desktop Integration, see the ADF Desktop Integration page on Oracle Technology Network (OTN) at:

<http://www.oracle.com/technetwork/developer-tools/adf/overview/index-085534.html>

1.2 About ADF Desktop Integration with Microsoft Excel

Currently, ADF Desktop Integration supports integration with Microsoft Excel 2007, and other higher versions of Microsoft Excel.

Note: This guide uses the term *integrated Excel workbook* to refer to Excel workbooks that you integrate with a Fusion web application and to distinguish these workbooks from workbooks that have not been integrated with a Fusion web application or configured with Oracle ADF functionality.

1.2.1 Overview of Creating an Integrated Excel Workbook

Creating an integrated Excel workbook involves the steps described in [Table 1–1](#).

Table 1–1 Steps to Create an Integrated Excel Workbook

Use	To
JDeveloper	<ul style="list-style-type: none"> ■ Create a Fusion web application. For information about creating a Fusion web application, see the <i>Developing Fusion Web Applications with Oracle Application Development Framework</i>. ■ Add data controls that expose the elements you require in Microsoft Excel. ■ Create page definition files that expose the Oracle ADF bindings to use in Excel. For more information, see Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook."

Table 1–1 (Cont.) Steps to Create an Integrated Excel Workbook

Use	To
Excel	<ul style="list-style-type: none"> <li data-bbox="634 260 1442 380"> <p>■ Create the Excel workbooks that you intend to configure with Oracle ADF functionality.</p> <p>For more information, see Section 4.2, "Adding an Integrated Excel Workbook to a Fusion Web Application."</p> <li data-bbox="634 394 1442 1677"> <p>■ Configure the Excel workbook using the Oracle ADF bindings that you exposed in the page definition files and the Oracle ADF components that ADF Desktop Integration provides.</p> <p>For more information, see the following sections and chapters:</p> <ul style="list-style-type: none"> <li data-bbox="889 527 1442 730"> <p>– Chapter 5, "Getting Started with the Development Tools"</p> <p>This chapter provides an overview of the tools that ADF Desktop Integration provides to configure an Excel workbook with Oracle ADF functionality.</p> <li data-bbox="889 745 1442 989"> <p>– Chapter 6, "Working with ADF Desktop Integration Form-Type Components"</p> <p>This chapter describes how to insert ADF Desktop Integration form-type components into Excel worksheets and configure their properties to determine behavior at runtime.</p> <li data-bbox="889 1003 1442 1247"> <p>– Chapter 7, "Working with ADF Desktop Integration Table-Type Components"</p> <p>This chapter describes how to use the ADF Table and Read-only Table components to provide end users with a means of displaying and editing data hosted by a Fusion web application.</p> <li data-bbox="889 1262 1442 1430"> <p>– Chapter 12, "Adding Validation to an Integrated Excel Workbook"</p> <p>This chapter describes how to provide validation for your integrated Excel workbook.</p> <li data-bbox="634 1444 1442 1543"> <p>■ Test your integrated Excel workbook.</p> <p>For more information, see Chapter 13, "Testing Your Integrated Excel Workbook."</p> <li data-bbox="634 1558 1442 1677"> <p>■ After completing the integration of the Excel workbook with the Fusion web application, you deploy it to make it available to the end users.</p> <p>For information about this task, see Chapter 14, "Deploying Your Integrated Excel Workbook."</p>

1.2.2 Advantages of Integrating Excel with a Fusion Web Application

Advantages that accrue from integrating Microsoft Excel workbooks with your Fusion web application include:

- Providing end users with access to data and functionality hosted by a Fusion web application through a desktop interface (Microsoft Excel) that may be more familiar to them.
- End users can access data hosted by a Fusion web application while not connected to the application. They must log on to the Fusion web application to download data. Once data is downloaded to an Excel workbook, they can modify it while disconnected from the Fusion web application.
- Bulk entry and update of data may be easier to accomplish through a spreadsheet-style interface.
- End users can use native Excel features such as macros and calculation.

Introduction to the ADF Desktop Integration Sample Application

This chapter provides an overview of the Summit sample application for ADF Desktop Integration. The Summit sample application for ADF Desktop Integration contains several Microsoft Excel workbooks that are integrated with the sample's Fusion web application.

This chapter includes the following sections:

- [Section 2.1, "About the Summit Sample Application for ADF Desktop Integration"](#)
- [Section 2.2, "Setting Up and Running the Summit Sample Application for ADF Desktop Integration"](#)
- [Section 2.3, "Overview of the Fusion Web Application in the Summit Sample Application for ADF Desktop Integration"](#)
- [Section 2.4, "Overview of the Integrated Excel Workbooks in the Summit Sample Application for ADF Desktop Integration"](#)

2.1 About the Summit Sample Application for ADF Desktop Integration

The Summit sample application for ADF Desktop Integration is a set of sample demonstrations that illustrate the main capabilities from ADF Desktop Integration. Each of the samples contain specific features that can also be identified on the developer's guide. All of the samples use the same underlying database schema which makes it very easy for accessing the source code, and also to experience the runtime behavior in a standalone way.

2.2 Setting Up and Running the Summit Sample Application for ADF Desktop Integration

Set up the development environment as described in [Chapter 3, "Setting Up Your Development Environment"](#) before you download and run the Summit sample application for ADF Desktop Integration.

To download the Summit sample application for ADF Desktop Integration:

1. Download and install Oracle JDeveloper Release 12c. For more information, see *Installing Oracle JDeveloper*.
2. Download the Summit sample application for ADF Desktop Integration ZIP file (SummitADF_DI1213.zip) from Oracle Technology Network.

http://www.oracle.com/pls/topic/lookup?ctx=E26099_01&id=jdevcodesamples

3. Download the `SummitADF_Schema1213.zip` file and install the Summit ADF schema. For more information, see the "How to Install the Summit ADF Schema" section in *Developing Fusion Web Applications with Oracle Application Development Framework*.
4. Install ADF Desktop Integration. For more information, see [Section 3.4, "Installing ADF Desktop Integration."](#)

Note: If you have an old version of ADF Desktop Integration installed on your system, upgrade ADF Desktop Integration as described in [Section 3.6, "Upgrading ADF Desktop Integration."](#)

To run the Summit sample application for ADF Desktop Integration:

1. Extract the contents of `SummitADF_DI1213.zip` file to a local directory.
2. Open the `SummitADFdi.jws` file in JDeveloper.
This file is located in the `Summit_ADFDI` directory.
3. In the Applications window, click and expand the Model project.
4. Open **Model > Application Sources > oracle.summitdi.model > Model.jpx** file.
5. Expand the Connection group of the General tab, and click the **Add** icon to create a database connection.
6. In the Create Database Connection dialog, add the connection information shown in [Table 2–1](#) for your environment.

Table 2–1 Database Connection Properties for the Summit Sample Application for ADF Desktop Integration

Property	Description
Username	summit_adf
Password	summit_adf
Host Name	The host name for your database. For example: localhost
JDBC Port	The port for your database. For example: 1521
SID	The SID of your database. For example: ORCL or XE

Click **Test Connection** to verify the connection, and then click **OK** to close the dialog.

7. Save the `Model.jpx` file.
8. Right-click **Model** project and choose **Rebuild Model.jpr**.
9. Expand the **ViewController** project and choose **Web Content > MainPage.jsf**.
10. Right-click **MainPage.jsf** and choose **Run**.

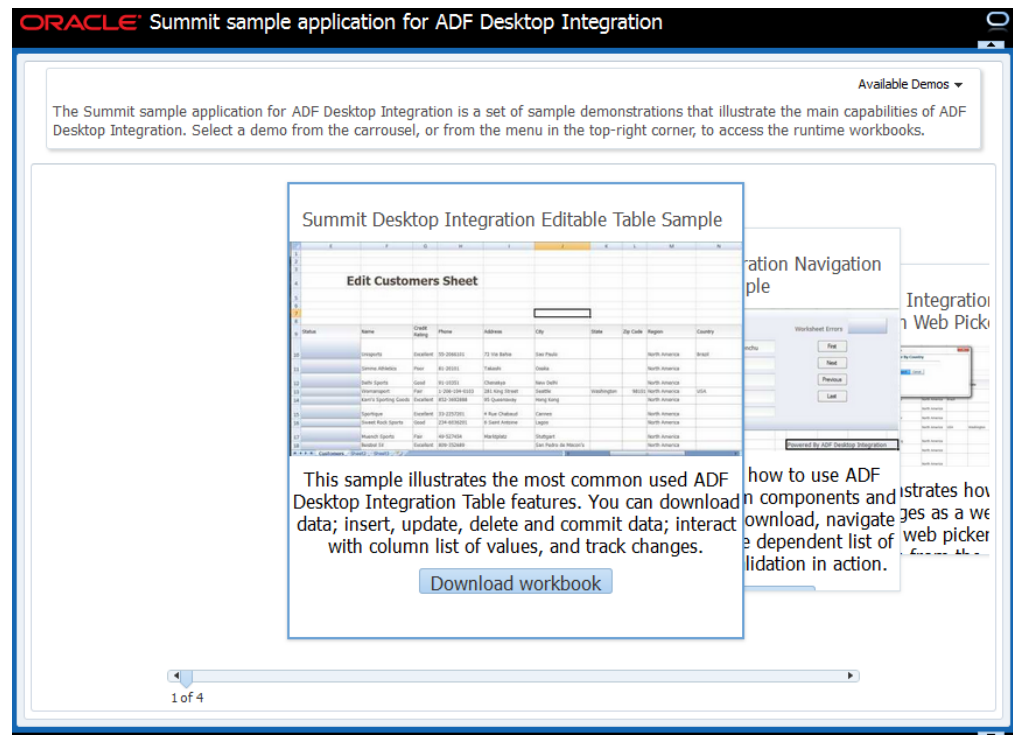
2.3 Overview of the Fusion Web Application in the Summit Sample Application for ADF Desktop Integration

The Fusion web application in the Summit sample application for ADF Desktop Integration enables end users to download the integrated Excel workbooks.

2.3.1 About the Fusion Web Application in the Summit Sample Application for ADF Desktop Integration

When the end user runs the Summit sample application for ADF Desktop Integration in JDeveloper, the default browser opens the application home page. The end user can download various integrated Excel workbooks from the home page.

Figure 2–1 Home page of Summit Sample Application for ADF Desktop Integration



2.3.2 Downloading Integrated Excel Workbooks

The Summit sample application for ADF Desktop Integration provides various integrated Excel workbooks to meet different requirements. End users can navigate and download different workbooks using the slider control, or from the **Available Demos** dropdown list.

Table 2–2 lists the menu options and the downloaded integrated Excel workbooks.

Table 2–2 Integrated Excel Workbooks of Summit sample application for ADF Desktop Integration

Menu Option	Description
Editable Table Sample	Downloads EditCusotmers.xlsx workbook.
Navigation Form Sample	Downloads EditWarehouses.xlsx workbook.

Table 2–2 (Cont.) Integrated Excel Workbooks of Summit sample application for ADF Desktop Integration

Menu Option	Description
Editable Table with Web Picker	Downloads EditableCusotmerSearch.xlsx workbook.
Form and Table Quick Location Change	Downloads WarehouseLocations.xlsx workbook.

2.4 Overview of the Integrated Excel Workbooks in the Summit Sample Application for ADF Desktop Integration

The Summit sample application for ADF Desktop Integration provides the EditCustomers.xlsx, EditWarehouses.xlsx, EditCustomerSearch.xlsx, and WarehouseLocations.xlsx integrated Excel workbooks.

The EditCustomers.xlsx workbook enable end users to:

- Download and view customer data in a tabular format
- Modify and upload the information in the workbook
- Upload multiple rows of data

The EditWarehouses.xlsx workbook enable end users to:

- Download and view warehouse data in a navigational format
- Modify and upload warehouse information in the workbook
- Upload the updated information immediately when user navigates to another record.

The EditCustomerSearch.xlsx workbook enable end users to:

- Download and view customer data in a tabular format
- Filter customers by Country

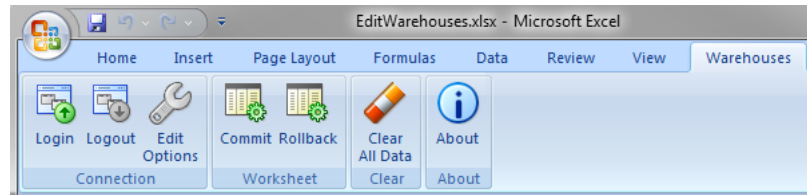
The WarehouseLocations.xlsx workbook enable end users to:

- Download and view warehouse data and their locations in a tabular format
- Modify and update the region of all warehouses
- Upload the updated information

Subsequent sections in this chapter provide more information about the functionality in the workbooks along with cross-references to implementation details.

2.4.1 Log on to the Fusion Web Application from an Integrated Excel Workbook

At runtime, the integrated Excel workbooks in the Summit sample application for ADF Desktop Integration render an Excel ribbon tab that allows end users to log on to the Fusion web application. [Figure 2–2](#) shows the runtime Warehouses tab in the Ribbon of the EditWarehouses.xlsx workbook.

Figure 2–2 Runtime Warehouses Tab

2.4.2 Downloading Data Rows

Some workbooks, such as `EditCustomers.xlsx` workbook, use an ADF Table component to host information downloaded from the Fusion web application. This component allows end users to edit rows and upload modified rows to the Fusion web application.

The following sections provide information about how to implement the download functionality:

- Each worksheet that you integrate with a Fusion web application requires an associated page definition file.

For example, the `Customers` worksheet in the `EditCustomers.xlsx` workbook is associated with the `ExcelCustomers.xml` page definition file. In JDeveloper, expand the following nodes in the Applications window to view this file:

ViewController > Application Sources > oracle.summitdi.view > pageDefs

For information about how to configure a page definition file, see [Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook."](#)

- The ADF Table component `Download` action downloads data from the Fusion web application to the worksheet. For information about how you invoke this action, see [Section 7.4, "Configuring Oracle ADF Component to Download Data to an ADF Table Component."](#)
- In the `EditCustomers.xlsx` workbook, the worksheet `Startup` event invokes an action set that includes the ADF Table component `Download` action. For information about configuring worksheet events, see [Section 8.2.4, "How to Invoke an Action Set from a Worksheet Event."](#)

2.4.3 Modify Customers and Warehouses Information in the Workbooks

The `EditCustomers.xlsx` and `EditWarehouses.xlsx` workbooks enable end users to edit customers and warehouses information that the ADF Table component and form components downloads from the Fusion web application. Columns in the runtime ADF Table component that have an `UpdateComponent` property configured permit end users to modify values and upload the changes to the Fusion web application. For example, end users can modify the values that appear in the **Name**, **Phone**, and **Address** columns in `EditCustomers.xlsx`.

Other columns, such as **Status** and **Changed**, appear in the ADF Table component to provide status information about upload operations and changed columns.

The following sections provide information about how to implement this functionality:

- For information about inserting an ADF Table component, see [Section 7.3, "Inserting ADF Table Component into Excel Worksheet."](#)

- For information about using Excel formulas, see [Section 8.10, "Using Calculated Cells in an Integrated Excel Workbook."](#)
- For information about special columns, such as **Status** and **Changed**, see [Section 7.12, "Special Columns in the ADF Table Component."](#)

2.4.4 Upload Modified Information to the Fusion Web Application

The integrated workbooks allow end users to upload modified data in the ADF Table component to the Fusion web application. An action set is configured for the runtime **Upload** button that invokes the ADF Table component's `Upload` action. For information about implementing this functionality, see [Section 7.8, "Configuring an ADF Component to Upload Changes from an ADF Table Component."](#)

Setting Up Your Development Environment

This chapter describes how to set up the development environment to integrate an Excel workbook with a Fusion web application, how to upgrade and remove ADF Desktop Integration, and how to localize the Visual Studio tools for Microsoft Office.

This chapter includes the following sections:

- [Section 3.1, "About Setting Up Your Development Environment"](#)
- [Section 3.2, "Required Oracle ADF Modules and Third-Party Software"](#)
- [Section 3.3, "Configuring Excel to work with ADF Desktop Integration"](#)
- [Section 3.4, "Installing ADF Desktop Integration"](#)
- [Section 3.5, "Removing ADF Desktop Integration"](#)
- [Section 3.6, "Upgrading ADF Desktop Integration"](#)
- [Section 3.7, "Using an Integrated Excel Workbook with Older Versions of ADF Desktop Integration"](#)
- [Section 3.8, "Using ADF Desktop Integration on a System with Multiple Instances of JDeveloper"](#)

3.1 About Setting Up Your Development Environment

Setting up the development environment involves making sure that you have the correct versions of JDeveloper, Microsoft Office, and Microsoft Internet Explorer installed, as described in [Section 3.2, "Required Oracle ADF Modules and Third-Party Software."](#)

After verifying that you have the required software, complete the setup of your development environment by:

- Allowing ADF Desktop Integration to access Microsoft Excel
- Installing ADF Desktop Integration

Note: The instructions in this guide assume that you are using Windows 7 operating system and Microsoft Excel 2007. Note that the steps might be different for different editions of Windows and Excel.

3.2 Required Oracle ADF Modules and Third-Party Software

Before you begin to integrate your Excel workbook with a Fusion web application, ensure that you have the required Oracle ADF modules and third-party software installed and configured:

- Oracle JDeveloper

Install the current release of JDeveloper. ADF Desktop Integration is available as a JDeveloper feature.

- Microsoft Windows

Microsoft Windows operating systems support the development and deployment of Excel workbooks that integrate with Fusion web applications. For more information about supported versions of Windows, see the "Oracle JDeveloper and Application Development Framework Certification Information" page on OTN at:

<http://www.oracle.com/technetwork/developer-tools/jdev/documentation/121300-cert-2164864.html>

- Microsoft Excel

ADF Desktop Integration supports the integration of Fusion web applications with the following types of Excel workbook:

- Excel Workbook

The default file format for Excel workbooks is the Excel XML-based file format (.xlsx).

- Excel Macro-Enabled Workbook

Workbooks in this format (.xlsm) use the Excel XML-based file format and can store VBA macro code.

ADF Desktop Integration does not support the use of other Excel file formats. For more information about supported versions of Excel, see the "Oracle JDeveloper and Application Development Framework Certification Information" page on OTN at:

<http://www.oracle.com/technetwork/developer-tools/jdev/documentation/121300-cert-2164864.html>

Note: Microsoft Excel 2003 or older versions of Microsoft Excel are not supported.

- Internet Explorer

Some features in ADF Desktop Integration use a web browser control from the Microsoft .NET Framework. This browser control relies on the local Internet Explorer installation to function properly.

ADF Desktop Integration uses Internet Explorer to render web pages inside Excel, regardless of other browsers installed on the system or any other browser set as the default browser.

- Application server

For information about the application servers that you can use to deploy an application developed using ADF Desktop Integration, see the "Oracle JDeveloper

and Application Development Framework Certification Information" page on OTN at:

<http://www.oracle.com/technetwork/developer-tools/jdev/documentation/121300-cert-2164864.html>

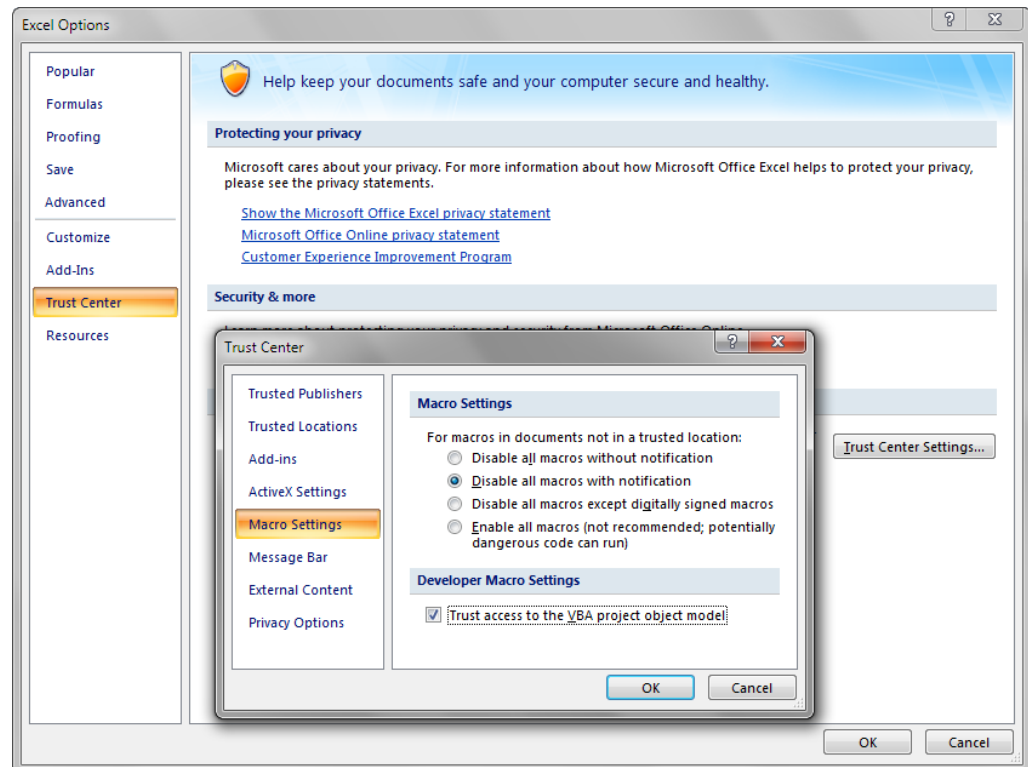
3.3 Configuring Excel to work with ADF Desktop Integration

You must configure Microsoft Excel settings to make it accessible from ADF Desktop Integration. You only need to perform this procedure once.

To allow Excel to run an integrated Excel workbook:

1. Open Excel.
2. Click the **Microsoft Office** button, and choose **Excel Options**.
3. In the Excel Options dialog, choose the **Trust Center** tab, and then click **Trust Center Settings**.
4. In the Trust Center dialog, choose the **Macro Settings** tab, and then click the **Trust access to the VBA project object model** checkbox, as shown in [Figure 3-1](#).

Figure 3-1 Excel Trust Center Dialog



5. Click **OK**.

For more information about securing an Excel workbook that is integrated with a Fusion web application, see [Chapter 11, "Securing Your Integrated Excel Workbook."](#)

3.4 Installing ADF Desktop Integration

When you run the ADF Desktop Integration setup tool, it verifies whether software in the following list is installed on the system where you want to install the framework. If one or more of these pieces of software is not installed, the setup tool automatically downloads and installs it in the order specified.

1. Windows Installer 3.1
2. Microsoft .NET Framework

The Microsoft .NET Framework 4 provides the runtime and associated files required to run applications developed to target the Microsoft .NET Framework. You can download the framework from

<http://www.microsoft.com/download/>.

Notes:

- Do not download the Client Profile edition of Microsoft .NET Framework as it is insufficient to run ADF Desktop Integration.
 - Installation of Microsoft .NET Framework may require you to restart the system where you install it. After the restart, the setup tool automatically recommences to finalize installation.
-
-

3. Microsoft Visual Studio 2010 Tools for Office Runtime

The Microsoft Visual Studio 2010 Tools for Office Runtime (version 4) is required to run VSTO solutions for the Microsoft Office system. You can download the Microsoft Visual Studio 2010 Tools for Office Runtime from

<http://www.microsoft.com/download/>.

4. ADF Desktop Integration add-in

You can install the ADF Desktop Integration add-in from JDeveloper, or from the setup tool provided in *MW_HOME\jdeveloper\adfdi*. For more information about how to set up ADF Desktop Integration, see [Section 3.4.1, "How to Set Up ADF Desktop Integration."](#)

Note that the ADF Desktop Integration installation is specific to the current Windows user profile. If you have multiple Windows user profiles on your system, and you want to use ADF Desktop Integration integrated Excel workbooks from some specific user profiles, you must log in to each user profile and install the ADF Desktop Integration add-in. For more information, see [Section 3.4.1, "How to Set Up ADF Desktop Integration."](#)

3.4.1 How to Set Up ADF Desktop Integration

The ADF Desktop Integration add-in is available in two editions, the Designer edition and the Runtime edition. Use the Designer edition to create and test integrated Excel workbooks, and the Runtime edition to enable end users to use ADF Desktop Integration and integrated Excel workbooks.

Note: Do not install both editions of ADF Desktop Integration on the same system.

Although you do not require administrator privileges to install the ADF Desktop Integration add-in, administrator privileges may be required to run the installers for additional software that the installer attempts to download and install. You should also ensure that the proxy settings for Internet Explorer are configured to allow access to *.microsoft.com because the installer attempts to automatically download missing prerequisite software from Microsoft's website.

By default, the installer runs in English. You can change the language that appears by following the instructions in the "Localizing the ADF Desktop Integration Installer" section of *Administering Oracle ADF Applications*.

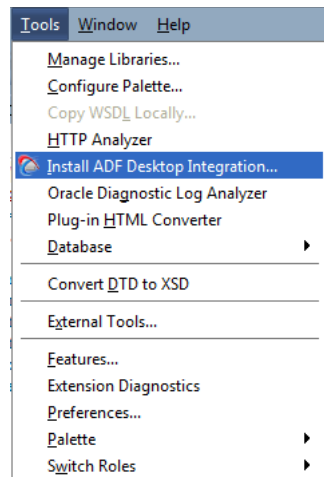
Before you begin:

It may be helpful to have an understanding of ADF Desktop Integration requirements. For more information, see [Section 3.4, "Installing ADF Desktop Integration."](#)

To install the Designer edition of ADF Desktop Integration:

1. Open JDeveloper.
2. From the **Tools** menu, choose **Install ADF Desktop Integration**, as shown in [Figure 3–2](#).

Figure 3–2 Tools Menu in JDeveloper



Note: The **Install ADF Desktop Integration** menu option is available only on the Windows installation of JDeveloper.

3. Follow the instructions that appear in the dialog boxes to successfully install the required components.

If you encounter an error during the installation process, ensure that you have removed the previous version of ADF Desktop Integration. For more information, see [Section 3.5, "Removing ADF Desktop Integration."](#)

4. If prompted, click **Yes** to restart the system and complete the setup of ADF Desktop Integration.

Tip: You can also install the Designer edition of ADF Desktop Integration by running `setup.exe` available in the `MW_HOME\jdeveloper\adfdi\bin\excel\addin\designer` directory.

If multiple instances of JDeveloper are installed, or if you have an existing instance of the ADF Desktop Integration add-in on the system, review the information in [Section 3.8, "Using ADF Desktop Integration on a System with Multiple Instances of JDeveloper"](#) before you perform the installation procedure.

If you want to install the Runtime edition of ADF Desktop Integration, see [Section H.1, "Installing, Upgrading, and Removing the Runtime Edition of ADF Desktop Integration."](#)

3.5 Removing ADF Desktop Integration

Use the Microsoft Windows Control Panel to remove the ADF Desktop Integration add-in from the system where you set it up. After removing ADF Desktop Integration, you can no longer use integrated Excel workbooks on this system unless you reinstall ADF Desktop Integration.

To remove the ADF Desktop Integration add-in:

1. Click the Windows **Start** button, and then choose **Control Panel**.
2. In the Control Panel, select and open **Programs and Features**.
3. Select the **Oracle ADF Desktop Integration Add-in for Excel** entry in the Uninstall or change a program window and click **Uninstall**.

Note: If you have installed ADF Desktop Integration on multiple user profiles, you must remove it from each user profile.

3.6 Upgrading ADF Desktop Integration

If you are using an old version of ADF Desktop Integration, you must upgrade to the current version.

To upgrade the ADF Desktop Integration add-in:

1. Uninstall the old version of ADF Desktop Integration.
For more information, see [Section 3.5, "Removing ADF Desktop Integration."](#)
2. Download and install the latest version of Oracle JDeveloper.
3. Install the new version of ADF Desktop Integration.
For more information, see [Section 3.4, "Installing ADF Desktop Integration."](#)

Note: If you do not uninstall the old version of ADF Desktop Integration, an error occurs unless the new installer is in the exact same location as the old installer.

3.6.1 How to Migrate an Integrated Excel Workbook to the Current Version of ADF Desktop Integration

When you open the integrated Excel workbook after upgrading the ADF Desktop Integration add-in, the add-in detects and compares the ADF Desktop Integration version information of the workbook with the version installed on the client system. If required, you are asked to upgrade the configuration of the integrated workbook to the version installed on the client.

Note: Integrated Excel workbooks created using an older ADF Desktop Integration client (version X) do not require migration on a system running a new version of ADF Desktop Integration client (version X+1), but integrated Excel workbooks used and saved with a newer client (version X+1) may no longer work with older clients (version X).

When the integrated Excel workbook is not compatible with the installed version of the ADF Desktop Integration client, a message is displayed when you open the workbook. In such a case, you should install the newer version of the ADF Desktop Integration client in order to interact with the newer workbook.

If you are migrating your integrated Excel workbooks created using the ADF Desktop Integration client of 11.1.1.3.0 version, or older, the Migrate Workbook dialog appears confirming your action.

Before you begin:

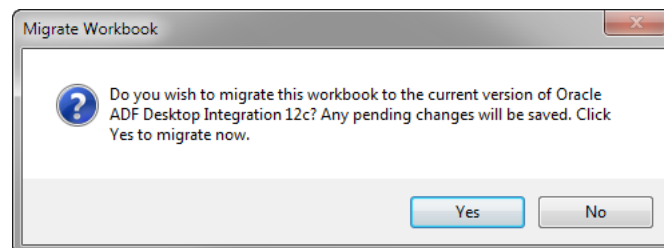
It may be helpful to have an understanding of ADF Desktop Integration upgrade. For more information, see [Section 3.6, "Upgrading ADF Desktop Integration."](#)

To migrate an integrated Excel workbook after upgrading:

1. Open the integrated Excel workbook.

The Migrate Workbook dialog prompts you to migrate the workbook to the current version of ADF Desktop Integration, as shown in [Figure 3-3](#).

Figure 3-3 Migrate Workbook Dialog



If you get one or more Microsoft Office Customization Installer error messages when you open the integrated Excel workbook, ignore the messages and continue. The error messages appear because ADF Desktop Integration cannot remove the old version information from the workbook before Excel detects it and reports the error.

2. Click **Yes** to migrate the workbook. The ADF Desktop Integration migration process closes the workbook and then reopens it, ready to be used with the current version of ADF Desktop Integration.

3.7 Using an Integrated Excel Workbook with Older Versions of ADF Desktop Integration

When you open a workbook created, or last updated, by a newer version of ADF Desktop Integration on a system running an older version of ADF Desktop Integration, an error message is displayed if the workbook is incompatible with the

installed client version. An integrated workbook and the installed ADF Desktop Integration client are incompatible if the first element of their Oracle Release number is different. For example, an integrated workbook created using version 12.1.3 of ADF Desktop Integration client is incompatible with the client from version 11.1.1.7.0.

Tip: To see the release number of the installed client and integrated workbook, examine the values of **ADF Desktop Integration** and **Workbook Edited by Version** entries in Versions tab of the About ADF Desktop Integration dialog box.

At runtime, ADF Desktop Integration ignores an incompatible workbook. The data in the workbook is not removed, but ADF Desktop Integration treats the workbook as a non-integrated workbook and the following functionalities are affected:

- ADF Desktop Integration does not respond to workbook events (for example, Activate and Startup)
- ADF Button components do not respond to events
- Worksheet command buttons do not appear in the Excel Ribbon

In design mode, or test mode, the workbook continues to function as expected, but you might notice some unknown exceptions and functionality issues.

3.8 Using ADF Desktop Integration on a System with Multiple Instances of JDeveloper

There can be only one active installation of ADF Desktop Integration on a given system. By default, when you install JDeveloper, ADF Desktop Integration is extracted to `MW_HOME\jdeveloper\adfdi`. If you move to another version of JDeveloper installed in a different directory, you should remove the old version of ADF Desktop Integration, as described in [Section 3.5, "Removing ADF Desktop Integration."](#) You should then set up ADF Desktop Integration from the new version of JDeveloper, as described in [Section 3.4, "Installing ADF Desktop Integration,"](#) to keep ADF Desktop Integration consistent with JDeveloper.

Alternatively, you can set up ADF Desktop Integration in a directory that is independent of your JDeveloper installation. This approach means that you do not have to remove ADF Desktop Integration before moving to a newer version.

To set up ADF Desktop Integration in an independent directory:

1. Create a directory independent of the JDeveloper installation directory. For example, you may create the following directory:

```
D:\adfdi-excel-setup
```

2. When you move to a newer version of JDeveloper, copy the contents of the following directory:

```
MW_HOME\jdeveloper\adfdi\bin\excel\addin\designer
```

to:

```
D:\adfdi-excel-setup
```

where `MW_HOME` is the Middleware Home directory.

3. Run the `setup.exe` tool located in `D:\adfdi-excel-setup`.
4. Follow the instructions that appear in the dialog boxes launched by `setup.exe` to set up the new version of ADF Desktop Integration.

5. If prompted, click **Yes** to restart the system and complete the setup of ADF Desktop Integration.

WARNING: After you install ADF Desktop Integration, do not delete the directory where you copied the setup files. You can delete the files after removing ADF Desktop Integration from the system.

Preparing Your Integrated Excel Workbook

This chapter describes how to prepare Excel workbooks and integrate them with Fusion web applications using ADF Desktop Integration, how to use the page definition files with an integrated Excel workbook, and how to enable ADF Desktop Integration manually to integrate an existing workbook with the Fusion web application.

This chapter includes the following sections:

- [Section 4.1, "About Preparing Your Integrated Excel Workbooks"](#)
- [Section 4.2, "Adding an Integrated Excel Workbook to a Fusion Web Application"](#)
- [Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook"](#)
- [Section 4.4, "Enabling ADF Desktop Integration Manually"](#)

4.1 About Preparing Your Integrated Excel Workbooks

This chapter (and the guide as a whole) assumes that you have developed a functioning Fusion web application, as described in the *Developing Fusion Web Applications with Oracle Application Development Framework*.

Having developed the Fusion web application, you perform the tasks described in this chapter to configure an integrated Excel workbook with the Fusion web application. The subsequent chapters of the guide enable you to configure the integrated workbook with advanced features, such as configuration with Oracle ADF functionality, and adding Oracle ADF components that provide the functionality you require at runtime.

Note: Before you start, ensure that the Designer edition of ADF Desktop Integration is installed. For more information about the ADF Desktop Integration editions, see [Section 3.4, "Installing ADF Desktop Integration."](#)

4.2 Adding an Integrated Excel Workbook to a Fusion Web Application

The Fusion web application is automatically enabled with ADF Desktop Integration when you add an integrated Excel workbook to a project. An integrated Excel workbook enables you to add ADF components and ADF data bindings.

4.2.1 How to Add an Integrated Excel Workbook to a Fusion Web Application

To add an integrated Excel workbook, open the Fusion web application in JDeveloper and add an Excel workbook to the project from New Gallery.

Before you begin:

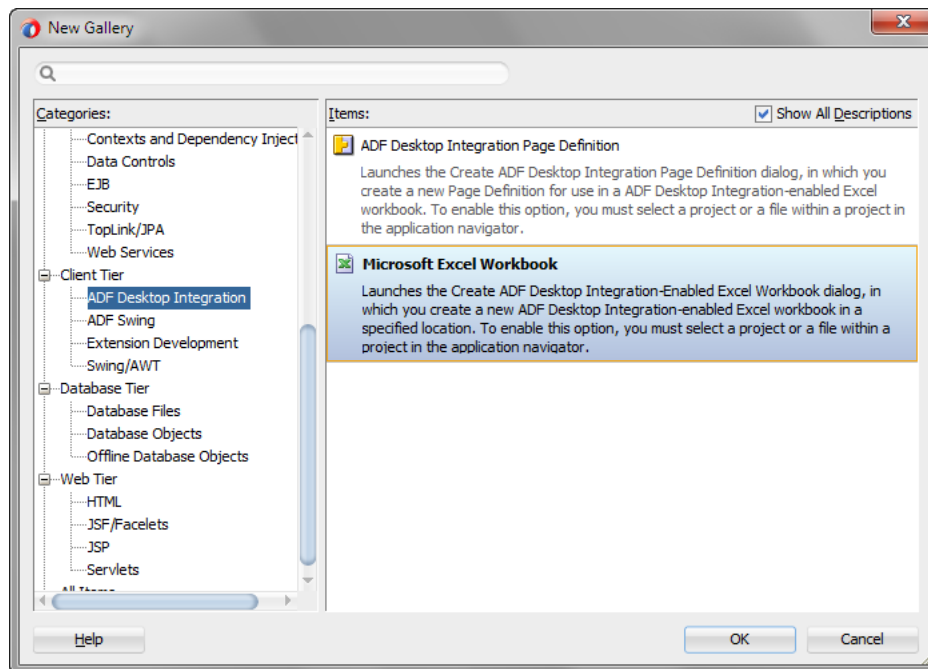
It may be helpful to have an understanding of adding ADF Desktop Integration to a Fusion web application. For more information, see [Section 4.2, "Adding an Integrated Excel Workbook to a Fusion Web Application."](#)

To add an integrated Excel workbook in JDeveloper:

1. Open the Fusion web application in JDeveloper.
2. In the Applications window, select the user interface project, such as **ViewController**, to which you want to add the new integrated Excel workbook.
3. From the **File** menu, choose **New > From Gallery**.
4. In the New Gallery, expand **Client Tier**, select **ADF Desktop Integration**, then **Microsoft Excel Workbook**, and then click **OK**.

[Figure 4–1](#) shows the New Gallery with **ADF Desktop Integration** category and the **Microsoft Excel Workbook** option.

Figure 4–1 New Gallery - Microsoft Excel Workbook



Click **OK**.

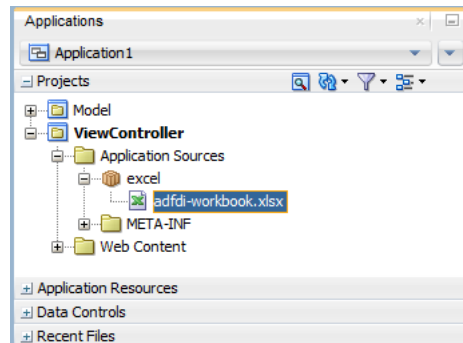
5. In the Create ADF Desktop Integration-Enabled Excel Workbook dialog, if required, edit the file name of the workbook and its location.

By default, the integrated Excel workbook is saved as `adfdi-workbook.xlsx` in the `<PROJECT_HOME>\src\excel` directory of the selected project. Although you can save the workbook anywhere you choose, you should save the workbook with the other files of the Fusion web application.

6. Click **OK**.

JDeveloper adds the integrated Excel workbook into the Fusion web application, and automatically enables the project with ADF Desktop Integration. [Figure 4–2](#) shows the **ViewController** project in the Applications window.

Figure 4–2 *adfdi-workbook.xlsx* in Applications Window



4.2.2 How to Configure a New Integrated Excel Workbook

After adding the integrated Excel workbook, you must configure it.

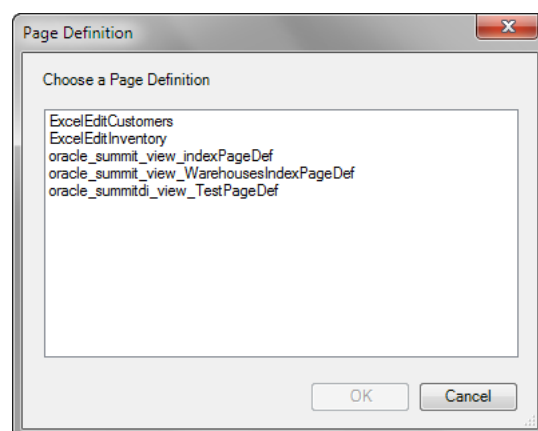
Before you begin:

It may be helpful to have an understanding of adding an integrated Excel workbook to a Fusion web application. For more information, see [Section 4.2, "Adding an Integrated Excel Workbook to a Fusion Web Application."](#)

To configure a new integrated Excel workbook:

1. Open the integrated Excel workbook.
 - If you have saved the workbook with other files of the Fusion web application, the Page Definition dialog automatically appears, as illustrated in [Figure 4–3](#).

Figure 4–3 *Page Definition Dialog*



Select the page definition file for the active worksheet from the Page Definition dialog, and click **OK**.

- If you have saved the workbook elsewhere, configure the workbook as described in [Section 4.4.3, "How to Manually Configure a New Integrated Excel Workbook."](#)

2. In the Workbook group of the **Oracle ADF** tab, click **Workbook Properties**.
3. In the Edit Workbook Properties dialog, set or verify the values for the following properties so that you can switch between design mode and test mode as you configure the workbook:

- `ApplicationHomeFolder`

The value for this property corresponds to the absolute path for the root directory of the JDeveloper application workspace (`.jws`). If the workbook is located within the JDeveloper application workspace, the value of the `ApplicationHomeFolder` workbook property is assigned automatically.

Note: If you are opening the Excel file after moving the application directory, ensure that the `ApplicationHomeFolder` property's value reflects the correct path.

- `Project`

The value for this property corresponds to the name of the JDeveloper project (`.jpr`) in the JDeveloper application workspace. To change the project, click the browse (...) icon and choose the project from the Project dialog, which lists the projects defined in the JDeveloper application workspace.

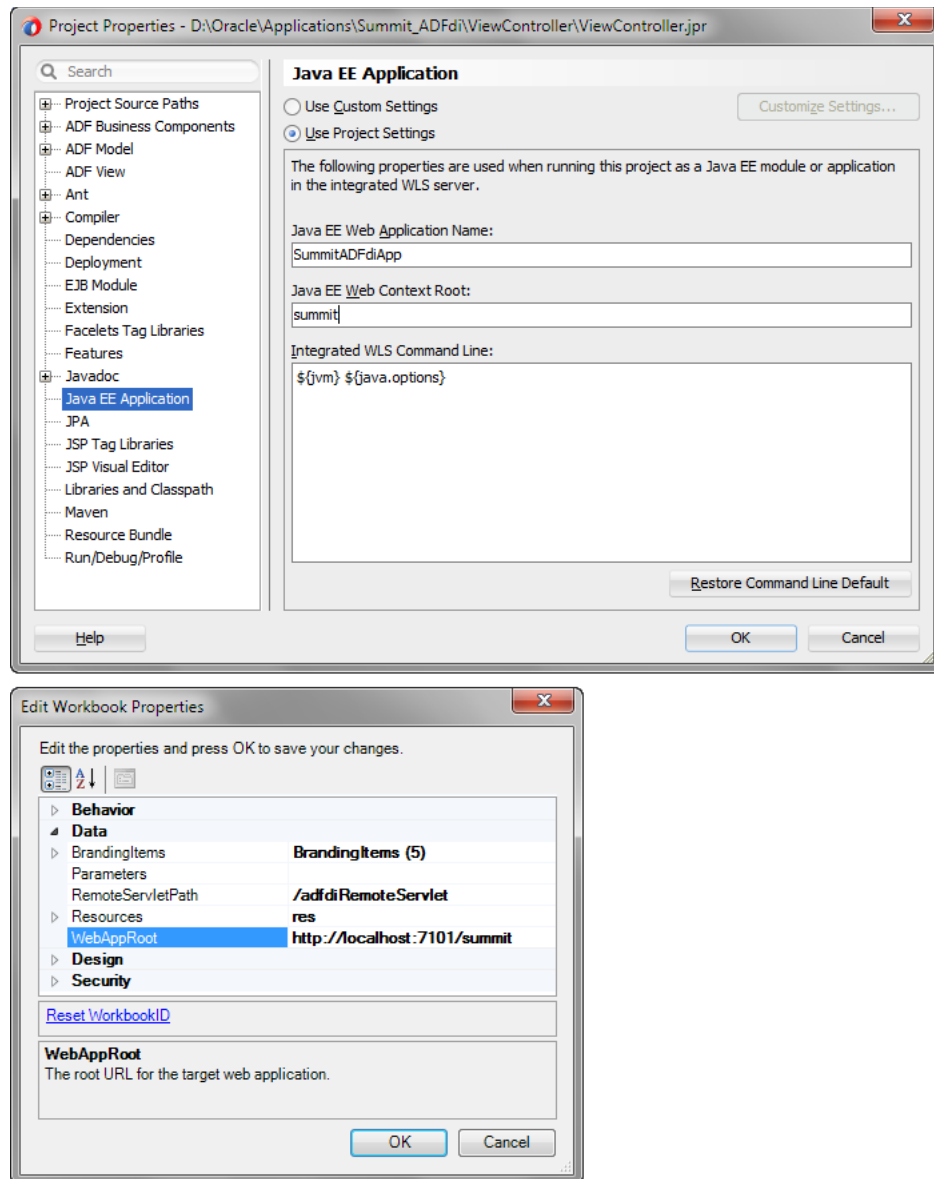
By default, `Project` is set to the name of the project that contains the Excel document. ADF Desktop Integration loads the names of the available projects from the `application_name.jws` specified as a value for `ApplicationHomeFolder`.

- `WebAppRoot`

Set the value for this property to the fully qualified URL for the web context root that you want to integrate the Fusion web application with. The fully qualified URL has the following format:

```
http://<hostname>:<portnumber>/context-root
```

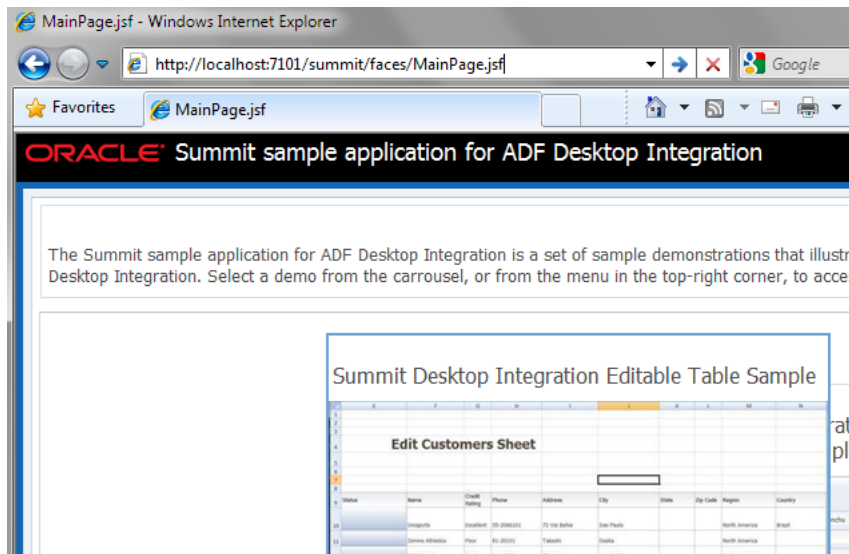
In JDeveloper, you specify the web context root (`context-root`) in the Java EE Application page of the Project Properties dialog. [Figure 4-4](#) shows the web context root used for the Summit sample application for ADF Desktop Integration in JDeveloper and integrated Excel workbook.

Figure 4–4 Setting Web Context Root in JDeveloper and Integrated Excel Workbook

Note that the fully qualified URL is similar to the following if you set up a test environment on the system using the Summit sample application for ADF Desktop Integration, as shown in [Figure 4–5](#).

`http://localhost:7101/summit`

Figure 4–5 Home Page of Summit Sample Application for ADF Desktop Integration in a Browser

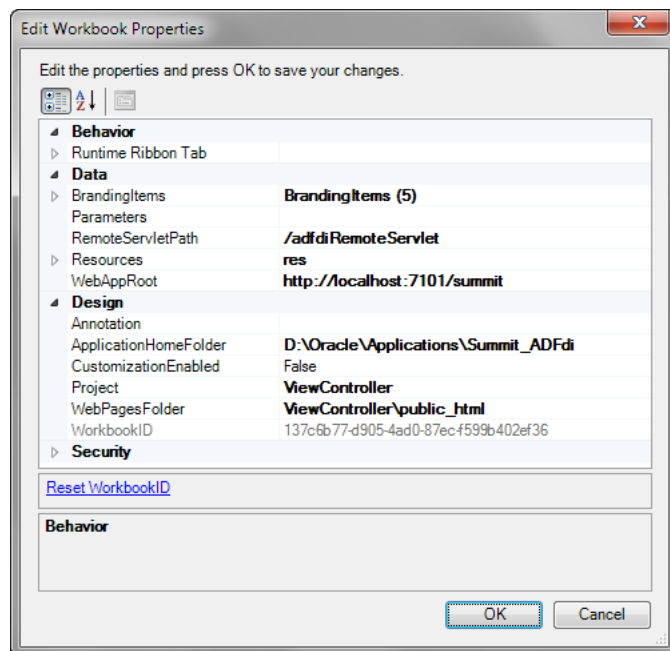


For information about how to verify that the Fusion web application is online and that it supports ADF Desktop Integration, see [Section C.1, "Verifying That Your Fusion Web Application Supports ADF Desktop Integration."](#)

If you are integrating an Excel file with a secure Fusion web application, you should use the `https` protocol while entering the value for `WebAppRoot`. For more information about securing the Fusion web application, see *Developing Applications with the WebLogic Security Service*.

- `WebPagesFolder`
Set the value for this property to the directory that contains web pages for the Fusion web application. The directory path should be relative to the value of `ApplicationHomeFolder`. For example, in the `EditCustomers-DT.xlsx` workbook, `WebPagesFolder` is set to `ViewController\public_html`.

[Figure 4–6](#) shows an example of workbook properties in the Edit Workbook Properties dialog of the Summit sample application for ADF Desktop Integration `EditCustomers-DT.xlsx` workbook.

Figure 4–6 Edit Workbook Properties Dialog

4. Click OK.

Note: In Step 1, if the fully qualified path of the selected page definition file contains more than 259 characters, a warning message will appear when the Workbook Properties dialog is closed, and the page definition will not be loaded.

5. Save the Excel workbook.

4.2.3 How to Add Additional Worksheets to an Integrated Excel Workbook

To use Oracle ADF functionality, associate each worksheet with a page definition file. You associate a page definition file with a worksheet when you add a worksheet to the integrated Excel workbook. You can integrate multiple worksheets in an integrated Excel workbook with a Fusion web application. Use a different page definition file for each worksheet in the integrated Excel workbook.

Before you begin:

It may be helpful to have an understanding of adding an integrated Excel workbook to a Fusion web application. For more information, see [Section 4.2, "Adding an Integrated Excel Workbook to a Fusion Web Application."](#)

To associate a page definition file with an Excel worksheet:

1. While the Excel workbook is in design mode, click the **Home** tab in the Excel ribbon, and then choose **Insert > Insert Sheet** in the **Cells** group.
2. In the Choose Page Definition dialog, select the page definition file.

This populates the bindings palette in the ADF Desktop Integration task pane with the bindings contained in the page definition file. You can now configure the worksheet with Oracle ADF functionality.

Note: If you get an error message **Programmatic access to Visual Basic Project is not trusted** when you run an integrated Excel workbook after inserting a new worksheet, enable the **Trust access to the VBA project object model** checkbox in Excel Options. For more information, see [Section 3.3, "Configuring Excel to work with ADF Desktop Integration."](#)

4.2.4 What Happens When You Deploy an ADF Desktop Integration-Enabled Fusion Web Application from JDeveloper

When you deploy the ADF Desktop Integration-enabled Fusion web application from JDeveloper, references to the ADF Desktop Integration shared libraries are added to the appropriate descriptor files. For any Fusion web application that contains one or more projects referencing the ADF Desktop Integration Model API library or the ADF Desktop Integration Runtime library, a platform-dependent reference to the ADF Desktop Integration Model API shared library is added during deployment.

For any web application module (WAR) project that contains a reference to the ADF Desktop Integration Runtime library, a platform-dependent reference to the ADF Desktop Integration Runtime shared library is added during deployment.

4.2.4.1 Fusion Web Application is Deployed on Oracle WebLogic Server

When you deploy the Fusion web application on Oracle WebLogic Server, the following happens:

- The META-INF/weblogic-application.xml file of the deployed application EAR file contains a library reference to oracle.adf.desktopintegration.model.

For example:

```
<library-ref>
  <library-name>oracle.adf.desktopintegration.model</library-name>
</library-ref>
```

The shared library is delivered in *MW_HOME*/oracle_common/modules/oracle.adf.desktopintegration.model_12.1.3, in the oracle.adf.desktopintegration.model.ear.

- The WEB-INF/weblogic.xml file of the deployed web application WAR file contains a library reference to oracle.adf.desktopintegration.

For example:

```
<library-ref>
  <library-name>oracle.adf.desktopintegration</library-name>
</library-ref>
```

The shared library is delivered in *MW_HOME*/oracle_common/modules/oracle.adf.desktopintegration_12.1.3, in the oracle.adf.desktopintegration.war.

4.3 Working with Page Definition Files for an Integrated Excel Workbook

Page definition files define the bindings that populate the data in the Oracle ADF components at runtime. Page definition files also reference the action bindings and method action bindings that define the operations or actions to use on this data. You must define a separate page definition file for each Excel worksheet that you are going

to integrate with a Fusion web application. The integrated Excel workbook can include worksheets that do not reference page definition files.

The ADF Desktop Integration task pane displays only those bindings that ADF Desktop Integration supports in the bindings palette. If a page definition file references a binding that ADF Desktop Integration does not support (for example, a graph binding), it is not displayed.

[Table 4–1](#) lists and describes the binding types that the ADF Desktop Integration module supports.

Table 4–1 Binding Requirements for ADF Desktop Integration Components

ADF Desktop Integration component	Supported Binding	Additional comments
ADF Input Text	Attribute binding	
ADF Output Text	Attribute binding	
ADF Label	Attribute and list bindings	This ADF Desktop Integration component uses the label property of a control binding.
ADF List of Values	List binding	
Tree Node List	Tree binding attributes and list binding	Tree binding attributes must be associated with a model-driven list.
ADF Button	Various	The ADF Button component in ADF Desktop Integration can invoke action sets. Action sets can reference action bindings, method action bindings, or actions exposed by components in ADF Desktop Integration. For more information about action sets, see Section 8.2, "Using Action Sets."
ADF Read-only Table	Tree binding	
ADF Table	Tree binding	

For information about the bindings that components in ADF Desktop Integration use, see [Appendix A, "ADF Desktop Integration Component Properties and Actions."](#)

For information about the elements and attributes in page definition files, see the "pageNamePageDef.xml" section of the *Developing Fusion Web Applications with Oracle Application Development Framework*.

For information about ADF data binding and page definition files in a Fusion web application, see the "Using ADF Model in a Fusion Web Application" chapter of the *Developing Fusion Web Applications with Oracle Application Development Framework*.

4.3.1 How to Create ADF Desktop Integration Page Definition File

You create and configure a page definition file that determines the Oracle ADF bindings to expose in the JDeveloper project.

Before you begin:

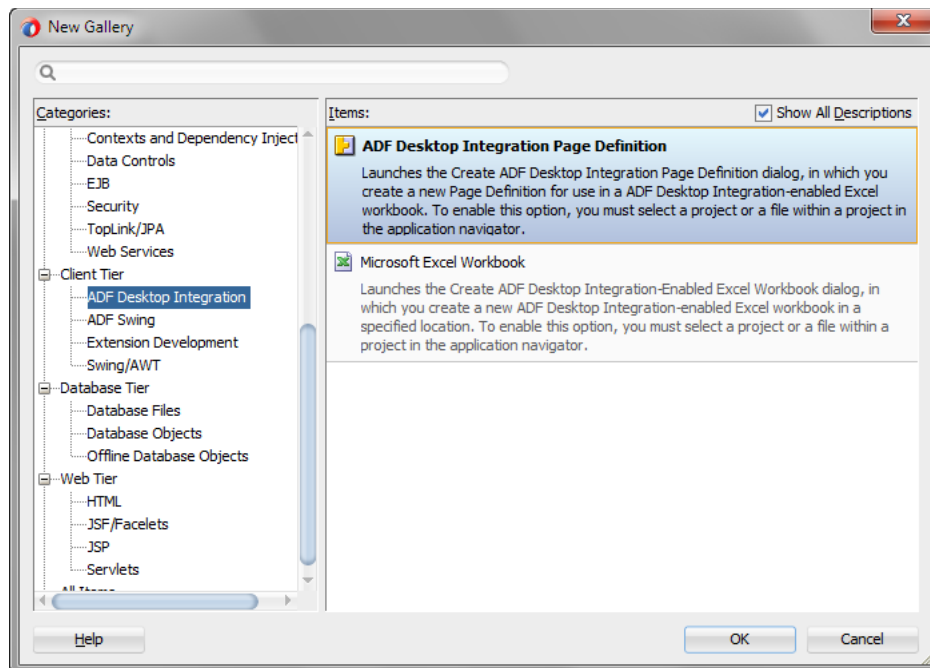
It may be helpful to have an understanding of page definition files. For more information, see [Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook."](#)

To create an ADF Desktop Integration page definition file:

1. Open the Fusion web application in JDeveloper.
2. In the Applications window, select the user interface project, such as **ViewController**, to which you want to add the page definition file.
3. From the **File** menu, choose **New > From Gallery**.
4. In the New Gallery, expand **Client Tier**, select **ADF Desktop Integration**, then **ADF Desktop Integration Page Definition**, and then click **OK**.

Figure 4–7 shows the New Gallery with **ADF Desktop Integration** category and the **ADF Desktop Integration Page Definition** option.

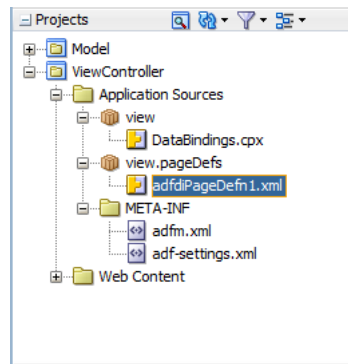
Figure 4–7 New Gallery - ADF Desktop Integration Page Definition



Click **OK**.

5. In the Create ADF Desktop Integration Page Definition dialog, if required, edit the page definition file name.
6. Click **OK**.

JDeveloper adds the page definition into the Fusion web application and opens the new page definition's editor. Figure 4–8 shows the **ViewController** project with the new page definition in the Applications window.

Figure 4–8 *adfdiPageDefn.xml* in Applications Window

For information about working with page definition files, see the "Working with Page Definition Files" section in the *Developing Fusion Web Applications with Oracle Application Development Framework*.

4.3.2 What Happens When You Create a Page Definition File

JDeveloper creates the `DataBindings.cpx` file the first time you add a page definition file in the JDeveloper project using the procedure described in [Section 4.3.1, "How to Create ADF Desktop Integration Page Definition File."](#)

The `DataBindings.cpx` file defines the binding context for the Fusion web application and provides the configuration from which the Oracle ADF bindings are created at runtime. Information about working with this file can be found in the "Working with the `DataBindings.cpx` File" section of the *Developing Fusion Web Applications with Oracle Application Development Framework*. Information about the elements and attributes in the file can be found in the "DataBindings.cpx" section of the same guide.

4.3.3 How to Reload a Page Definition File in an Excel Workbook

If you make changes in your JDeveloper desktop integration project to a page definition file that is associated with an Excel worksheet, rebuild the JDeveloper desktop integration project and reload the page definition file in the Excel worksheet to ensure that the changes appear in the ADF Desktop Integration task pane. You associate a page definition file with an Excel worksheet when you choose the page definition file, as described in [Section 4.2.2, "How to Configure a New Integrated Excel Workbook."](#)

The **Oracle ADF** tab provides a button that reloads all page definition files in an Excel workbook.

Errors may occur when you switch an integrated Excel workbook from design mode to runtime if you do not rebuild the JDeveloper desktop integration project and restart the application after making changes to a page definition file. For example, if you:

- Remove an element in a page definition file
- Do not rebuild and restart the Fusion web application
- Or do not reload the page definition file in the integrated Excel workbook

an error message such as the following may appear when you attempt to switch a workbook to test mode:

```
[ADFDI-05530] unable to initialize worksheet: MyWorksheet
[ADFDI-05517] unable to find control MyBindingThatWasRemoved
```

Before you begin:

It may be helpful to have an understanding of page definition files. For more information, see [Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook."](#)

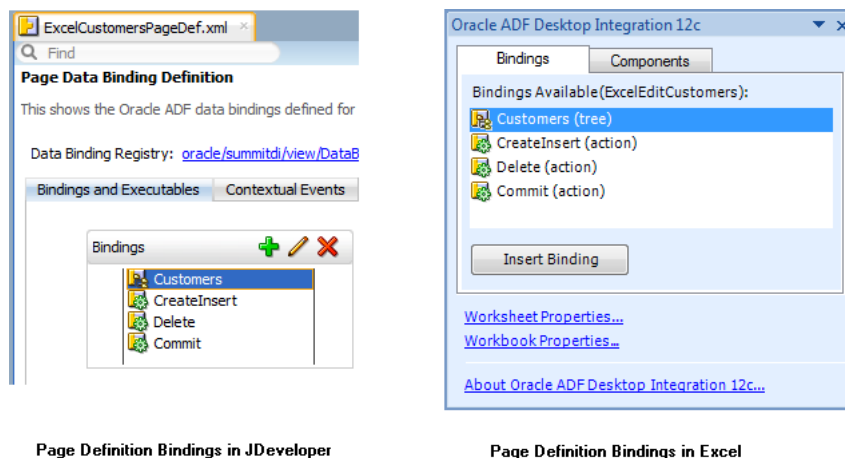
To reload page definition files in an Excel workbook:

1. Ensure that you have saved the updated page definition file in JDeveloper.
2. In the Excel workbook, click the **Refresh Bindings** button in the Components group of the **Oracle ADF** tab.

For information about the Refresh Bindings button, see [Section 5.1, "About Development Tools."](#)

After reloading the page definition file, the ADF Desktop Integration task pane of the worksheet displays the same bindings that are available in its associated page of the Fusion web application. For example, [Figure 4–9](#) shows the bindings in the `ExcelCustomersPageDef.xml` page definition file and the same bindings in the worksheet of the `EditCustomers-DT.xlsx` workbook.

Figure 4–9 Page Definition Bindings in JDeveloper and Integrated Excel Workbook



Page Definition Bindings in JDeveloper

Page Definition Bindings in Excel

4.3.4 What You May Need to Know About Page Definition Files in an Integrated Excel Workbook

Note the following points about page definition files in an ADF Desktop Integration project:

- **Integrating Multiple Excel Worksheets:** You can integrate multiple worksheets in an Excel workbook with a Fusion web application. You associate a page definition file with each worksheet as described in [Section 4.2.3, "How to Add Additional Worksheets to an Integrated Excel Workbook."](#)
- **EL Expressions in a Page Definition File:** Use the following syntax to write EL expressions in a page definition file:

```
Dynamic (${})
```

Do not use the syntax `Deferred (#{})` to write EL expressions. EL expressions using this syntax generate errors because they attempt to access the ADF Faces context, which is not available.

Note: EL expressions that you write for ADF Desktop Integration component in the integrated Excel workbook, such as the Input Text component, must use the Deferred (`#{}`) syntax.

4.4 Enabling ADF Desktop Integration Manually

To enable ADF Desktop Integration in the Fusion web application without adding the integrated Excel workbook, you must add the ADF Desktop Integration feature manually.

4.4.1 How to Manually Add ADF Desktop Integration In Fusion Web Application

Use the Project Properties dialog in JDeveloper to add ADF Desktop Integration to the feature list of your project.

Before you begin:

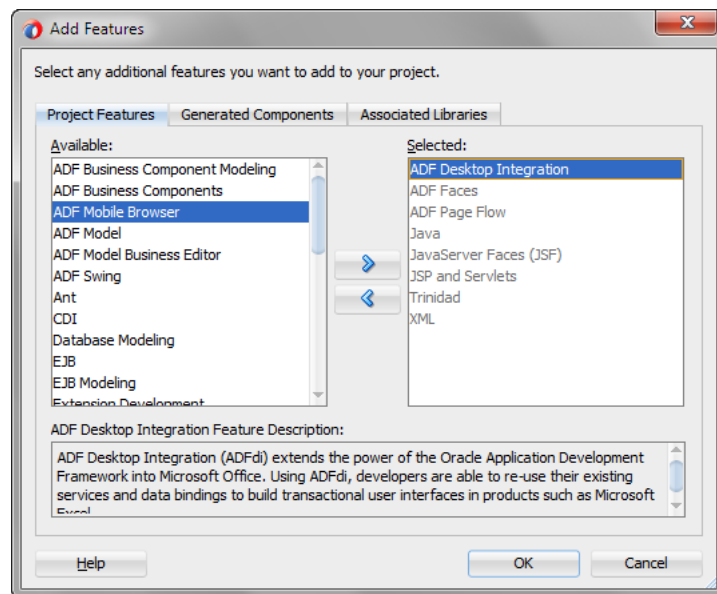
It may be helpful to have an understanding of adding ADF Desktop Integration to a Fusion web application. For more information, see [Section 4.4, "Enabling ADF Desktop Integration Manually."](#)

To manually add ADF Desktop Integration to your project:

1. Open the project in JDeveloper.
2. In the Applications window, right-click the project to which you want to add ADF Desktop Integration and choose **Project Properties**.

If the application uses the Fusion Web Application (ADF) application template, select the user interface project, such as **ViewController**. If the application uses another application template, select the project that corresponds to the web application.

3. In the Project Properties dialog, select **Features** to view the list of available features.
4. Click **Add Features**.
5. In the Add Features dialog, select the **ADF Desktop Integration** feature and add it to the **Selected** list, as shown in [Figure 4-10](#).

Figure 4–10 Add Features Dialog

6. Click **OK** to close the Add Features dialog.
7. Click **OK** to close the Project Properties dialog.

For more information about what happens when you add ADF Desktop Integration, see [Section 4.4.4, "What Happens When You Add ADF Desktop Integration to Your JDeveloper Project."](#)

Note: If you plan to distribute integrated Excel workbooks by adding them to ADF library files through EAR and JAR files, add **ADF Library Web Application Support** to your project. For more information, see [Section 4.4.5, "Adding ADF Library Web Application Support."](#)

4.4.2 How to Enable ADF Desktop Integration in an Existing Workbook

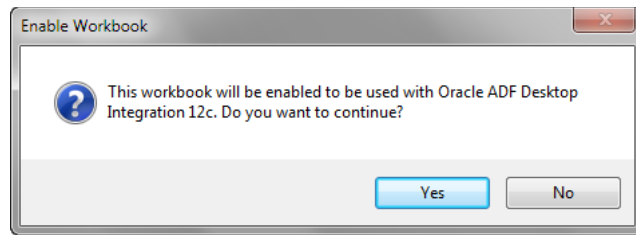
To integrate an existing workbook with the ADF Desktop Integration enabled Fusion web application, you must manually enable ADF Desktop Integration for the workbook. For information about the file formats of Excel workbooks that you can convert for integration with a Fusion web application, see [Section 3.2, "Required Oracle ADF Modules and Third-Party Software."](#)

Before you begin:

It may be helpful to have an understanding of adding integrated Excel workbook to a Fusion web application. For more information, see [Section 4.4, "Enabling ADF Desktop Integration Manually."](#)

To enable ADF Desktop Integration in an existing Excel workbook:

1. In Excel, open the workbook.
2. In the Workbook group of the Oracle ADF tab, click **Workbook Properties**.
3. In the Enable Workbook dialog, click **Yes**, as shown in [Figure 4–11](#).

Figure 4–11 Enable Workbook Dialog

ADF Desktop Integration prepares your workbook, displays the ADF Desktop Integration Designer task pane, and opens the Browse For Folder dialog. For more information, see [Section 4.4.3, "How to Manually Configure a New Integrated Excel Workbook."](#)

4. Save the workbook.

Although you can store the Excel workbooks that you integrate with Fusion web applications anywhere you choose, there are several advantages to storing them with the other files of the Fusion web application. Some of these advantages are:

- Source control of the workbooks
- Facilitating the download of workbooks from web pages
- The file system folder picker that appears the first time a workbook is opened defaults to the location where you store the workbook

For example, the Summit sample application for ADF Desktop Integration stores the Excel workbooks it integrates in the following subdirectory:

```
Summit_HOME\ViewController\src\oracle\summitdi\excel
```

where *Summit_HOME* is the root directory that stores the source files for the Summit sample application for ADF Desktop Integration.

4.4.3 How to Manually Configure a New Integrated Excel Workbook

After enabling ADF Desktop Integration manually in a workbook, you would need to configure it.

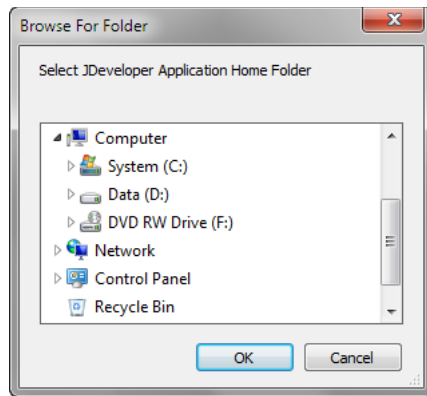
Before you begin:

It may be helpful to have an understanding of adding an integrated Excel workbook to a Fusion web application. For more information, see [Section 4.4, "Enabling ADF Desktop Integration Manually."](#)

To manually configure a new integrated Excel workbook:

1. Open the integrated Excel workbook.

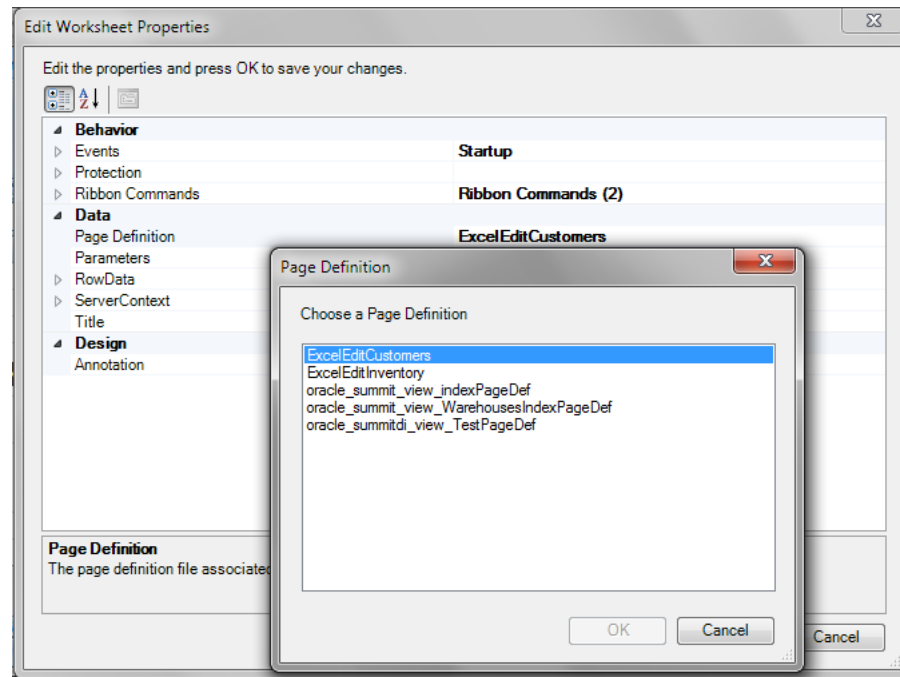
The Browse For Folder dialog automatically appears, as illustrated in [Figure 4–12](#).

Figure 4–12 Browse For Folder Dialog

Use the Browse for Folder dialog to select the JDeveloper application home directory. In a typical JDeveloper project, the JDeveloper application home directory stores the *application_name.jws* file. The value you select is assigned to the `ApplicationHomeFolder` workbook property.

Note: The Browse for Folder dialog does not appear if the workbook is located within the JDeveloper application workspace. In such a case, the value of the `ApplicationHomeFolder` workbook property is assigned automatically.

2. In the Workbook group of the **Oracle ADF** tab, click **Workbook Properties**.
3. In the Edit Workbook Properties dialog, configure the properties as described in Step 3 of [Section 4.2.2, "How to Configure a New Integrated Excel Workbook."](#)
4. Click **OK**.
5. In the Workbook group of the **Oracle ADF** tab, click **Worksheet Properties**.
6. In the Edit Worksheet Properties dialog, click the browse (...) icon beside the **Page Definition** input field and select a page definition file from the Page Definition dialog, as shown in [Figure 4–13](#).

Figure 4–13 Page Definition Dialog

7. Click **OK**.

The Excel worksheet appears with ADF Desktop Integration in the task pane. The bindings of the page definition file that you selected in Step 6, appear in the **Bindings** tab.

8. Save the Excel workbook.

By default, when you prepare a new Excel workbook with ADF Desktop Integration, it is assumed that the workbook will be integrated with an non-secured Fusion web application. If you want to integrate a workbook with a secure Fusion web application, see [Chapter 11, "Securing Your Integrated Excel Workbook."](#)

4.4.4 What Happens When You Add ADF Desktop Integration to Your JDeveloper Project

When you add the ADF Desktop Integration feature to a project, the following events occur:

- The project adds the ADF Desktop Integration runtime library. This library references the following .jar files in its class path:
 - adf-desktop-integration.jar
 - adf-desktop-integration-model-api.jar
 - resourcebundle.jar
- The project's deployment descriptor (web.xml) is modified to include the following entries:
 - An ADF bindings filter (adfBindings)
 - A servlet named adfdiRemote

Note: The value for the `url-pattern` attribute of the `servlet-mapping` element for `adfdiRemote` must match the value of the `RemoteServletPath` workbook property described in [Table A-21](#).

- A filter named `adfdiExcelDownload`
- A MIME mapping for Excel files (`.xlsx` and `.xlsm`)

The previous list is not exhaustive. Adding ADF Desktop Integration to a project makes other changes to `web.xml`. Note that some entries in `web.xml` are added only if they do not already appear in the file.

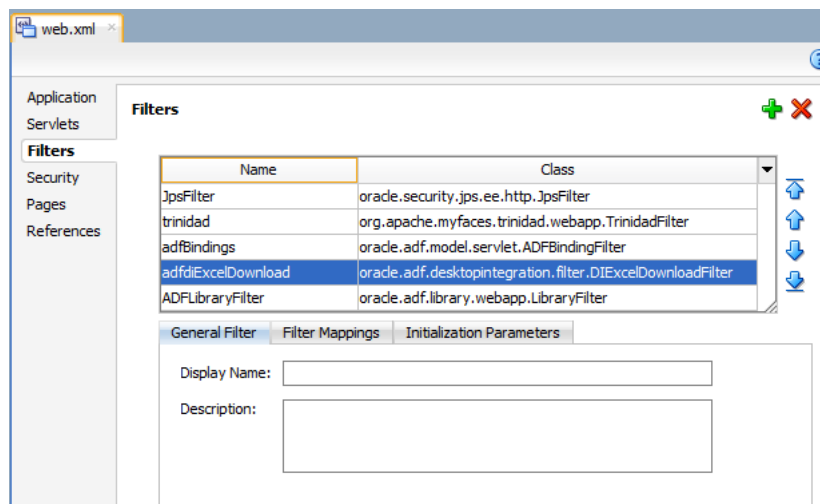
4.4.5 Adding ADF Library Web Application Support

If you want to distribute integrated workbooks by adding them to ADF library files, add ADF Library web application support to the Fusion web application. For more information, see the "Packaging a Reusable ADF Component into an ADF Library" section in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

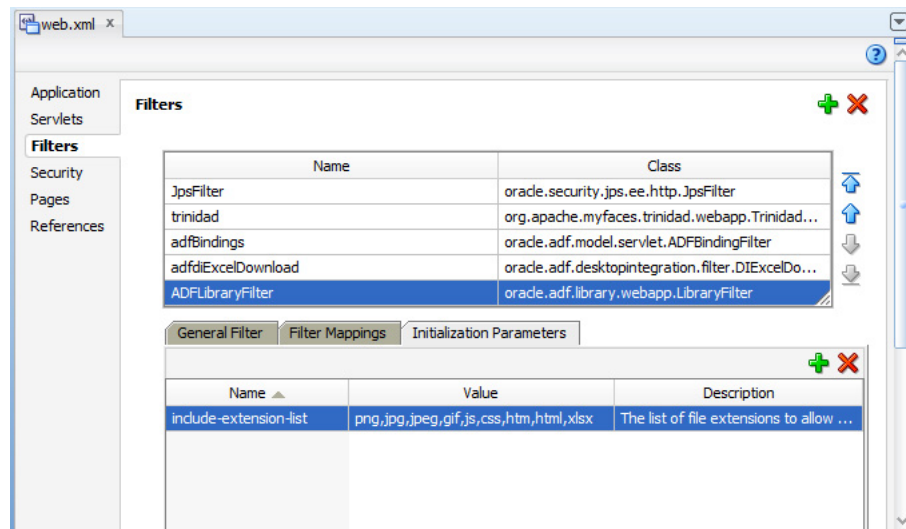
When updating filter and filter mapping information in the `web.xml` file, ensure that the filter for ADF Library Web Application Support (`<filter-name>ADFLibraryFilter</filter-name>`) appears below the `adfdiExcelDownload` filter entries, so that integrated Excel workbooks can be downloaded from the Fusion web application.

[Figure 4-14](#) shows the **Filters** tab of the overview editor of the `web.xml` in JDeveloper.

Figure 4-14 Filters Tab of `web.xml`



You should also update the `include-extension-list` initialization parameter to add the Excel file extensions (such as `.xlsx` and `.xlsm`), as shown in [Figure 4-15](#).

Figure 4–15 ADFLibraryFilter Using `include-extension-list` Parameter

For more information about `web.xml`, see [Appendix D, "ADF Desktop Integration Settings in the Web Application Deployment Descriptor."](#)

Getting Started with the Development Tools

This chapter describes how to use the development tools (such as Bindings Palette, Components Palette, Property Inspector, and Expression Builder) provided by ADF Desktop Integration. It provides an overview of the development environment that ADF Desktop Integration exposes in the Excel Ribbon.

This chapter includes the following sections:

- [Section 5.1, "About Development Tools"](#)
- [Section 5.2, "Designer Ribbon Tab"](#)
- [Section 5.3, "ADF Desktop Integration Designer Task Pane"](#)
- [Section 5.4, "Using the Bindings Palette"](#)
- [Section 5.5, "Using the Components Palette"](#)
- [Section 5.6, "Using the Property Inspector"](#)
- [Section 5.7, "Using the Binding ID Picker"](#)
- [Section 5.8, "Using the Expression Builder"](#)
- [Section 5.9, "Using the Web Page Picker"](#)
- [Section 5.10, "Using the File System Folder Picker"](#)
- [Section 5.11, "Using the Page Definition Picker"](#)
- [Section 5.12, "Using the Collection Editors"](#)
- [Section 5.13, "Using the Cell Context Menu"](#)
- [Section 5.14, "Removing ADF Desktop Integration Components"](#)
- [Section 5.15, "Exporting and Importing Excel Workbook Integration Metadata"](#)

5.1 About Development Tools

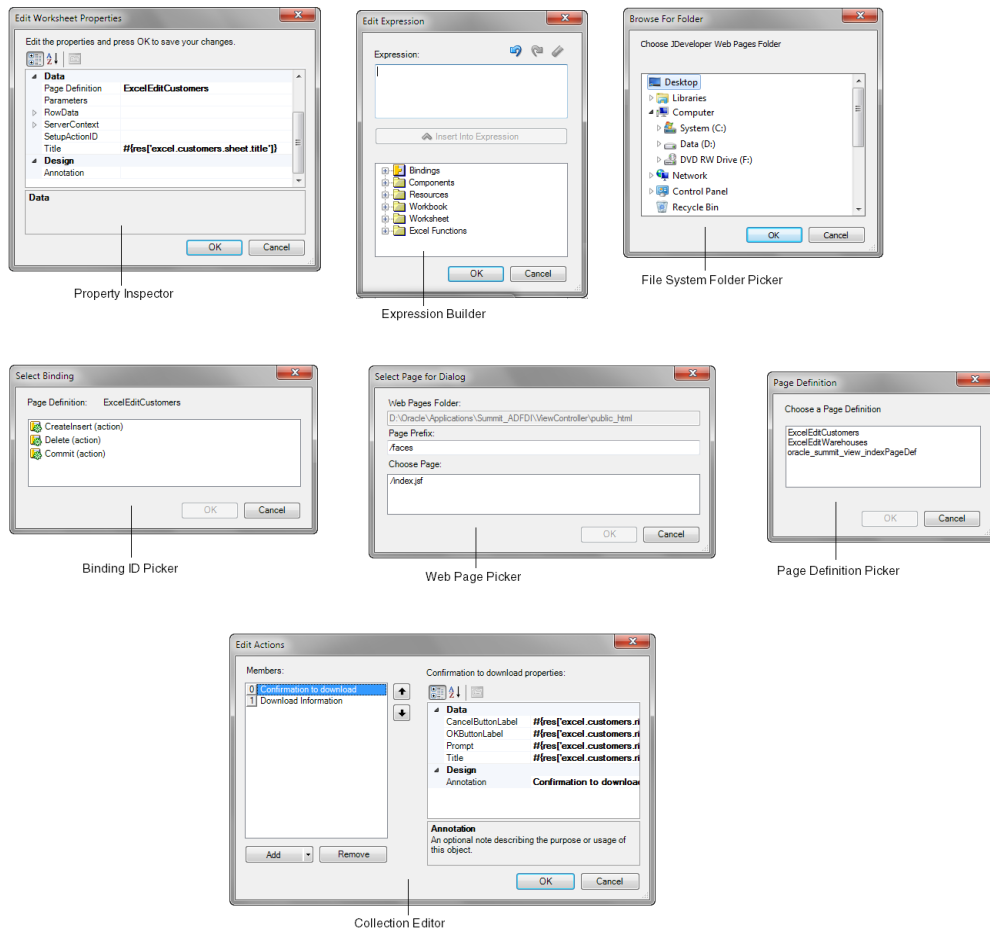
ADF Desktop Integration provides several tools to configure Excel workbooks so that they can access Oracle ADF functionality. Using these tools you configure the workbook and corresponding worksheets to display, and edit, data from the Fusion web application in the integrated Excel workbook. The tools are available in the **Oracle ADF** tab and in the ADF Desktop Integration Designer task pane.

ADF Desktop Integration development tools include the following tools, also shown in [Figure 5-1](#):

- Bindings Palette
- Components Palette

- Property Inspector
- Binding ID Picker
- Expression Builder
- Web Page Picker
- File System Folder Picker
- Page Definition Picker
- Collection Editors

Figure 5–1 ADF Desktop Integration Development Tools



ADF Desktop Integration provides two modes, design mode and the test mode, in which you can work while you configure the Excel workbook.

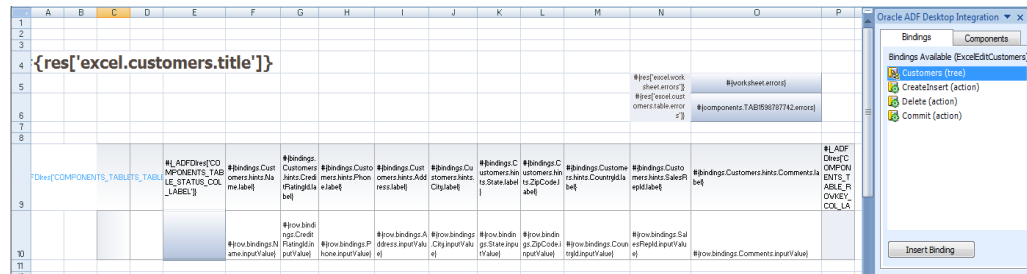
In *design mode*, you use the tools provided by Oracle ADF in Excel to design and configure the integrated Excel workbook. In *test mode*, you can view and test the changes you made in the design mode, in the same way that the end user views the published integrated Excel workbook.

5.1.1 ADF Desktop Integration Development Tools Use Cases and Examples

You use the development tools to configure and design the integrated Excel workbook. For example, as shown in [Figure 5–2](#), in `EditCustomers-DT.xlsx` a bound ADF Table

component is inserted in the integrated Excel workbook using the Customers binding from the Bindings palette.

Figure 5–2 ADF Desktop Integration Components and Bindings



Other ADF Desktop Integration components, such as ADF Button and ADF Label, are inserted from the Components palette, and configured using the Property Inspector and Expression Builder.

5.1.2 Additional Functionality for ADF Desktop Integration Development Tools

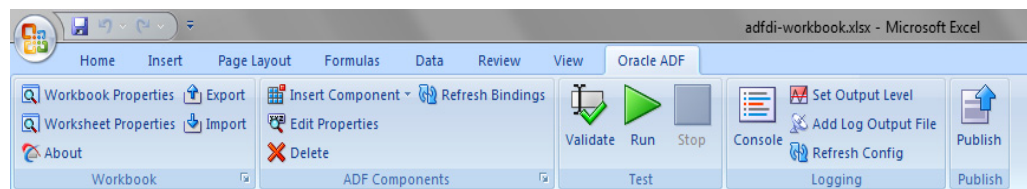
After adding the desired components and configuring your work, you may find that you need additional functionality such as changing the appearance of the workbook, and localizing it. Following are links to other functionality that you can use:

- **Localization:** You can customize the integrated Excel workbook as part of the process to internationalize and localize with the Fusion web application. For more information, see [Chapter 10, "Internationalizing Your Integrated Excel Workbook."](#)
- **Styles:** You can configure the display of your components using several predefined Excel styles. For more information, see [Section 9.2, "Working with Styles."](#)
- **EL Expressions:** You can use EL expressions with the ADF Desktop Integration components. For more information, see [Appendix B, "ADF Desktop Integration EL Expressions."](#)

5.2 Designer Ribbon Tab

You use the **Oracle ADF** tab, also called as Designer Ribbon tab, for various tasks such as configuring the integrated workbook and worksheets properties, insert Oracle ADF components and edit their properties, run the workbook in test mode, and publish the workbook. The **Oracle ADF** tab, also shown in [Figure 5–3](#), provides various buttons in design mode.

Figure 5–3 Oracle ADF Tab in Design Mode



Tip: To access **Oracle ADF** tab from the keyboard, press Alt+C. Press the Alt key again to view the shortcut keys for Oracle ADF tab command buttons.

You can use **Oracle ADF** tab buttons to invoke the actions described in [Table 5-1](#).

Table 5-1 Oracle ADF Tab Options

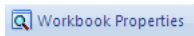
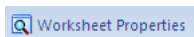

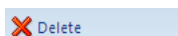
In this group...	Click this button...	To...	Mode when the button is available...
Workbook		Display the Edit Workbook Properties dialog to view and edit integrated Excel workbook properties. The button is also used to enable ADF Desktop Integration in a non-integrated Excel workbook.	Design
Workbook		Display the Edit Worksheet Properties dialog to view and edit the current worksheet properties.	Design
Workbook		Open the About ADF Desktop Integration dialog that provides version and property information of integrated Excel workbook. The button is also available in non-integrated Excel workbooks after ADF Desktop Integration is installed.	Design, Test
Workbook		Open the Save Workbook Definition as dialog that exports the current workbook definition as .xml file.	Design
Workbook		Open the Choose Workbook Definition File to Import dialog that imports the workbook integration metadata from the saved .xml file.	Design
ADF Components		Display a dropdown list of Oracle ADF components that you can insert in the selected cell.	Design
ADF Components		Display the property inspector window to view and edit component properties of the selected component.	Design
ADF Components		Delete the selected component from the Excel worksheet.	Design

Table 5–1 (Cont.) Oracle ADF Tab Options

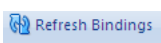




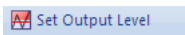
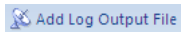
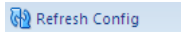

In this group...	Click this button...	To...	Mode when the button is available...
ADF Components		<ul style="list-style-type: none"> Reload the application workspace file (.jws) and project file (.jpr) referenced by the workbook properties of the integrated Excel workbook. Refresh all information from the page definition files used in the active integrated Excel workbook. <p>Any modifications that you made to the page definition files in the JDeveloper project now become available in the Excel workbook. For more information, see Section 4.3.3, "How to Reload a Page Definition File in an Excel Workbook."</p>	Design
Test		<p>Validate the Excel workbook configuration against ADF Desktop Integration validation rules.</p> <p>For more information about validating a workbook, see Section 13.3, "Validating the Integrated Excel Workbook Configuration."</p>	Design
Test		Switch the Excel workbook from design mode to test mode. This button is active only when you are in design mode.	Design
Test		<p>Switch the Excel workbook from test mode to design mode. This button is active only when you are in test mode.</p> <p>For more information about switching between design mode and test mode, see Section 13.4, "Testing Your Integrated Excel Workbook."</p>	Test
Logging		Display a dialog to review the client-side log entries. For more information, see Section C.3.2, "About Client-Side Logging."	Design, Test
Logging		Display the Set Output Level dialog to choose client-side log output level. For more information, see Section C.3.2, "About Client-Side Logging."	Design, Test

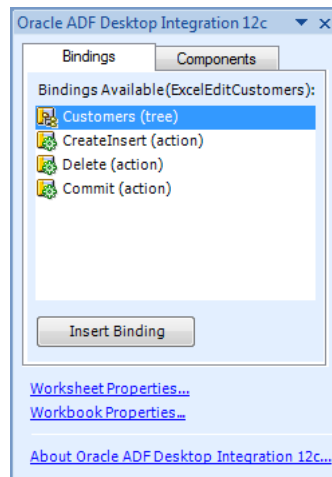
Table 5–1 (Cont.) Oracle ADF Tab Options

In this group...	Click this button...	To...	Mode when the button is available...
Logging		Create a new temporary logging listener to act as a client-side log output file. For more information, see Section C.3.2, "About Client-Side Logging."	Design, Test
Logging		Reload the ADF Desktop Integration configuration file. For more information, see Section C.3.2, "About Client-Side Logging."	Design, Test
Publish		Publish the Excel workbook after you complete the integration between the Excel workbook and the Fusion web application. For more information about publishing an integrated Excel workbook, see Chapter 14, "Deploying Your Integrated Excel Workbook."	Design

Tip: For quick and easy access, you can add **Oracle ADF** tab buttons to the Excel Quick Access toolbar.

5.3 ADF Desktop Integration Designer Task Pane

The development tools in ADF Desktop Integration Designer Task Pane are organized in two palettes, the Bindings palette and the Component palette. You use the Bindings palette of ADF Desktop Integration Designer task pane to insert a predefined binding into the integrated Excel workbook. ADF Desktop Integration inserts an Oracle ADF component that references the binding you selected, and prepopulates the properties of the Oracle ADF component with appropriate values. Similarly, you use the Components palette to insert an Oracle ADF component in the integrated Excel workbook. [Figure 5–4](#) displays the ADF Desktop Integration Designer task pane.

Figure 5–4 ADF Desktop Integration Designer Task Pane

You invoke the ADF Desktop Integration Designer task pane through launcher buttons (highlighted by the red boxes in [Figure 5–5](#)) available in the bottom-right corner of the Workbook and ADF Components group on the **Oracle ADF** tab.

Figure 5–5 ADF Desktop Integration Designer Task Pane Launcher Buttons

[Table 5–2](#) lists the view tabs and links that appear in the task pane, provides a brief description of each item.

Table 5–2 Overview of ADF Desktop Integration Designer Task Pane

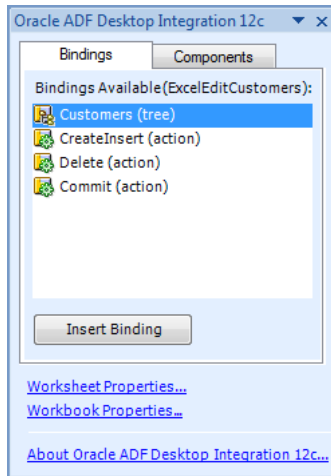
Task Pane UI Element	Description
Workbook Properties	Click to display the Edit Workbook Properties dialog. This dialog enables you to view and edit properties that affect the whole workbook. Examples include properties that reference the directory paths to page definition files, the URL for your Fusion web application, and so on.
Worksheet Properties	Click to display the Edit Worksheet Properties dialog. This dialog enables you to view and edit properties specific to the active worksheet. An example is the file name of the page definition file that you associate with the worksheet.
About	Click to display the About dialog. This dialog provides the version and property information that can be useful when troubleshooting an integrated Excel workbook. For example, it provides information about the underlying Microsoft .NET and Oracle ADF frameworks that support an integrated Excel workbook.

5.4 Using the Bindings Palette

The bindings palette presents the available Oracle ADF bindings that you can insert into the Excel worksheet. The page definition file for the current Excel worksheet determines what Oracle ADF bindings appear in the bindings palette. [Figure 5–6](#) shows a bindings palette populated with Oracle ADF bindings in the ADF Desktop

Integration Designer task pane. Note that the bindings palette does not display bindings that an integrated Excel workbook cannot use, so the bindings that appear may differ from those that appear in the page definition file viewed in JDeveloper. Check the log for ignored bindings (see [Section C.3, "Generating Log Files for an Integrated Excel Workbook"](#)).

Figure 5–6 Oracle ADF Bindings Palette in the ADF Desktop Integration Designer Task Pane



You use the bindings palette in design mode to insert a binding. When you attempt to insert a binding, ADF Desktop Integration inserts an Oracle ADF component that references the binding you selected. ADF Desktop Integration also prepopulates the properties of the Oracle ADF component with appropriate values. For example, if you insert a binding, such as the **Commit (action)** binding illustrated in [Figure 5–6](#), the property inspector for an Oracle ADF Button component appears. This Oracle ADF Button component has values specified for its `ClickActionSet` that include invoking the `Commit` action binding.

To insert an Oracle ADF binding, select the cell to anchor the Oracle ADF component that is going to reference the binding in the Excel worksheet, and then insert the binding in one of the following ways:

- Double-click the Oracle ADF control binding you want to insert.
- Select the binding that you want to insert, and drag it to the desired cell.
- Select the control binding and click **Insert Binding** in the ADF Desktop Integration Designer task pane.

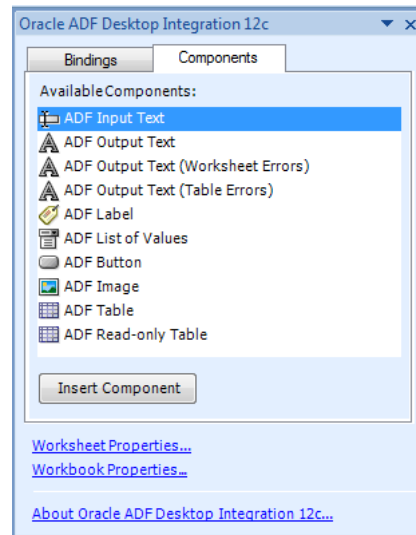
A property inspector for the Oracle ADF component that is associated with the binding you attempt to insert appears. In some instances, you may be prompted to select one Oracle ADF component from a list of Oracle ADF components where multiple Oracle ADF components can be associated with the binding. After you select an Oracle ADF component from the list, a property inspector appears.

If you choose the Oracle ADF component as ADF Input Text, ADF Output Text, or ADF Label, the binding name is assigned to the **Value** property. If you choose the Oracle ADF component as ADF Button or ADF Ribbon Command, the binding name is assigned to the **Label** property. If you choose the Oracle ADF component as ADF Table or ADF Read-only Table, the binding name is assigned to the **TreeID** property.

5.5 Using the Components Palette

The components palette displays the available ADF Desktop Integration components that you can insert into an Excel worksheet. [Figure 5–7](#) shows the components palette as it appears in the ADF Desktop Integration Designer task pane.

Figure 5–7 Oracle ADF Components Palette in the ADF Desktop Integration Designer Task Pane



You use the components palette in design mode to insert an Oracle ADF component. First, select the cell to anchor the Oracle ADF component in the Excel worksheet, and then insert the Oracle ADF component in one of the following ways:

- Double-click the Oracle ADF component you want to insert.
- Select the component that you want to insert, and drag it to the desired cell.
- Select the component and click **Insert Component** in the ADF Desktop Integration Designer task pane.

In all of the above cases, the Oracle ADF component's property inspector appears. Use the property inspector to specify values for the component before you complete its insertion into the Excel worksheet.

Note: The ADF Desktop Integration components are also available in the **Insert Component** dropdown list of **Oracle ADF** tab.

5.6 Using the Property Inspector

The property inspector is a dialog that enables you to view and edit the properties of Oracle ADF bindings, Oracle ADF components, Excel worksheets, or the Excel workbook. You can open the property inspector in one of the following ways:

- Select the component or binding, and click the **Edit Properties** icon in the **Oracle ADF** tab.
- Select the component or binding, right-click and choose **Edit ADF Component Properties**.
- Double-click the component or binding.

To open property inspector of ADF Table or ADF Read-only Table, double-click any cell that is part of the table.

Note: ADF Button does not support the right-click or double-click action, click the button to open the property inspector dialog.



The property inspector also appears automatically after you insert an Oracle ADF binding or component into an Excel worksheet. [Figure 5–8](#) shows a property inspector where you can view and edit the properties of an Oracle ADF Button component.

At design time, you can edit key properties of certain Oracle ADF components by editing the Excel cell where the component appears. For example, you can edit the Value property of ADF Label and ADF Input Text components by editing the value displayed in the cell.

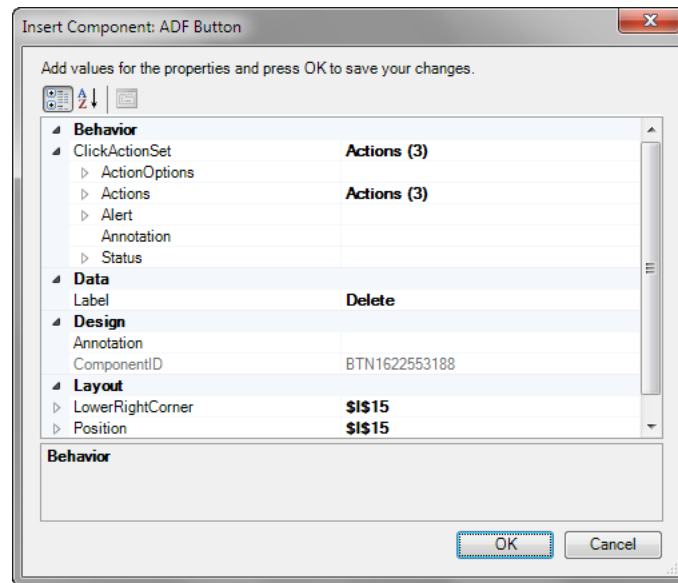
Note: The property inspector does not validate the values you enter for a property, or combinations of properties. Invalid values may cause runtime errors. To avoid runtime errors, make sure you specify valid values for properties in the property inspector.

You can display the properties in an alphabetical list or in a list where the properties are grouped by categories such as Behavior, Data, and so on. [Table 5–3](#) describes the buttons that you can use to change how properties display in the property inspector.

Table 5–3 Buttons to Configure Properties Display in Property Inspector

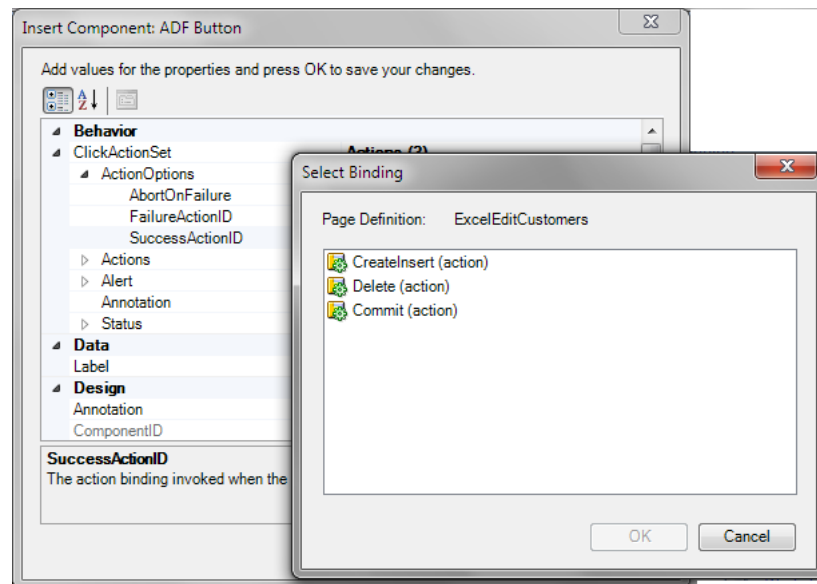
Button	Description
	Use this button to display the properties according to category.
	Use this button to display the properties in an alphabetical list.

In [Figure 5–8](#), the property inspector displays the properties grouped by category.

Figure 5–8 Property Inspector Window for ADF Button Component

5.7 Using the Binding ID Picker

The binding ID picker is a dialog that enables you to select Oracle ADF bindings at design time to configure the behavior of Oracle ADF components at runtime. You invoke the binding ID picker from the property inspector. The binding ID picker filters the Oracle ADF bindings that appear, based on the type of binding that the Oracle ADF component property accepts. For example, the `SuccessActionID` property for an ADF Button component supports only action bindings. Therefore, the binding ID picker filters the bindings from the page definition file so that only action bindings appear, as illustrated in [Figure 5–9](#).

Figure 5–9 Binding ID Picker

For more information about ADF Desktop Integration component properties and the bindings they support, see [Appendix A, "ADF Desktop Integration Component"](#)

Properties and Actions."

5.8 Using the Expression Builder

You use the expression builder to write Expression Language, or EL, expressions that configure the behavior of components at runtime in the Excel workbook. You invoke the expression builder from the property inspector of component properties that support EL expressions. For example, the `Label` property in [Figure 5–10](#) supports EL expressions and, as a result, you can invoke the expression builder to set a value for this property.

You can reference bindings in the EL expressions that you write. Note that the expression builder does not filter bindings. It displays all bindings that the page definition file exposes. See [Table 4–1](#) to identify the types of bindings that each ADF Desktop Integration component supports.

To add an expression in the **Expression** box, select the item and click **Insert Into Expression**. You can also double-click the item to add it in the **Expression** box. [Table 5–4](#) describes the folders available in the expression builder.

Figure 5–10 *Expression Builder*

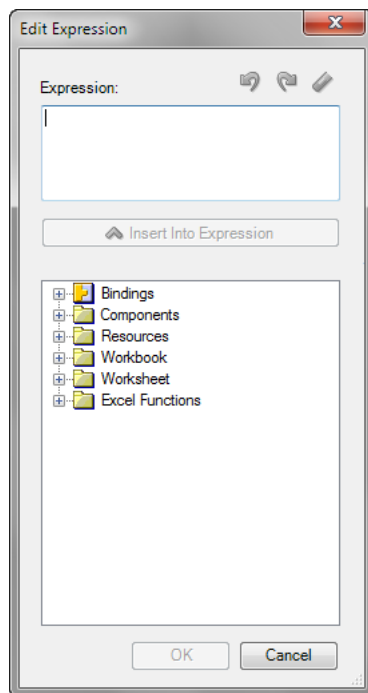


Table 5–4 *Expression Builder Folders*

Folder Name	Description
Bindings	Lists the bindings supported in ADF Desktop Integration from the current worksheet's page definition.
Components	Lists the ADF components available in the current worksheet.
Resources	Lists the resource bundles registered in <code>Workbook.Resources</code> along with the built-in resource bundle <code>_ADFDIres</code> .
Styles	Lists all Excel styles defined in the current workbook. For more information, see Section 9.2, "Working with Styles."

Table 5–4 (Cont.) Expression Builder Folders

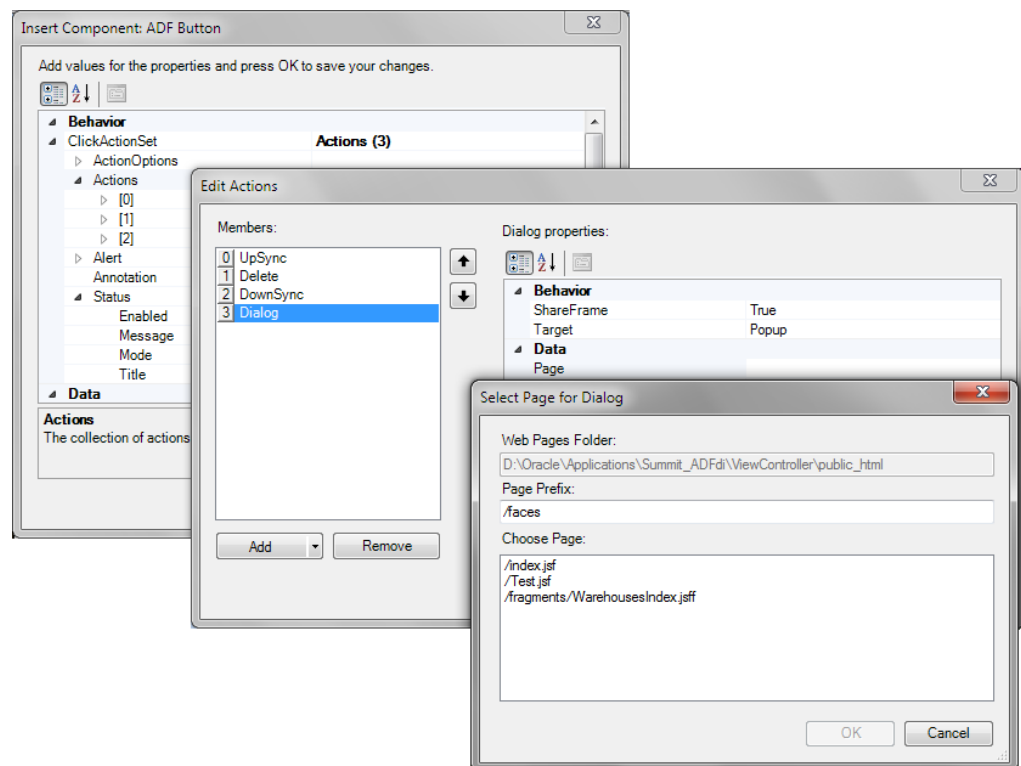
Folder Name	Description
Workbook	Lists parameters defined in <code>Workbook.Parameters</code> .
Worksheet	Lists the <code>errors</code> expression.
Excel Functions	Lists sample Excel functions that you can use with ADF Desktop Integration. For more information, see Excel's documentation.

For more information about using the expression builder, see [Section 9.3, "Applying Styles Dynamically Using EL Expressions."](#) For information about the syntax of EL expressions in ADF Desktop Integration, and guidelines on how you write these expressions, see [Appendix B, "ADF Desktop Integration EL Expressions."](#)

5.9 Using the Web Page Picker

Use the web page picker to select a web page from your Fusion web application. At runtime, an Oracle ADF component, for example an Oracle ADF Button component, can invoke the web page that you associate with the Oracle ADF component.

You can invoke the web page picker when you add a Dialog action to an action set in the Action Collector Editor. You use the web page picker to specify a web page for the Page property of the Dialog action, as illustrated in [Figure 5–11](#).

Figure 5–11 Web Page Picker Dialog

For more information about displaying web pages in your integrated Excel workbook, see [Section 8.4, "Displaying Web Pages from a Fusion Web Application."](#)

5.10 Using the File System Folder Picker

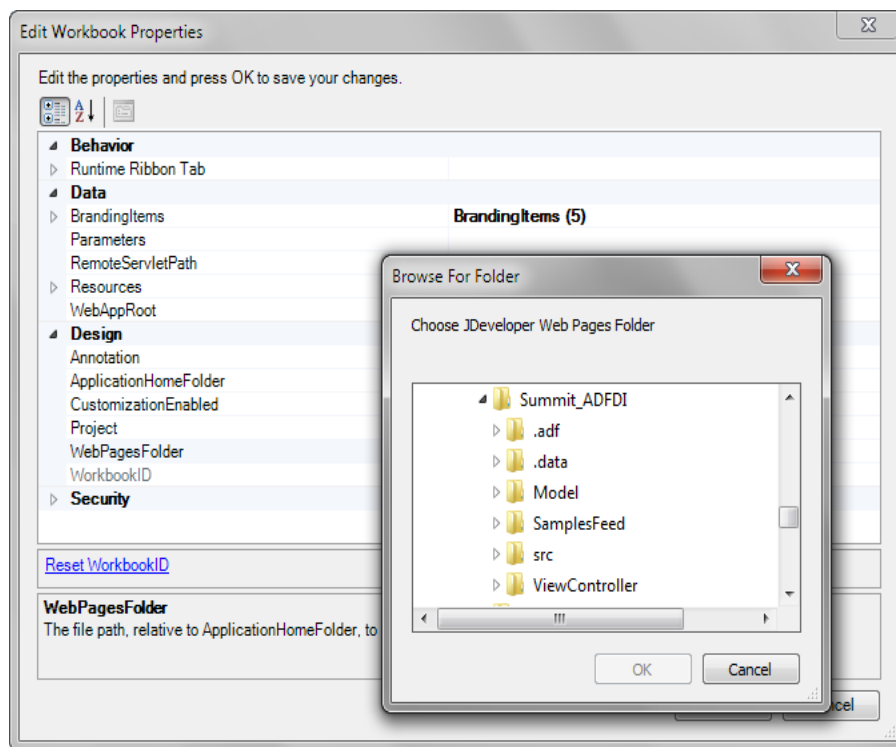
Use the file system folder picker to navigate over the Windows file system and select folders. You use this picker to specify values for the following workbook properties:

- ApplicationHomeFolder
- WebPagesFolder

The first time you open an Excel workbook the picker appears so that you can set values for the previously listed properties. For more information about opening an Excel workbook for the first time and the properties you set, see [Section 4.2.2, "How to Configure a New Integrated Excel Workbook."](#)

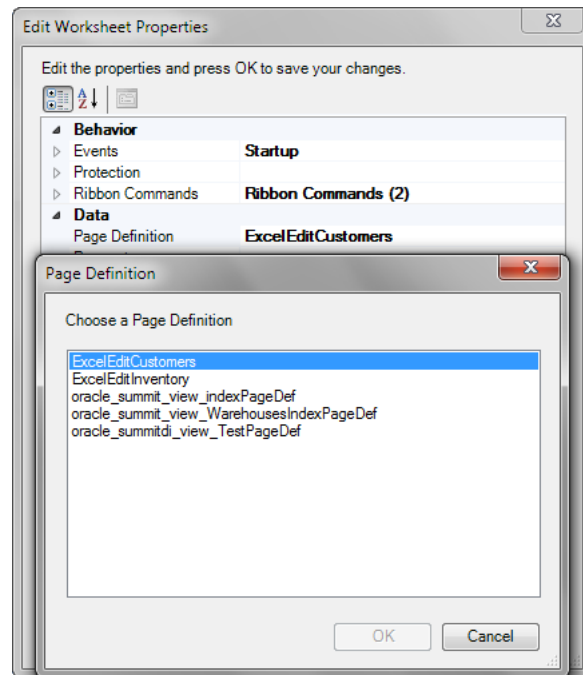
[Figure 5–12](#) shows the file system folder picker selecting a value for the ApplicationHomeFolder workbook property.

Figure 5–12 File System Folder Picker



5.11 Using the Page Definition Picker

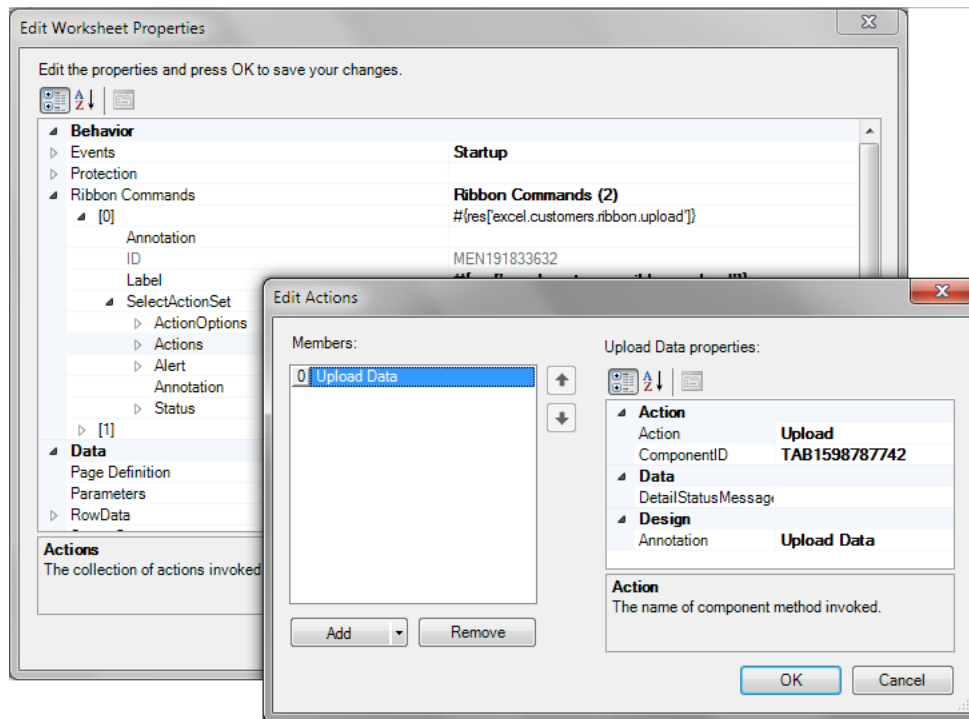
Use the page definition picker to select the page definition ID of a page definition file and associate the file with a worksheet. The picker appears the first time that you activate a worksheet in an integrated Excel workbook. It can also be invoked when you attempt to set a value for the worksheet property, PageDefinition, as illustrated in [Figure 5–13](#).

Figure 5–13 Page Definition Picker

For more information about page definition files, see [Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook."](#)

5.12 Using the Collection Editors

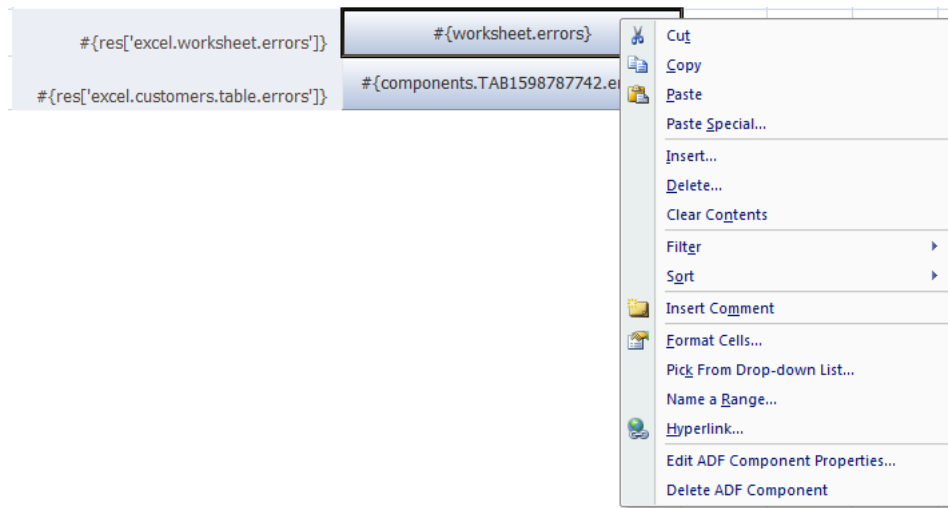
ADF Desktop Integration uses collection editors to manage the properties of elements in a collection. The title that appears in a collection editor's title bar describes what the collection editor enables you to configure. Examples of titles for collection editors include **Edit CachedAttributes**, **Edit Columns**, and the **Edit Actions**. These collection editors allow you to configure collections of cached data, table columns in the ADF Table component, and actions in an action set. [Figure 5–14](#) shows the collection editor.

Figure 5–14 Collection Editor

Tip: Write a description in the Annotation field for each element that you add to the Edit Action dialog. The description you write appears in the **Members** list view and, depending on the description you write, may be more meaningful than the default entry that ADF Desktop Integration generates.

5.13 Using the Cell Context Menu

When working with ADF components at design time, right-click any cell of the component to get menu options to edit or delete the component. Some keyboards feature a key that invokes the context menu. Using this key, you will see the edit and delete menu options as well. [Figure 5–15](#) shows the context menu options of an ADF Output Text component.

Figure 5–15 Context Menu Options of the ADF Output Text Component

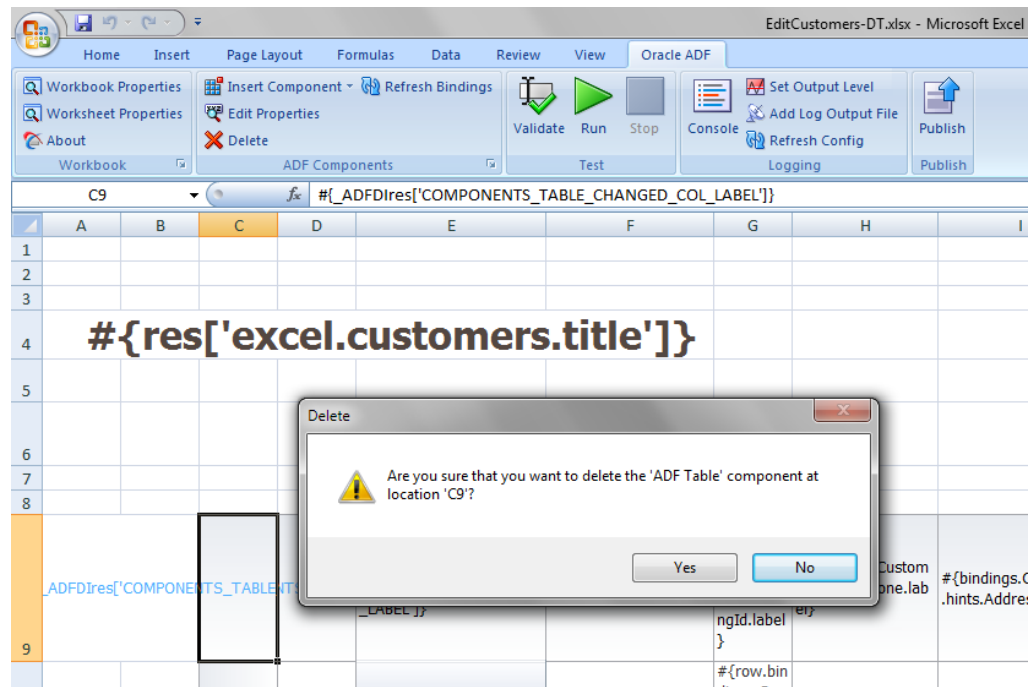
You should use either the keyboard context menu key or the mouse's right-click button during any given session. If you use both the mouse right-click and the context menu key, the menu options may not always appear when expected. The context menu options do not appear if you select a range of cells and then invoke the context menu.

5.14 Removing ADF Desktop Integration Components

At design time, you can remove the inserted ADF Desktop Integration components (or bindings) from the integrated workbooks using the **Delete** ribbon command, or the **Delete ADF Component** context menu option (see [Figure 5–15](#)).

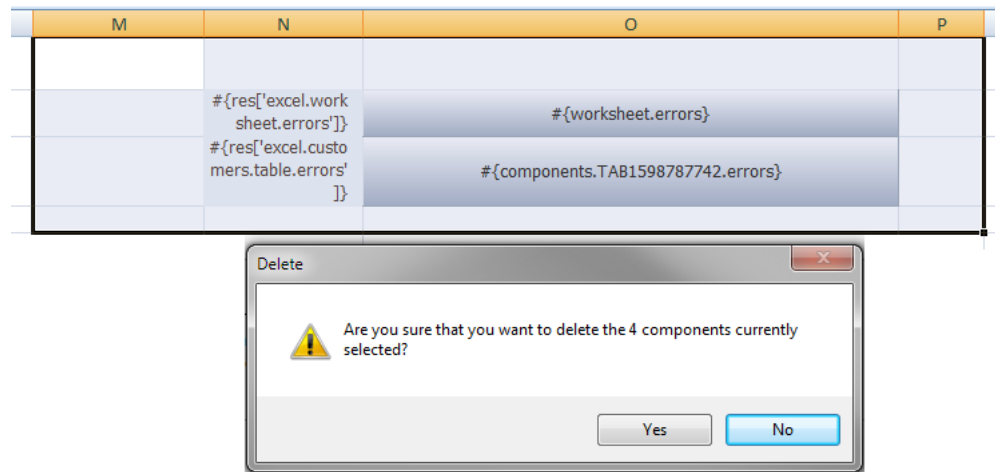
When you remove a component, ADF Desktop Integration prompts you to confirm your action, as shown in [Figure 5–16](#).

Figure 5–16 Removing ADF Desktop Integration Component



You can also remove multiple components by selecting a range of cells anchoring the components (see [Figure 5–17](#)), or select individual component cells using the Ctrl key, and then click the **Delete** ribbon button.

Figure 5–17 Removing Multiple ADF Desktop Integration Components



While removing the components, make a note of the following:

- You can also remove a single component using the Delete key of the keyboard when the component origin cell is selected.
You cannot remove a table component using the Delete key of the keyboard, use the Delete ribbon command instead.
- To delete a component that occupies more than one cell (such as a table component, or a component in a merged cell), you need not select the entire

component. If the selected range intersects any cell of the component, it will be removed.

- Do not delete cells or clear cells of the workbook if your selection includes one or more ADF Desktop Integration components. Always use the Delete ribbon command to remove a ADF Desktop Integration component.
- ADF Desktop Integration context menu options are not available if multiple cells are selected when the context menu is invoked.

5.15 Exporting and Importing Excel Workbook Integration Metadata

Workbook integration metadata, also known as the workbook definition, is a set of information that describes how a given workbook is integrated with a particular Fusion web application. It includes the placement and configuration of components as well as workbook- and worksheet-level properties. Workbook integration metadata is defined by Oracle ADF. It does not include settings of a workbook that are native to Excel.

You can export the integration metadata of your Excel workbook to an XML file with a name and location that you specify. The XML file contains child elements for each worksheet in the workbook, resources such as the relative path to the remote servlet, and so on. The exported XML file enables you to do the following actions:

- Edit, or analyze the Excel workbook that is integrated with a Fusion web application.
- Using an XML editor, copy or move components between worksheets and workbooks.
- Copy action-set definitions between buttons or events.
- Perform global search and replace operations.
- Quickly rearrange, or copy, columns of table components.

5.15.1 How to Export Workbook Integration Metadata

The following procedure describes how you export XML configuration metadata from an integrated Excel workbook.

Before you begin:

It may be helpful to have an understanding of workbook integration metadata. For more information, see [Section 5.15, "Exporting and Importing Excel Workbook Integration Metadata."](#)

To export workbook integration metadata from an integrated Excel workbook:

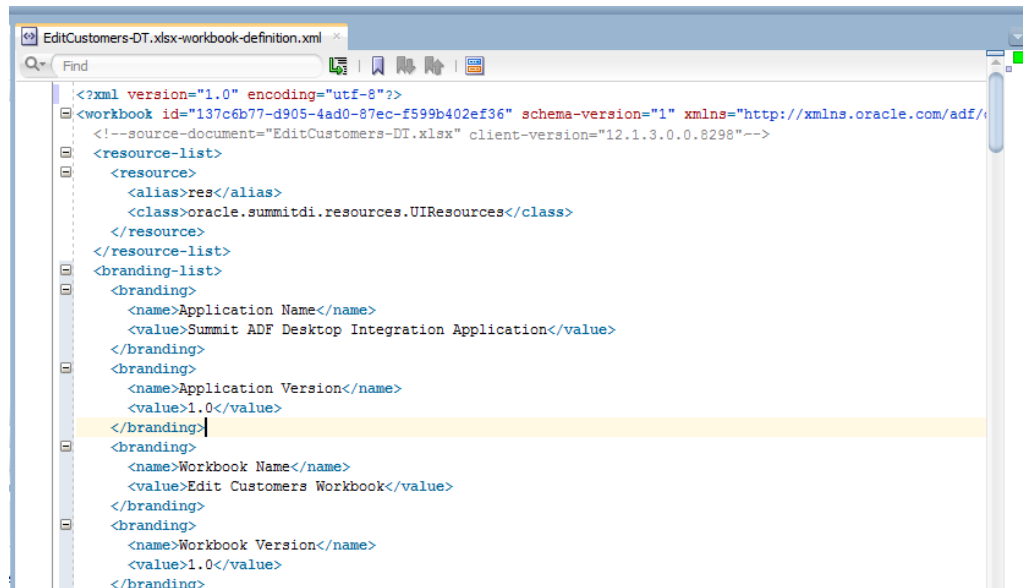
1. Open the integrated Excel workbook.
2. Click **Export** in the **Oracle ADF** tab.
The Save Workbook Definition As dialog box appears.
3. Specify the file name and location of the XML file that stores the exported metadata, and click **Save**. ADF Desktop Integration writes the workbook definition to the specified file.
4. In Export Workbook Metadata dialog, click **OK** to complete the export process.

Note: The exported XML file does not contain any native Excel settings such as named styles, named ranges, cell properties, content in unbound cells, and so on. The file name comprises the full name of the design-time workbook suffixed with `-workbook-definition.xml`. For example, the exported XML file name of `EditCustomers-DT.xlsx` is `EditCustomers-DT.xlsx-workbook-definition.xml`.

Publishing a workbook also exports the workbook definition. For more information about publishing a workbook, see [Section 14.3, "Publishing Your Integrated Excel Workbook."](#)

After exporting the workbook definition, you can edit the XML file in any XML editor, such as JDeveloper. [Figure 5–18](#) shows the workbook definition of `EditCustomers-DT.xlsx` in JDeveloper. While editing the workbook definition file in JDeveloper, JDeveloper automatically validates your changes against the workbook definition schema. It will display warnings that help you avoid problems later on.

Figure 5–18 *Editing Workbook Definition in JDeveloper*



5.15.2 How to Import Workbook Integration Metadata

After editing, you can import the integrated metadata into an empty integrated workbook to create a copy of the source integrated Excel workbook. Note that the empty workbook must be enabled with ADF Desktop Integration before you import the metadata.

The following procedure describes how to import XML configuration metadata to an integrated Excel workbook.

Before you begin:

It may be helpful to have an understanding of workbook integration metadata. For more information, see [Section 5.15, "Exporting and Importing Excel Workbook Integration Metadata."](#)

Before you import the integration metadata from an XML file, perform basic XML validations such as whether the XML code is well formed and the XML file contains the root element. You may import the workbook definition into the same workbook from which it was exported, or import it in a new workbook.

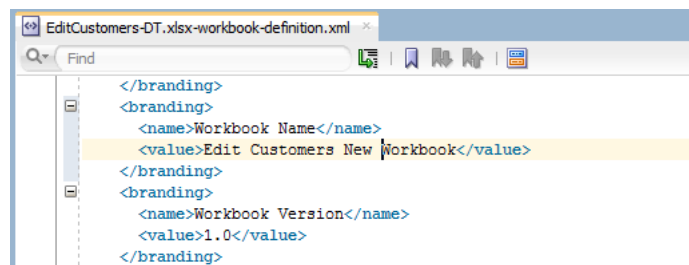
To import workbook integration metadata to an integrated Excel workbook:

1. Open the integrated Excel workbook.
2. Click **Import** in the **Oracle ADF** tab.
The Choose a Workbook Definition file to Import dialog box appears.
3. Select the XML file that stores the workbook integration metadata, and click **Open**.
4. In Import Workbook Metadata dialog, click **OK** to complete the import process.

The changes made in the workbook definition appear automatically in the integrated Excel workbook.

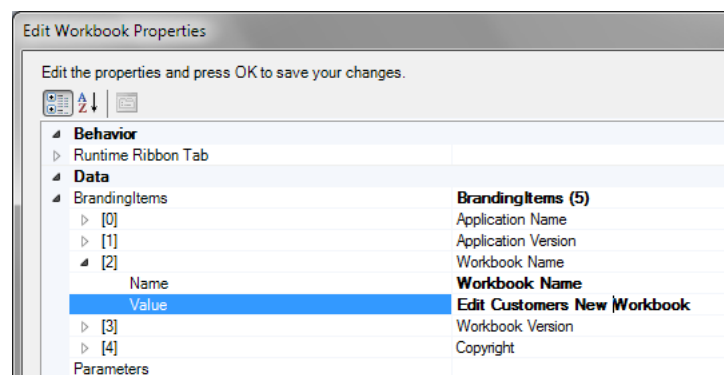
For example, [Figure 5–19](#) shows the branding value of workbook changed to Edit Customers New Workbook in the workbook definition file.

Figure 5–19 *Editing Branding Value in the Workbook Definition*



[Figure 5–20](#) shows the changed branding workbook value in the Edit Workbook Properties dialog after importing the workbook definition.

Figure 5–20 *Updated Branding Value in Edit Workbook Properties Dialog*



5.15.3 What You May Need to Know About Exporting and Importing Excel Workbook Integration Metadata

The workbook integration metadata XML file uses the `adfdi-workbook-definition.xsd` XML schema document, which defines the XML namespace as `http://xmlns.oracle.com/adf/desktopintegration/workbook`. The schema is integrated into JDeveloper through the ADF Desktop Integration add-in.

You can find a copy of the schema at `<MW_HOME>\jdeveloper\adfdi\etc\adfdi-workbook-definition.xsd`, where `MW_HOME` is the Middleware Home directory.

While importing the workbook integration metadata, make a note of following points:

- When the import process is initiated, the schema version number (`schema-version` attribute of `<workbook>`) of the XML file is compared against the schema version number of the installed ADF Desktop Integration client.

If both values match, the workbook integration metadata is imported to the workbook. If the schema version of the XML file is lower than the schema version of the installed client, the XML file is migrated to use the installed client's schema. No prompt appears when the file is migrated, but a log of the same is maintained. If the schema version of the XML file is greater than the schema version of the installed client, the import process fails and an error message appears.

- After verifying the schema version, the imported XML file is validated against the schema of the installed client. If the validation fails, the validation failure details are logged, an error is reported to the user, and the import process aborts. If the schema validation succeeds, the import process continues.
- If an element is missing in the imported XML file, the default value of the element is used in the integrated Excel workbook.
- All pre-existing worksheet and component metadata is removed before the import.
- If the imported worksheet's name matches an existing worksheet in the integrated workbook, that worksheet is used. Otherwise, a new worksheet is created.
- All non-integrated worksheets of the integrated Excel workbook are not affected by the import.
- If the imported component does not have valid origin information, the import process attempts to place that component on the first unused row in the target integrated worksheet.
- After the XML file is imported, the integrated Excel workbook's **Workbook ID** is replaced with the Workbook ID of the XML file. If the workbook ID is missing in the XML file, a new ID is generated.

Working with ADF Desktop Integration Form-Type Components

This chapter describes how to insert and configure components (such as buttons, labels, input and output text, and list of values) that ADF Desktop Integration provides to allow end users to manage data retrieved from a Fusion web application, and how to display calculated data in these components using Excel formulae.

This chapter includes the following sections:

- [Section 6.1, "About ADF Desktop Integration Form-Type Components"](#)
- [Section 6.2, "Inserting an ADF Button Component"](#)
- [Section 6.3, "Inserting an ADF Label Component"](#)
- [Section 6.4, "Inserting an ADF Input Text Component"](#)
- [Section 6.5, "Inserting an ADF Output Text Component"](#)
- [Section 6.6, "Inserting an ADF List of Values Component"](#)
- [Section 6.7, "Inserting an ADF Image Component"](#)
- [Section 6.8, "Inserting Error Components"](#)
- [Section 6.9, "Displaying Tooltips in ADF Desktop Integration Form-Type Components"](#)
- [Section 6.10, "Displaying Output from a Managed Bean in an ADF Component"](#)
- [Section 6.11, "Displaying Concatenated or Calculated Data in Components"](#)
- [Section 6.12, "Using Navigation Buttons"](#)

6.1 About ADF Desktop Integration Form-Type Components

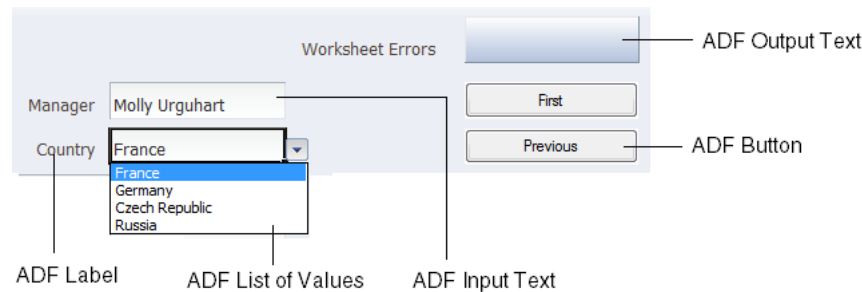
The ADF Desktop Integration Form-type components allow end users to manage data retrieved from the Fusion web application in the integrated Excel workbook. Rather than expose an ADF Form component in the components palette described in [Section 5.5, "Using the Components Palette,"](#) ADF Desktop Integration uses the following components to create form-type functionality in an integrated Excel workbook:

- ADF Input Text
- ADF Output Text
- ADF Label
- ADF List of Values

- ADF Button

Figure 6–1 shows these components.

Figure 6–1 ADF Desktop Integration Form-Type Components



You can insert a form component or a binding in a merged cell, or merge cells after inserting the form component or binding, but you cannot insert multiple form components in a merged cell or merge cells that are occupied by different form components.

Before you insert a component in a merged cell, make a note of the following:

- Drag-and-drop functionality is not supported for inserting component in a merged cell.
- Do not merge a component cell with non-empty cells that are above or left to it. When two or more cells are merged, Excel keeps the data and style of the most upper-left cell and discards the data of the remaining cells. So, merging a component cell with a non-empty cell above or left to itself results in the component data being overwritten.
- Do not merge an empty component cell that has no value or binding with empty cells above or left to it. Merging an empty component cell with empty cells above or left to itself results in the style of that component cell being overwritten.
- ADF Buttons do not expand to the whole merged area automatically. You can edit the `Position` and `LowerRightCorner` properties of the button to resize it as needed.

6.1.1 ADF Desktop Integration Form-Type Components Use Cases and Examples

The ADF Desktop Integration Form-type components are used to build forms in the integrated Excel workbook for user input, and output from the Fusion web application. As shown in Figure 6–2, the form-type components used in navigation form of `EditWarehouses-DT.xlsx` enable end users to navigate and update data easily.

Figure 6–2 Navigation Form Using ADF Desktop Integration Form-Type Components

6.1.2 Additional Functionality for ADF Desktop Integration Form-Type Components

After you have added a component to the worksheet, you may find that you need to add functionality such as responding to events or end user actions. Following are links to other functionality that form components can use:

- **Displaying output from a managed bean:** You can use ADF Label or ADF Output Text components to display output from a managed bean. For more information, see [Section 6.10, "Displaying Output from a Managed Bean in an ADF Component."](#)
- **Styles:** You can configure the display of your form-type components using several predefined Excel styles. For more information, see [Section 9.2, "Working with Styles."](#)
- **EL Expressions:** You can use EL expressions with form-type components. For more information, see [Appendix B, "ADF Desktop Integration EL Expressions."](#)

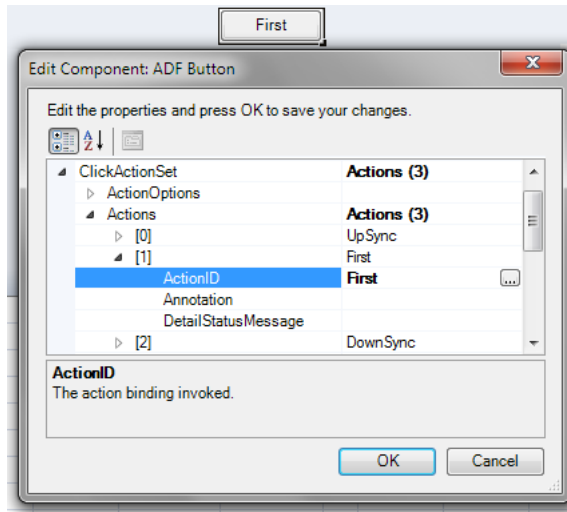
6.2 Inserting an ADF Button Component

The ADF Button component renders a button in the Excel worksheet at runtime. End users click this button to invoke one or more actions specified by the `ClickActionSet` group of properties.

The `LowerRightCorner` and `Position` properties determine the area that the button occupies on the Excel worksheet at runtime.

[Figure 6–3](#) shows an ADF Button component with its property inspector. When an end user clicks the button at runtime, it invokes the array of actions specified by `ClickActionSet`.

Figure 6–3 ADF Button Component in Design Mode



For more information about the properties of the ADF Button component, see [Section A.11, "ADF Button Component Properties."](#)

To insert an ADF Button component:

1. Open the integrated Excel workbook.
2. Select the cell in the Excel worksheet where you want to anchor the component.
3. In the components palette, select **ADF Button** and click **Insert Component**. Alternatively, in the **Oracle ADF** tab, select **ADF Button** from the **Insert Component** dropdown list.
4. Configure properties in the property inspector to determine the actions the component invokes at runtime in addition to the appearance, design, and layout of the component. [Table 6–1](#) outlines some properties you must specify values for, and provides links to additional information.

Table 6–1 ADF Button component properties

For this property...	Specify...
Label	<p>A string or an EL expression that resolves to a label at runtime to indicate the purpose of the ADF Button component. The button label defaults to the action binding ID.</p> <p>The EL expression, if used, references a string key in the <code>res</code> resource bundle. For more information about resource bundles, see Section 10.2, "Using Resource Bundles in an Integrated Excel Workbook." For more information about using labels in integrated Excel workbooks, see Section 9.4, "Using Labels in an Integrated Excel Workbook."</p> <p>To include the ampersand (&) character in the label, use <code>&&</code>. A single <code>&</code> character acts as a special character and is not displayed in the label.</p>
ClickActionSet	<p>Specify one or more actions in the <code>Actions</code> array of the <code>ClickActionSet</code> that the end user invokes when he or she clicks the ADF Button component. For more information about action sets, see Section 8.2, "Using Action Sets."</p>

5. Click **OK**.

Figure 6–4 shows an example of the ADF Button component (in red box) at runtime.

Figure 6–4 ADF Button Component at Runtime

Notes:

- If you change the view mode of the Excel worksheet to the Page Layout or Page Break mode, the ADF Button components may be rendered in an unexpected position. You must return back to Normal mode without saving the workbook, and then Run and stop the integrated Excel workbook to render the buttons to their original positions.
 - You can modify the properties of the component at a later time by selecting the cell in the worksheet. Click the ADF Button component to open its property inspector.
 - The ADF Button components are active at 100% zoom only, and are disabled when the end user zooms in or out on an integrated Excel worksheet.
 - To remove the component, use the Delete ribbon command. For more information, see [Section 5.14, "Removing ADF Desktop Integration Components."](#)
-
-

Tip: In design mode, you can click the button, or press the spacebar when the button is in focus, to open the property inspector. The right-click context menu is disabled for a button.

If you want to add navigation buttons in your integrated Excel workbook to navigate to previous or next record, see [Section 6.12, "Using Navigation Buttons."](#)

6.3 Inserting an ADF Label Component

The ADF Label component is a component that you can insert into the active worksheet to display a static string value. You specify a value in the input field for `Label` in the property inspector or alternatively you invoke the expression builder to write an EL expression that resolves to a string at runtime. The retrieved string can be defined in a resource bundle or in an attribute control hint for an entity or view object. For example, the following EL expression resolves to the value of label of **CountryId** attribute binding at runtime:

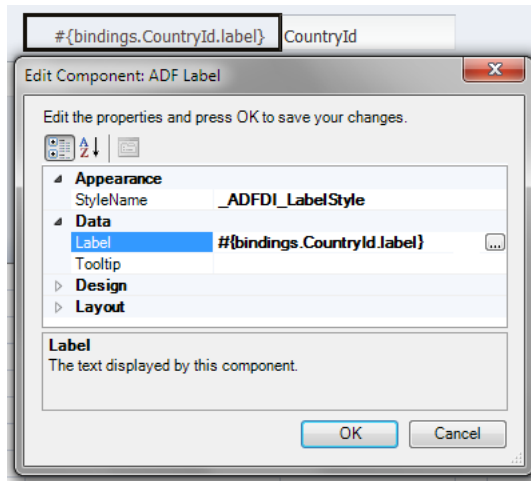
```
#{bindings.CountryId.label}
```

The value that you specify for the `Label` property in an ADF Label component or other Oracle ADF components is evaluated after the worksheet that hosts the Oracle ADF component is initialized (opened for the first time).

You can configure a number of properties for the component, such as style and position, in the worksheet using the property inspector.

Figure 6–5 shows an ADF Label component with its property inspector in the foreground. The ADF Label component references an EL expression that resolves to the label of `CountryId` attribute binding at runtime.

Figure 6–5 ADF Label Component in Design Mode

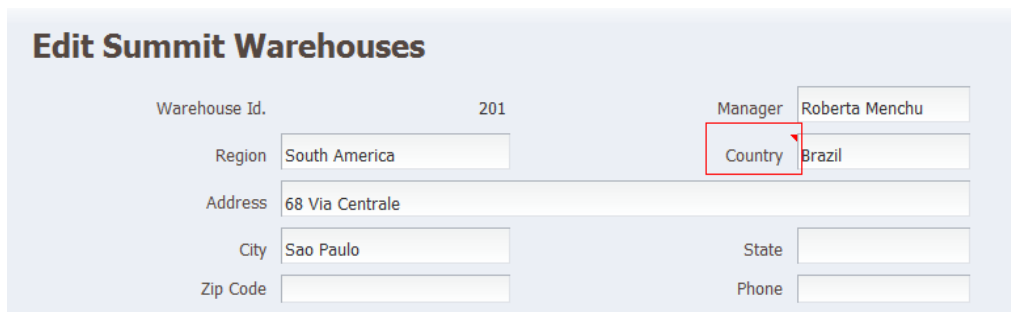


To insert an ADF Label component:

1. Open the integrated Excel workbook.
2. Select the cell in the Excel worksheet where you want to anchor the component.
3. In the components palette, select **ADF Label** and click **Insert Component**. Alternatively, in the **Oracle ADF** tab, select **ADF Label** from the **Insert Component** dropdown list
4. Configure properties in the property inspector to determine the appearance, design, and layout of the component.
5. Click **OK**.

Figure 6–6 shows an example of the ADF Label component (in red box) at runtime.

Figure 6–6 ADF Label Component at Runtime



Note: You can modify the properties of the component at a later time by selecting the cell in the worksheet that anchors the component and then displaying the property inspector. You can also right-click in the cell and choose **Edit ADF Component Properties** to open the property inspector.

To remove the component, use the Delete ribbon command. For more information, see [Section 5.14, "Removing ADF Desktop Integration Components."](#)

For more information about using labels in an integrated Excel workbook, see [Section 9.4, "Using Labels in an Integrated Excel Workbook."](#)

6.4 Inserting an ADF Input Text Component

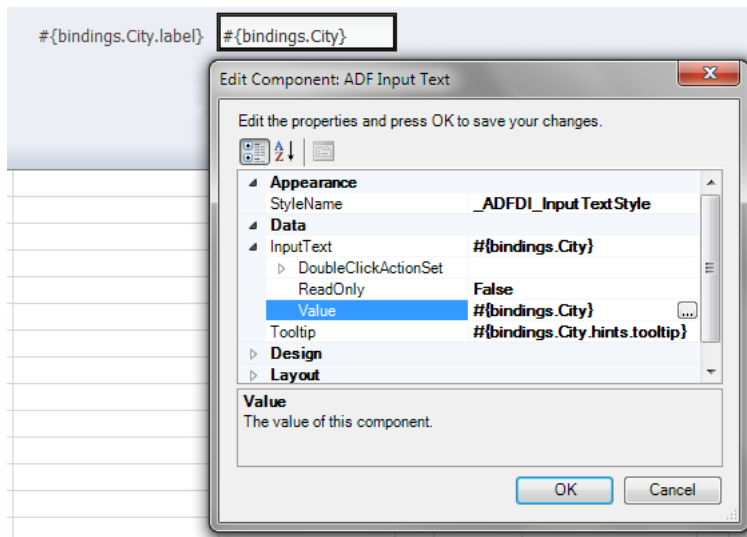
The ADF Input Text component is a component that you can insert into the active worksheet using the components palette. The active cell in the worksheet when you insert the component displays the current value from the component's binding after the worksheet `DownSync` action is invoked. End users can edit this value at runtime. You configure the worksheet `UpSync` action to transfer changes end users make to the value back to the Fusion web application and a `Commit` action binding to commit the changes in the Fusion web application.

You can configure a number of properties for the component, such as its position, style and behavior when a user double-clicks the cell (`DoubleClickActionSet` properties), in the worksheet using the property inspector. For more information about `DoubleClickActionSet`, see [Section 8.2, "Using Action Sets."](#)

The ADF Table component can invoke this component as a subcomponent when you set values for the ADF Table component column's `InsertComponent` or `UpdateComponent` properties. In this context, the ADF Input Text component enables the end user to input data into the ADF Table component. For more information, see [Section 7.7, "Configuring an ADF Table Component to Insert Data."](#)

[Figure 6-7](#) shows an ADF Input Text component with its property inspector in the foreground. The ADF Input Text component binds to the `City` attribute binding in the Summit sample application for ADF Desktop Integration. The end user enters a city name in this component.

Figure 6–7 ADF Input Text Component in Design Mode



To insert an ADF Input Text component:

1. Open the integrated Excel workbook.
2. Select the cell in the Excel worksheet where you want to anchor the component.
3. In the components palette, select **ADF Input Text** and click **Insert Component**. Alternatively, in the **Oracle ADF** tab, select **ADF Input Text** from the **Insert Component** dropdown list
4. Configure properties in the property inspector to determine the appearance, layout, and behavior of the component. [Table 6–2](#) outlines some properties that you must specify values for. For information about the component’s other properties, see [Section A.2, "ADF Input Text Component Properties."](#)

Table 6–2 ADF Input Text component properties

For this property...	Specify...
InputText.Value	An EL expression for the Value property to determine what binding the component references.
InputText.ReadOnly	An EL expression that resolves to <code>False</code> so that changes the end user makes are uploaded. Write an EL expression that resolves to <code>True</code> if you want the component to ignore changes. <code>False</code> is the default value.

5. Click **OK**.

Note: You can modify the properties of the component at a later time by selecting the cell in the worksheet that anchors the component and then displaying the property inspector. You can also right-click in the cell and choose **Edit ADF Component Properties** to open the property inspector.

To remove the component, use the Delete ribbon command. For more information, see [Section 5.14, "Removing ADF Desktop Integration Components."](#)

Figure 6–8 shows an example of the ADF Input Text component (in red box) at runtime.

Figure 6–8 ADF Input Text Component at Runtime

Edit Summit Warehouses

Warehouse Id. 201 Manager Roberta Menchu

Region South America Country Brazil

Address 68 Via Centrale

City Sao Paulo State

Zip Code Phone

6.5 Inserting an ADF Output Text Component

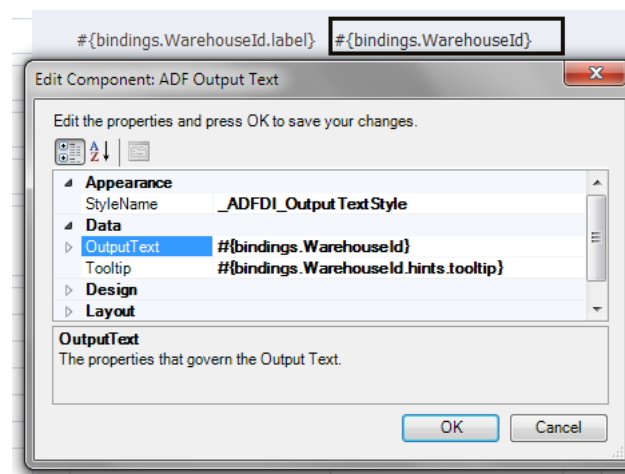
The ADF Output Text component is a component that you can insert into the active worksheet using the components palette. The active cell in the worksheet when you insert the component displays the current value from the component's binding after you invoke the worksheet `DownSync` action. The value the component displays is read-only. Changes that the end user makes to the value in the cell that anchors the component are ignored when changes are sent to the Fusion web application.

This component can also serve as a subcomponent for the ADF Table and ADF Read-only Table components. Columns in the ADF Table and ADF Read-only Table components can be configured to use the ADF Output Text component.

You can configure a number of properties for the component such as style, behavior when a user double-clicks the cell (`DoubleClickActionSet` properties), and position, in the worksheet using the property inspector.

Figure 6–9 shows an ADF Output Text component with its property inspector in the foreground.

Figure 6–9 ADF Output Text Component in Design Mode



To insert an ADF Output Text component:

1. Open the integrated Excel workbook.

2. Select the cell in the Excel worksheet where you want to anchor the component.
3. In the components palette, select **ADF Output Text**, and click **Insert Component**. Alternatively, in the **Oracle ADF** tab, select **ADF Output Text** from the **Insert Component** dropdown list
4. Configure properties in the property inspector to determine the appearance, layout, and behavior of the component.

For example, you must write or specify an EL expression for the `Value` property to determine what binding the ADF Output Text component references. For more information about the values that you specify for the properties of the ADF Output Text component, see [Section A.3, "ADF Output Text Component Properties."](#)

5. Click **OK**.

Note: You can modify the properties of the component at a later time by selecting the cell in the worksheet that anchors the component and then displaying the property inspector. You can also right-click in the cell and choose **Edit ADF Component Properties** to open the property inspector.

To remove the component, use the Delete ribbon command. For more information, see [Section 5.14, "Removing ADF Desktop Integration Components."](#)

Figure 6–10 shows an example of the ADF Output Text component (in red box) at runtime.

Figure 6–10 ADF Output Text Component at Runtime

Edit Summit Warehouses	
Warehouse Id.	201
Region	South America
Address	68 Via Centrale
City	Sao Paulo
Zip Code	
Manager	Roberta Menchu
Country	Brazil
State	
Phone	

6.6 Inserting an ADF List of Values Component

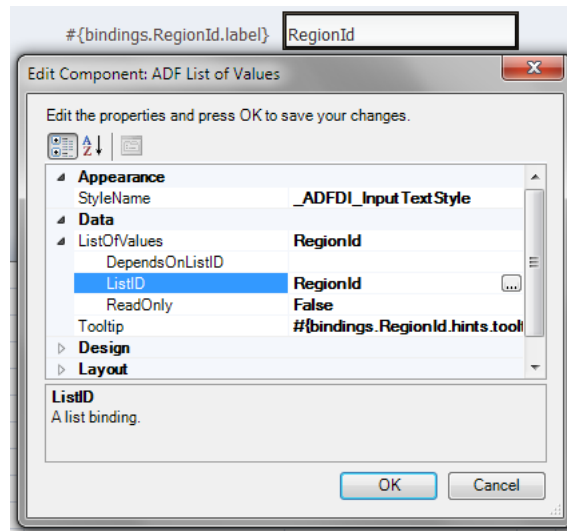
The ADF List of Values component is a component that displays a dropdown menu in the Excel worksheet cell at runtime. It displays a maximum of 250 values at runtime. You can insert the List of Values component into a cell in the Excel worksheet.

You must specify a value for the `ListID` property. The `ListID` property references the list binding which populates the dropdown menu with a list of values at runtime after you invoke the worksheet `DownSync` action.

Figure 6–11 shows an ADF List of Values component with its property inspector in the foreground. The ADF List of Values component references a list binding (`RegionId`) that populates a dropdown menu in the Excel worksheet at runtime.

Notes:

- You can display a dropdown menu in an ADF Table component's column by selecting `TreeNodeList` or `ModelDrivenColumnComponent` as the subcomponent to create when you specify a value for the `TableColumn` array's `InsertComponent` property. For more information, see [Section 7.14, "Creating a List of Values in an ADF Table Component Column."](#)
- ADF List of Values components using date values are not supported.

Figure 6–11 ADF List of Values Component**To insert an ADF List of Values component:**

1. Open the integrated Excel workbook.
2. Select the cell in the Excel worksheet where you want to anchor the component.
3. In the components palette, select **ADF List of Values** and click **Insert Component**. Alternatively, in the **Oracle ADF** tab, select **ADF List of Values** from the **Insert Component** dropdown list
4. Invoke the binding ID picker by clicking the browse (...) icon beside the input field for the **ListID** property and select a list binding that the page definition file exposes.
5. Configure other properties in the property inspector to determine the appearance, design, and layout of the component. For information about ADF List of Values component properties, see [Section A.7, "ADF List of Values Component Properties."](#)
6. Click **OK**.

Notes:

- You can modify the properties of the component at a later time by selecting the cell in the worksheet that anchors the component and then displaying the property inspector. You can also right-click in the cell and choose **Edit ADF Component Properties** to open the property inspector.

To remove the component, use the Delete ribbon command. For more information, see [Section 5.14, "Removing ADF Desktop Integration Components."](#)

- An Excel form cannot be configured to use ADF List of Values components that use model-driven list bindings if the form's bound iterator is expected to contain zero rows. As a workaround, you may configure the ADF List of Values component to use a dynamic list binding instead.

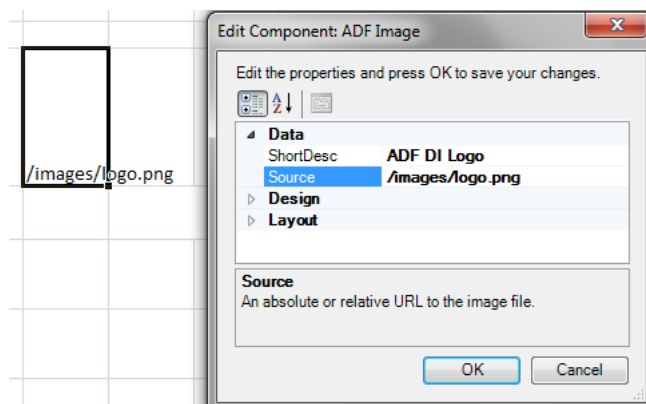
6.7 Inserting an ADF Image Component

Using the ADF Image component, you can insert an image (for example, a company logo) in the integrated Excel worksheet. At design time, specify the URL of the image file in *Source*, and the ADF Image component renders the image at runtime. The image is rendered at original size at runtime.

At runtime, when the image component is rendered, ADF Desktop Integration determines whether the *Source* property value is an absolute URL or a relative URL. The source URL is considered to be absolute if it starts with `http` and `https`, the only supported schemes. If the URL is absolute, it is used as is to fetch the image and insert that image into the worksheet. If the URL is not absolute, the partial URI is assumed to be relative to the workbook's *WebAppRoot*. In such a case, the *WebAppRoot* value and the *Source* value are concatenated to form the complete image URL. If the attempt to fetch and insert the image fails for any reason (such as, invalid URL), a log entry is created. The technical details regarding the failure are reported in the client logs. The worksheet initialization is not interrupted and there are no warnings or error messages presented to the end user. The end user sees the short description of the image in the cell location where the image would have been displayed.

[Figure 6–12](#) shows an ADF Image component in `EditCustomers-DT.xlsx` at design time.

Figure 6–12 ADF Image Component in the Design mode



To insert an ADF Image component:

1. Open the integrated Excel workbook.
2. Select the cell in the Excel worksheet where you want to anchor the component.
3. In the components palette, select **ADF Image** and click **Insert Component**. Alternatively, in the **Oracle ADF** tab, select **ADF Image** from the **Insert Component** dropdown list
4. Configure properties in the property inspector to determine the appearance, layout, and behavior of the component. [Table 6–3](#) outlines some properties that you must specify values for. For information about the component's other properties, see [Section A.8, "ADF Image Component Properties."](#)

Table 6–3 ADF Image component properties

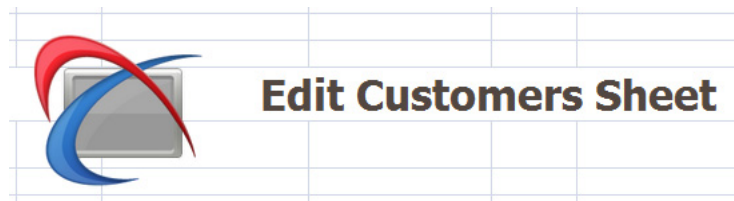
For this property...	Specify...
Source	<p>The URL of the image file.</p> <p>You can use absolute or relative URLs as the source of the image. If the URL is not absolute, the partial URI is assumed to be relative to the workbook's <code>WebAppRoot</code>.</p> <p>Examples:</p> <pre>/images/myLogo.png</pre> <pre>/resourceServlet?image=myLogo</pre> <pre>http://www.oracle.com/ocom/groups/public/@otn/documents/digitalasset/110224.gif</pre> <p>Note that the <code>Source</code> property does not support EL expressions.</p> <p>For the list of supported image formats, see Microsoft Excel documentation.</p>
ShortDesc	<p>The String message as the alternate text of the image, if the image is not found or cannot be rendered. You can also specify an EL expression that resolves to the alternate text of the image component.</p> <p>Note that the <code>ShortDesc</code> property does not support binding expressions.</p>

5. Click **OK**.

Note: You can modify the properties of the component at a later time by selecting the cell in the worksheet that anchors the component and then displaying the property inspector. You can also right-click in the cell and choose **Edit ADF Component Properties** to open the property inspector.

To remove the component, use the Delete ribbon command. For more information, see [Section 5.14, "Removing ADF Desktop Integration Components."](#)

[Figure 6–13](#) shows an example of the ADF Image component at runtime.

Figure 6–13 ADF Image Component at Runtime

Note: If the worksheet is not protected, the end user may resize or move the image at runtime. Depending on the size of the image, it might appear over (and hide) other worksheet contents, including other ADF Desktop Integration components.

6.8 Inserting Error Components

ADF Desktop Integration provides two error components, **ADF Output Text (Worksheet Errors)** and **ADF Output Text (Table Errors)** that you can use to display error details generated by the integrated Excel worksheet and the ADF Table component respectively.

Both error components use the ADF Output Text component to display the error message. The ADF Output Text (Worksheet Errors) component is automatically configured to display the worksheet errors, and the ADF Output Text (Table Errors) is automatically configured to display the table errors.

To insert an Error component:

1. Open the integrated Excel workbook.
2. Select the cell in the Excel worksheet where you want to anchor the component.
3. In the components palette, select **ADF Output Text (Worksheet Errors)** or **ADF Output Text (Table Errors)**, as desired, and click **Insert Component**.

If the current worksheet contains one, and only one, table component, the ADF Output Text (Table Errors) automatically displays all error messages of the table.

If the current worksheet contains multiple table components, you are prompted to choose a table before proceeding with the add component operation. Select the ID of the table to associate with the **ADF Output Text (Table Errors)** component in the Select Table dialog, and click **OK**. To insert the ADF Output Text (Table Errors) component, the worksheet must contain at least one ADF Table component.

Alternatively, in the **Oracle ADF** tab, select **ADF Output Text (Worksheet Errors)** or **ADF Output Text (Table Errors)** from the **Insert Component** dropdown list.

4. If required, configure component properties in the property inspector.
5. Click **OK**.

Note: You can modify the properties of the component at a later time by selecting the cell in the worksheet that anchors the component and then displaying the property inspector. You can also right-click in the cell and choose **Edit ADF Component Properties** to open the property inspector.

To remove the component, use the Delete ribbon command. For more information, see [Section 5.14, "Removing ADF Desktop Integration Components."](#)

6.8.1 What You May Need to Know About Error Components

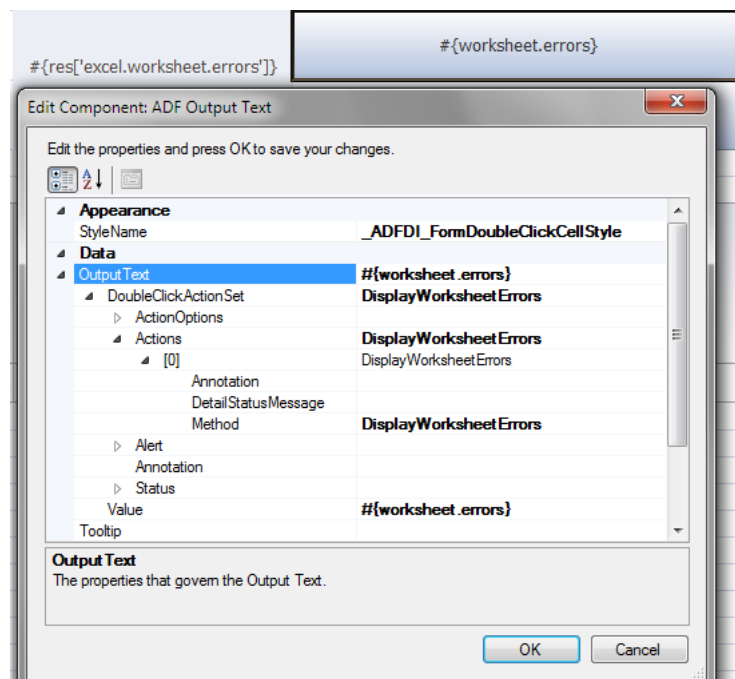
ADF Output Text (Worksheet Errors) Component

To display a worksheet's error message summaries generated by action sets invoked on a worksheet, the ADF Output Text (Worksheet Errors) component uses `#{worksheet.errors}` as the EL expression set to the Value property of the ADF Output Text component. At runtime, the previous error message summary is cleared (if one existed) when the action set starts the invocation. If no errors occur during invocation, the error message remains blank. If an error occurs, the ADF Output Text component displays the error message summary.

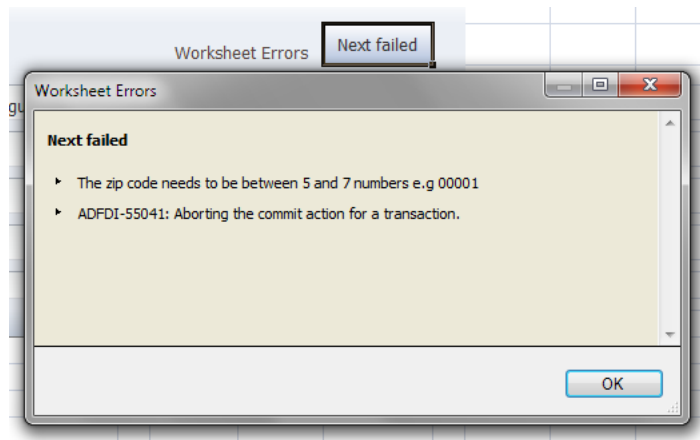
The ADF Output Text (Worksheet Errors) component is pre-configured to use the `DisplayWorksheetErrors` action as a `DoubleClickActionSet` item.

[Figure 6–14](#) shows the ADF Output Text (Worksheet Errors) component at design-time.

Figure 6–14 ADF Output Text (Worksheet Errors) Component in Design Mode



At runtime, double-clicking the ADF Output Text component invokes the `DisplayWorksheetErrors` action as shown in [Figure 6–15](#).

Figure 6–15 Runtime View of ADF Output Text (Worksheet Errors) Component

For more information about the Worksheet's `DisplayWorksheetErrors` action, see [Section A.16, "Worksheet Actions and Properties."](#)

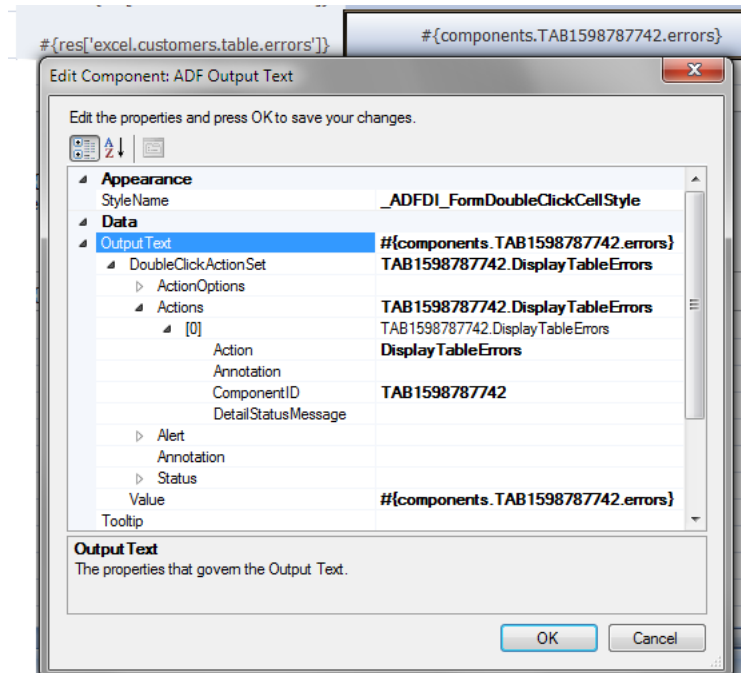
ADF Output Text (Table Errors) Component

To allow the user to view details of errors that occur during table upload batch commit operations, the ADF Output Text (Table Errors) component uses `#{components.tableID.errors}` as the EL expression set to the `Value` property of the ADF Output Text component. At runtime, the previous error message summary is cleared (if one existed) when the action set starts the invocation. If no errors occur during invocation, the error message remains blank. If an error occurs, the ADF Output Text component displays the error message summary.

The ADF Output Text (Table Errors) component is pre-configured to use the ADF Table component's `DisplayTableErrors` action as a `DoubleClickActionSet` item.

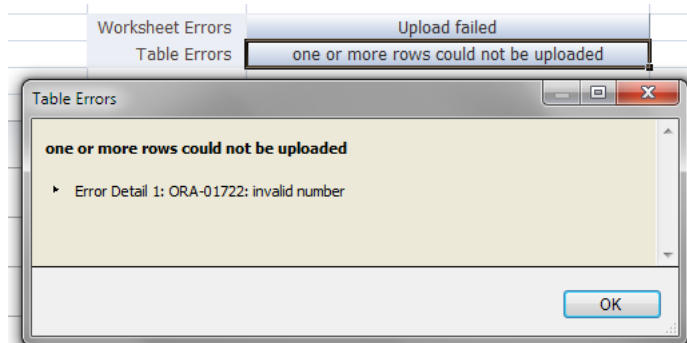
[Figure 6–16](#) shows the ADF Output Text (Table Errors) component at design-time.

Figure 6–16 ADF Output Text (Table Errors) Component in Design Mode



At runtime, double-clicking the ADF Output Text component invokes the `DisplayTableErrors` action as shown in [Figure 6–17](#).

Figure 6–17 Runtime View of `DisplayTableErrors` action



For more information about ADF Table component actions, see [Section A.12, "ADF Table Component Properties and Actions."](#)

6.9 Displaying Tooltips in ADF Desktop Integration Form-Type Components

Using tooltips, you can display a hint or an instruction text for the component. The tooltip is displayed in the Comment window of the cell that anchors the component.

If a component is created from a binding element, the tooltip is automatically set to the model-driven tooltip, else it is empty. Note that the tooltip is always initially empty for the ADF Label component, and is not applicable to the ADF Button component.

You can enter the hint or the instruction text in the `Tooltip` property of the component. You can also specify an EL expression (including a resource expression) that evaluates to the tooltip text.

Note: ADF Desktop Integration also supports `tooltip` attribute control hint in EL expressions. The support is similar to the `mandatory` control hint described in Table B-3 of Appendix B, "ADF Desktop Integration EL Expressions."

To add a tooltip to an inserted form-type component:

1. Open the integrated Excel workbook.
2. If the Form-type component has already been inserted in the Excel worksheet, select the component, and click **Edit Properties** in the Oracle ADF tab.

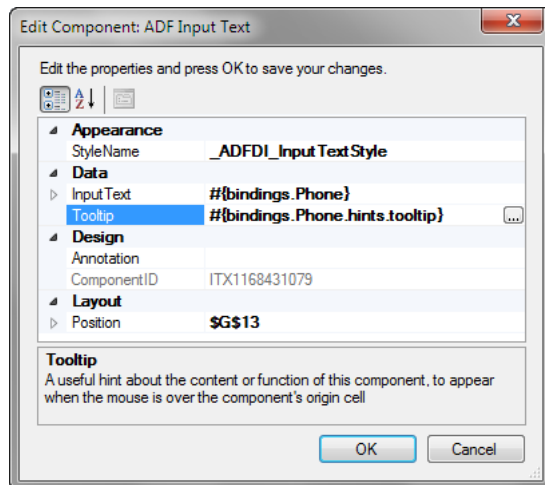
To insert a component to the worksheet, select the cell where you want to anchor the component. In the components palette or the bindings palette, select the Form-type component or the binding, and click **Insert Component** or **Insert Binding**.

3. In the Property Inspector, click the browse (...) icon of the `Tooltip` property, and enter the tooltip message.

If a component is created from a binding element, the `Tooltip` property would be set to the model-driven tooltip. If required, you can configure and change the tooltip message or the EL expression. The property would be empty if the component is not created from a binding element.

Figure 6-18 shows the `Tooltip` property of an Input Text component.

Figure 6-18 *Tooltip Property of Input Text Component*



4. Click **OK**.

Figure 6-19 shows the tooltip message of the **Phone** field of `EditWarehouses-DT.xlsx` at the runtime. Notice the small red arrow at the top-right of the Input Text component. It indicates the component, or the cell, has a comment. Hover mouse pointer over the component to see the tooltip message.

Figure 6–19 *Tooltip Message of Input Text Component at Runtime*

The screenshot shows a web form titled "Edit Summit Warehouses" with the following fields and values:

Warehouse Id.	201	Manager	Roberta Menchu
Region	South America	Country	Brazil
Address	68 Via Centrale		

6.9.1 What You May Need to Know About Tooltips

The tooltips are rendered once only, and are not updated after a call to `Worksheet.DownSync`.

At design-time, if you inserted a comment manually to a cell or a component, ADF Desktop Integration would override it at runtime. The old comment will be removed and replaced with the tooltip message of the component. When returning to design-time mode from test mode, all manually inserted Excel comments are removed as each component is rendered for its design-time appearance.

At runtime, if the `Tooltip` property is non-empty, the expression is evaluated and the resulting text is trimmed of whitespace. If the final value is non-empty, it is inserted into the target cell as an Excel comment.

When a component is positioned on a merged range of cells, the tooltip appears on the top-right corner of the merged range.

You can also add tooltips to table columns (see [Section 7.18.1, "What You May Need to Know About Tooltips for Table Columns"](#)) and Worksheet Ribbon commands (see [Section 8.3.2, "How to Configure a Worksheet Command for the Runtime Ribbon Tab"](#)).

Notes:

- Tooltips are not editable in a protected worksheet.
 - Ribbon command tooltips have a maximum size of 1024 characters. If a tooltip value exceeds that limit, only the first 1024 characters are shown.
 - If Excel Comments are disabled, tooltips for form components and table headers are not rendered.
 - Extensive usage of tooltips may impact runtime performance.
-
-

6.10 Displaying Output from a Managed Bean in an ADF Component

You can configure an ADF component to display output from a managed bean in your Fusion web application. Information about how to use managed beans in a Fusion web application can be found in the "Using a Managed Bean in a Fusion Web Application" section of the *Developing Fusion Web Applications with Oracle Application Development Framework*. You reference a managed bean in an integrated Excel workbook through an EL expression. Add a method action binding to the page definition file you associate

with the Excel worksheet to retrieve the value of the managed bean and assign it to an attribute binding. Use an EL expression to retrieve the value of the attribute binding at runtime.

6.10.1 How to Display Output from a Managed Bean

You write an EL expression for a property that supports EL expressions (for example, the `Label` property).

Before you begin:

It may be helpful to have an understanding of managed beans. For more information, see [Section 6.10, "Displaying Output from a Managed Bean in an ADF Component."](#)

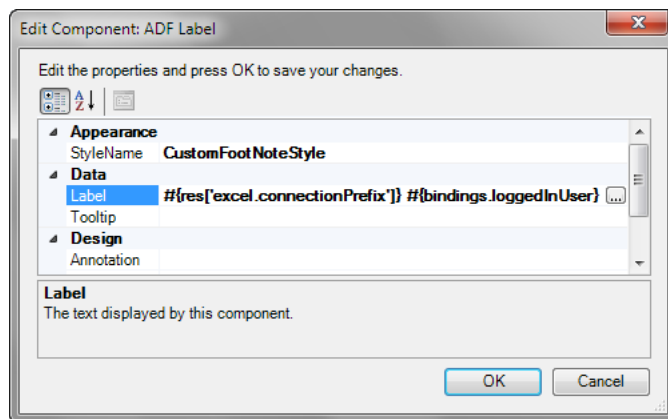
You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 6.1.2, "Additional Functionality for ADF Desktop Integration Form-Type Components."](#)

To display output from a managed bean:

1. Open the integrated Excel workbook.
2. Select the ADF component to display the output from the managed bean, and open its property inspector.

[Figure 6–20](#) shows an example where an ADF Label component is configured to display the output from an attribute binding that has its value populated by an action binding.

Figure 6–20 ADF Label Component That Displays Output from a Managed Bean at Runtime



3. Write an EL expression that gets the output from a managed bean at runtime.

The example in [Figure 6–20](#) shows an EL expression that retrieves the value of a string key (`excel.connectionPrefix`) from the `res` resource bundle and the value of the `loggedInUser` attribute binding. This attribute binding references the output from the managed bean.

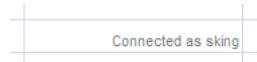
4. Click **OK**.

6.10.2 What Happens at Runtime: How an ADF Component Displays Output from a Managed Bean

The method action binding retrieves values from the managed bean and populates the attribute binding. The EL expression that you write retrieves the value from the attribute binding and displays it to the end user through the ADF component that you configured to display output. For example, the ADF Label component shown in design mode in [Figure 6–21](#) displays a string similar to the following at runtime:

Connected as sking

Figure 6–21 Output from a Managed Bean at Runtime



In [Figure 6–21](#), sking is the user name of the user that is logged on to the Fusion web application through the integrated Excel workbook.

6.11 Displaying Concatenated or Calculated Data in Components

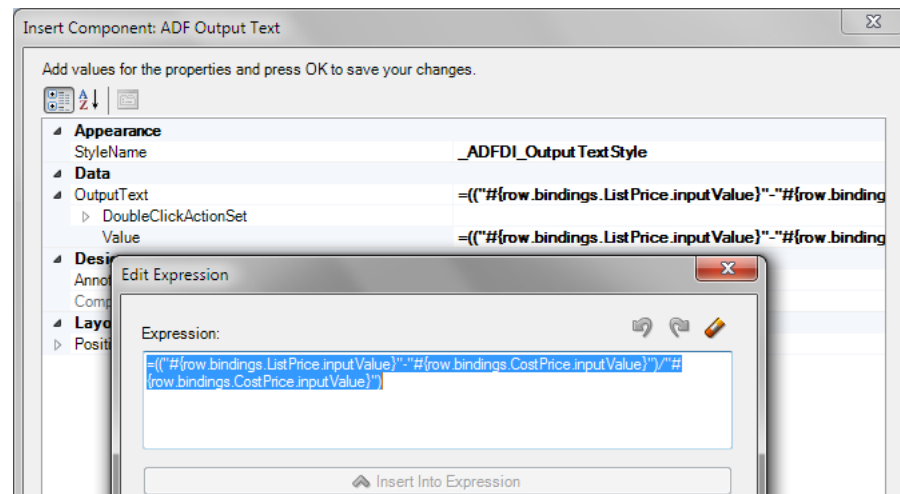
The ADF Desktop Integration module supports EL expressions within components that allow a single component to display data that is based on a calculation or concatenation of multiple binding expressions.

6.11.1 How to Configure a Component to Display Calculated Data

You write an EL expression for the Value property of an Input Text or Output Text component.

[Figure 6–22](#) shows an EL expression example where an ADF Output Text component is configured to display the margin between two fields: List Price and Cost Price.

Figure 6–22 ADF Output Text Component Displaying Calculated Data



Before you begin:

It may be helpful to have an understanding of how to display concatenated or calculated data in ADF components. For more information, see [Section 6.11](#), "Displaying Concatenated or Calculated Data in Components."

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 6.1.2, "Additional Functionality for ADF Desktop Integration Form-Type Components."](#)

To create an EL expression to display calculated data

1. Open the integrated Excel workbook.
2. Select the ADF Input Text or ADF Output Text component to display calculated data.
3. Open the property inspector and click the browse (...) icon of the Value property.
4. Write an EL expression that gets the output from two, or more, expressions.

[Example 6–1](#) shows an EL expression that calculates the difference between the values of two fields, List Price and Cost Price, and then divides it with value of Cost Price column to generate a margin.

Example 6–1 An EL Expression for Calculated Data

```
=((("#{row.bindings.ListPrice.inputValue}" - "#{row.bindings.CostPrice.inputValue}") /
 "#{row.bindings.CostPrice.inputValue} ")
```

5. Click OK.

For more information about EL expressions, see [Appendix B, "ADF Desktop Integration EL Expressions."](#)

Note: If the Value property of an ADF Input Text component contains a formula, the ADF Input Text component becomes read-only at runtime regardless of the value of the ReadOnly property.

6.12 Using Navigation Buttons

You can create navigation buttons (Next, Previous, First, and Last) to navigate from one record to another as shown in [Figure 6–23](#). If the end user changes a record's data before navigating to another record, you can choose to save those changes or ignore them.

Figure 6–23 Navigation Buttons in an Integrated Excel Workbook

The screenshot shows a web form titled "Edit Summit Warehouses". The form contains several input fields for warehouse information: Warehouse Id (201), Manager (Roberta Menchu), Region (South America), Country (Brazil), Address (68 Via Centrale), City (Sao Paulo), State, Zip Code, and Phone. On the right side of the form, there is a "Worksheet Errors" button and four navigation buttons: "First", "Next", "Previous", and "Last".

To save changes before navigating to another record, define the action sets of the button in the following order:

1. Worksheet.UpSync
2. Commit

3. Navigation action (for example, Next)
4. `Worksheet.DownSync`

Note: If you omit the `Commit` action from the action set, any pending changes to multiple records are lost when the end user's web application session ends.

To ignore changes before navigating to another record, define the action sets of the button in the following order:

1. Navigation action (for example, Next)
2. `Worksheet.DownSync`

Note: If you define button actions to ignore changes, then it is the end user's responsibility to save changes before navigating to another record.

Working with ADF Desktop Integration Table-Type Components

This chapter describes the table-type components that ADF Desktop Integration provides, how to configure and use them, how to download data from Fusion web application, how to insert, update, and delete data rows from the table-type components in the integrated Excel workbook, how to track the changes, how to configure special columns in the table-type components, and other tasks that you can do with table-type components.

This chapter includes the following sections:

- [Section 7.1, "About ADF Desktop Integration Table-Type Components"](#)
- [Section 7.2, "Page Definition Requirements for an ADF Table Component"](#)
- [Section 7.3, "Inserting ADF Table Component into Excel Worksheet"](#)
- [Section 7.4, "Configuring Oracle ADF Component to Download Data to an ADF Table Component"](#)
- [Section 7.5, "Configuring a Worksheet to Download Data as Pending Insert Rows in an ADF Table component"](#)
- [Section 7.6, "Configuring an ADF Table Component to Update Existing Data"](#)
- [Section 7.7, "Configuring an ADF Table Component to Insert Data"](#)
- [Section 7.8, "Configuring an ADF Component to Upload Changes from an ADF Table Component"](#)
- [Section 7.9, "Configuring an ADF Table Component to Upload Changes Using UploadAllOrNothing Action"](#)
- [Section 7.10, "Configuring an ADF Table Component to Delete Rows in the Fusion Web Application"](#)
- [Section 7.11, "Batch Processing in an ADF Table Component"](#)
- [Section 7.12, "Special Columns in the ADF Table Component"](#)
- [Section 7.13, "Configuring ADF Table Component Key Column"](#)
- [Section 7.14, "Creating a List of Values in an ADF Table Component Column"](#)
- [Section 7.15, "Adding a ModelDrivenColumnComponent Subcomponent to Your ADF Table Component"](#)
- [Section 7.16, "Adding a Dynamic Column to Your ADF Table Component"](#)
- [Section 7.17, "Configuring Column Widths in an ADF Table Component"](#)

- [Section 7.18, "Displaying Tooltips in ADF Desktop Integration Table-Type Components"](#)
- [Section 7.19, "Creating an ADF Read-Only Table Component"](#)
- [Section 7.20, "Limiting the Number of Rows Your Table-Type Component Downloads"](#)
- [Section 7.21, "Clearing the Values of Cached Attributes in an ADF Table Component"](#)
- [Section 7.22, "Tracking Changes in an ADF Table Component"](#)
- [Section 7.23, "Evaluating EL Expression for ReadOnly Properties"](#)
- [Section 7.24, "Using Explicit Worksheet Setup Action"](#)

7.1 About ADF Desktop Integration Table-Type Components

ADF Desktop Integration provides the following table-type components to display structured data:

- ADF Table component
- ADF Read-only Table component

The ADF Table and ADF Read-only Table components provide end users with the functionality to download and upload rows of data. The ADF Table component also enables end users to edit or delete downloaded data, or to insert new rows of data.

[Figure 7-1](#) shows the ADF Table and the ADF Read-only Table components.

Figure 7-1 ADF Desktop Integration Table-Type Components

Changed	Flagged	Status	Name	Credit Rating	Phone	Address	City	State	Zip Code	Country	Sales Rep.	Comments
			Unisports	Excellent	55-2066101	72 Via Bahia	Sao Paulo			Brazil	Henry Giljum	Customer usually
			Simms Athletics	Poor	81-20101	Takashi	Osaka			Japan	Mai Nguyen	Customer should
			Delhi Sports	Good	91-10351	Chanakya	New Delhi			India	Mai Nguyen	Customer specializ
			Womansport	Fair	1-206-104-0103	281 King Street	Seattle	Washington	98101	USA	Colin Magee	
			Kam's Sporting Goods	Excellent	852-3692888	95 Queensway	Hong Kong			Hong Kong	Andre Dumas	

ADF Table

Name	Address	Phone	City	State	Country	Zip Code	Sales Rep.	Credit Rating	Comments
Unisports	72 Via Bahia	55-2066101	Sao Paulo			1		12	1 Customer usually
Simms Athletics	Takashi	81-20101	Osaka			2		14	4 Customer should
Delhi Sports	Chanakya	91-10351	New Delhi			3		14	2 Customer specializ
Womansport	281 King Street	1-206-104-0103	Seattle	Washington		4	98101	11	3
Kam's Sporting Goods	95 Queensway	852-3692888	Hong Kong			5		15	1

ADF Read-only Table

Each ADF Table component contains a **Key** column. Do not remove the **Key** column, as it contains important information that is used by ADF Desktop Integration for the proper functioning of the table. Removal of the **Key** column, or any modification in the **Key** column cell, results in errors and data corruption. For more information about the **Key** column, see [Section 7.13, "Configuring ADF Table Component Key Column."](#)

The other ADF Desktop Integration components that you can use with these table-type components are described in [Chapter 6, "Working with ADF Desktop Integration Form-Type Components."](#)

7.1.1 ADF Desktop Integration Table-Type Components Use Cases and Examples

Tables are used to display the structured information. For example, [Figure 7–2](#) shows an ADF Table component of Summit sample application for ADF Desktop Integration with data downloaded from the respective Fusion web application.

Figure 7–2 ADF Table Component with Downloaded Data

Changed	Flagged	Status	Name	Credit Rating	Phone	Address	City	State	Zip Code	Country	Sales Rep.	Comments	Key
			Unisports	Excellent	55-2066101	72 Via Bahia	Sao Paolo			Brazil	Giljum	Customer usually orders large	
			Simms Athletics	Poor	81-20101	Takashi	Osaka			Japan	Nguyen	Customer should always pay by	
			Delhi Sports	Good	91-10351	Chanakya	New Delhi			India	Nguyen	Customer specializes in baseb	
			Womansport	Fair	1-206-104-0103	281 King Street	Seattle	Washington	98101	USA	Magee		
			Kam's Sporting Goods	Excellent	852-3692888	95 Queensway	Hong Kong			Hong Kong	Dumas		

7.1.2 Additional Functionality of Table-Type Components

After you have added a table component to your integrated Excel workbook, you may find that you need to add additional functionality to configure your table. Following are links to other functionalities that table components can use.

- **Change Tracking:** You can enable the table component for tracking changes. For more information, see [Section 7.22, "Tracking Changes in an ADF Table Component."](#)
- **Web Page Picker:** You configure a Dialog action to display a web page from the Fusion web application. For more information, see [Section 8.4, "Displaying Web Pages from a Fusion Web Application."](#)
- **Dependent List of Values:** You can add dependent list of values components in your table component. For more information, see [Section 8.8, "Creating Dependent Lists of Values in an Integrated Excel Workbook."](#)
- **Styles:** You can configure the display of your form-type components using several predefined Excel styles. For more information, see [Section 9.2, "Working with Styles."](#)
- **EL Expressions:** You can use EL expressions with table-type components. For more information, see [Appendix B, "ADF Desktop Integration EL Expressions."](#)

7.2 Page Definition Requirements for an ADF Table Component

The ADF Table component is one of the Oracle ADF components that ADF Desktop Integration exposes. It appears in the components palette of the ADF Desktop Integration Designer task pane and, after inserted into an Excel worksheet, allows the following operations:

- Read-only
- Insert-only
- Update-only
- Insert and update

Review the following sections for information about page definition file requirements specific to an ADF Table component.

Before you can configure an ADF Table component to provide data-entry functionality to your end users, you must configure the underlying page definition file for the Excel worksheet with ADF bindings. For general information about the page definition file

requirements for an integrated Excel workbook, see [Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook."](#)

Expose the following control bindings when you create a page definition file for authoring an ADF Table component:

- Tree binding that exposes the desired attribute bindings. Note that ADF Desktop Integration only supports `Scrollable` access mode for a `ViewObject`. The other access modes are not supported.
- Method action bindings and action bindings if you intend to configure values for the ADF Table component's `RowActions` and `BatchOptions` groups of properties. Examples of procedures where you set values for these groups of properties include:
 - [Section 7.3, "Inserting ADF Table Component into Excel Worksheet"](#)
 - [Section 7.7, "Configuring an ADF Table Component to Insert Data"](#)
 - [Section 7.5, "Configuring a Worksheet to Download Data as Pending Insert Rows in an ADF Table component"](#)

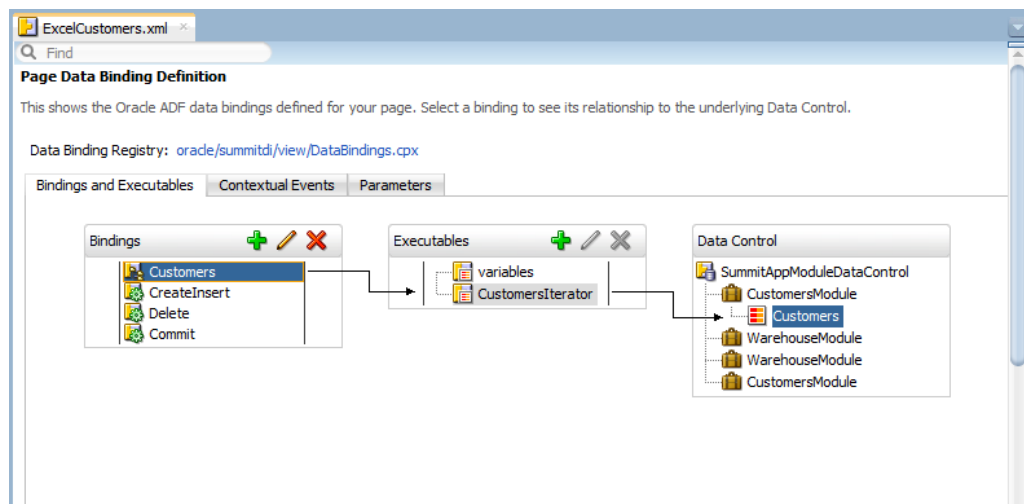
Note: The previous list is not exhaustive.

- (Optional) Update record action binding.

Note: As Excel displays a flat list of bindings and iterators are not displayed, use descriptive names for the attributes of different iterators.

[Figure 7–3](#) shows the bindings that the `ExcelCustomers.xml` page definition file includes. This page definition file can support the use of an ADF Table component in the Excel worksheet that it is associated with.

Figure 7–3 ADF Bindings Supporting Use of an ADF Table Component



7.3 Inserting ADF Table Component into Excel Worksheet

After you have configured a page definition file correctly, you can insert the ADF Table component into the worksheet and configure its properties to achieve the functionality you want. The ADF Table component enables you to download, edit, and upload rows of data.

7.3.1 How to Insert ADF Table Component

You can insert the ADF Table component from the ADF Desktop Integration Designer task pane.

Before you begin:

It may be helpful to have an understanding of ADF Table component. For more information, see [Section 7.3, "Inserting ADF Table Component into Excel Worksheet."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

To insert ADF Table component into Excel worksheet:

1. Open the integrated Excel workbook.
2. Select the cell in the Excel worksheet into which you want to insert the ADF Table component.

When inserting an ADF Table component, you must ensure that the data of two tables does not overlap at runtime, and the selected cell is not a merged cell.

3. In the bindings palette of the ADF Desktop Integration Designer task pane, select the tree binding to use and click **Insert Binding**.

Based on your selection, the Select Component dialog or the Insert Component dialog appears.

4. In the dialog that appears, select **ADF Table** and click **OK**.

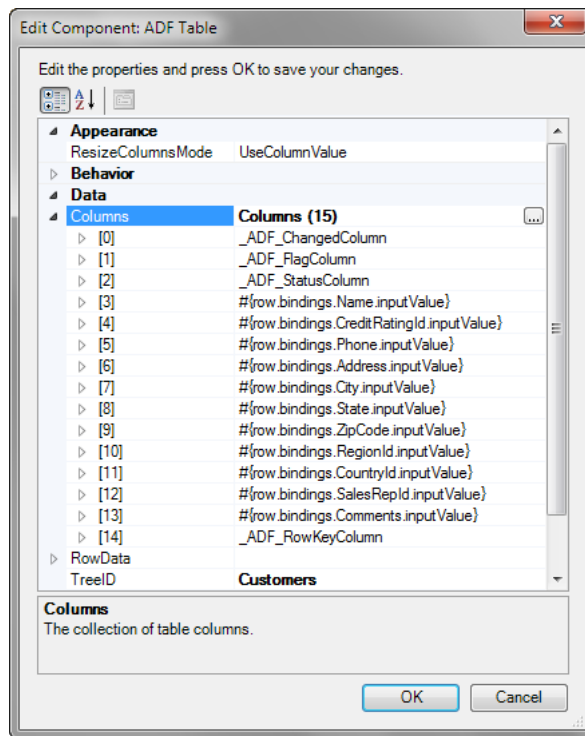
Notes:

- You can also insert an ADF Table component by using the components palette or the **Oracle ADF** tab. Select **ADF Table** and click **Insert Component**. Alternatively, in the **Oracle ADF** tab, select **ADF Table** from the **Insert Component** dropdown list. If you use either the components palette or the **Oracle ADF** tab to create the table component, you would have to add each column manually to appear in the component at runtime.
- You cannot insert an ADF Table component in a merged cell.
- When you insert an ADF Table component using Insert Binding, then by default, InputText is defined as the subcomponent type for all columns.

If you want a column to have a different subcomponent type, open the ADF Table property inspector (select any cell of the ADF Table component and click the **Edit Properties** button in the **Oracle ADF** tab), click the browse (...) icon of the **Columns** property. In the Edit Columns dialog, select the column, and click the browse (...) icon of the **UpdateComponent** property. In the Select Component dialog, select the desired subcomponent type, verify the binding and other properties, and click **OK**.

5. Configure properties for the ADF Table component using the property inspector shown in [Figure 7-4](#).

Figure 7-4 ADF Table Property Inspector



- Specify a binding expression for the attribute that uniquely identifies each row in the iterator associated with the tree binding. The `UniqueAttribute` property may be left blank if the binding's iterator supports row keys.
- Configure the `BatchOptions` properties of the ADF Table component as described in [Table 7-1](#).

Table 7-1 BatchOptions Properties of the ADF Table Component

Set this property to...	This value...
<code>CommitBatchActionID</code>	The <code>Commit</code> action binding that the page definition file exposes.

- Optionally, configure the `RowLimit` group of properties to determine what number of rows the ADF Table component can download.

For more information, see [Section 7.20, "Limiting the Number of Rows Your Table-Type Component Downloads."](#)

- Click **OK**.

[Figure 7-5](#) shows the ADF Table component in `EditCustomers-DT.xlsx` in design mode.

Figure 7-5 ADF Table Component in Design Mode

[Figure 7-6](#) shows the ADF Table component in `EditCustomers-DT.xlsx` at runtime.

Figure 7-6 ADF Table Component at Runtime

Changed	Flagged	Status	Name	Credit Rating	Phone	Address	City	State	Zip Code	Country	Sales Rep.	Comments	Key
			Unisports	Excellent	55-2066101	72 Via Bahia	Sao Paolo			Brazil	Giljum	Customer us	
			Simms Athletics	Poor	81-20101	Takashi	Osaka			Japan	Nguyen	Customer sh	
			Delhi Sports	Good	91-10351	Chanakya	New Delhi			India	Nguyen	Customer sp	
			Womansport	Fair	1-206-104-0103	281 King Street	Seattle	Washington	98101	USA	Magee		
			Kam's Sporting Goods	Excellent	852-3692888	95 Queensway	Hong Kong			Hong Kong	Dumas		
			Sportique	Excellent	33-2257201	4 Rue Chabaud	Cannes			France	Dumas	Customer sp	

For more information about the properties that you can set for the ADF Table component, see [Section A.12, "ADF Table Component Properties and Actions."](#)

To remove the table component, use the `Delete` ribbon command. For more information, see [Section 5.14, "Removing ADF Desktop Integration Components."](#)

7.3.2 How to Add a Column in an ADF Table Component

After inserting an ADF Table component in the worksheet of your integrated Excel workbook, you may want to add a column that is not available in the tree binding. For example, you may want to add a column that displays values calculated by an Excel formula.

Before you begin:

It may be helpful to have an understanding of ADF Table component. For more information, see [Section 7.3, "Inserting ADF Table Component into Excel Worksheet."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

To add a column in an ADF Table component:

1. Open the integrated Excel workbook.
2. Select the cell in the Excel worksheet that references the ADF Table component and click the **Edit Properties** button in the **Oracle ADF** tab.
3. In the Edit Component: ADF Table dialog, click the browse (...) icon of the **Columns** property.
The Edit Columns dialog appears, listing all the columns of the selected ADF Table component.
4. Click **Add** to add a new column. The new column is inserted at the end of the Members list. To move the column to a specific position, select the column and use the **Up** and **Down** arrow keys.
5. Configure the new column's properties in the right pane of the dialog.
6. Click **OK**.

Note: If you have not moved the new column as described in Step 4, it appears as the last column of the table.

ADF Desktop Integration does not limit the number of columns you can add in an ADF Table component you can add as many columns as your version of Excel supports. However, it has been observed that a very wide table gives slow performance and poor user experience. If you experience the same, you must try reducing the number of columns of the table before diagnosing other reasons for slow performance.

7.4 Configuring Oracle ADF Component to Download Data to an ADF Table Component

After you add an ADF Table component to a worksheet, you configure it and the worksheet that hosts it, so that the ADF Table component downloads data from the Fusion web application. To achieve this, you configure an Oracle ADF component, such as a worksheet ribbon command, to invoke an action set. The action set that is invoked must include the ADF Table component `Download` action among the actions that it invokes.

The number of rows that an ADF Table or an ADF Read-only Table component contains expands or contracts based on the number of rows to download from a Fusion web application. You should not place anything to the left or right of a table-type component unless you want to replicate it when Excel inserts rows to accommodate the data that one of the table-type components downloads. You can place other components above or below a table-type component as they maintain their position relative to the table-type component at runtime. End users who want to insert new rows of data into an ADF Table component at runtime must insert full rows into the Excel worksheet that hosts the ADF Table component.

7.4.1 How to Configure an ADF Component to Download Data to an ADF Table Component

Configure an Oracle ADF component, a worksheet ribbon button, or a worksheet event to invoke an action set that, in turn, invokes the ADF Table component `Download` action.

Before you begin:

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

It may be helpful to have an understanding of how to configure ADF component to download data to an ADF Table data component. For more information, see [Section 7.4, "Configuring Oracle ADF Component to Download Data to an ADF Table Component."](#)

To configure an ADF component to download data to an ADF Table component:

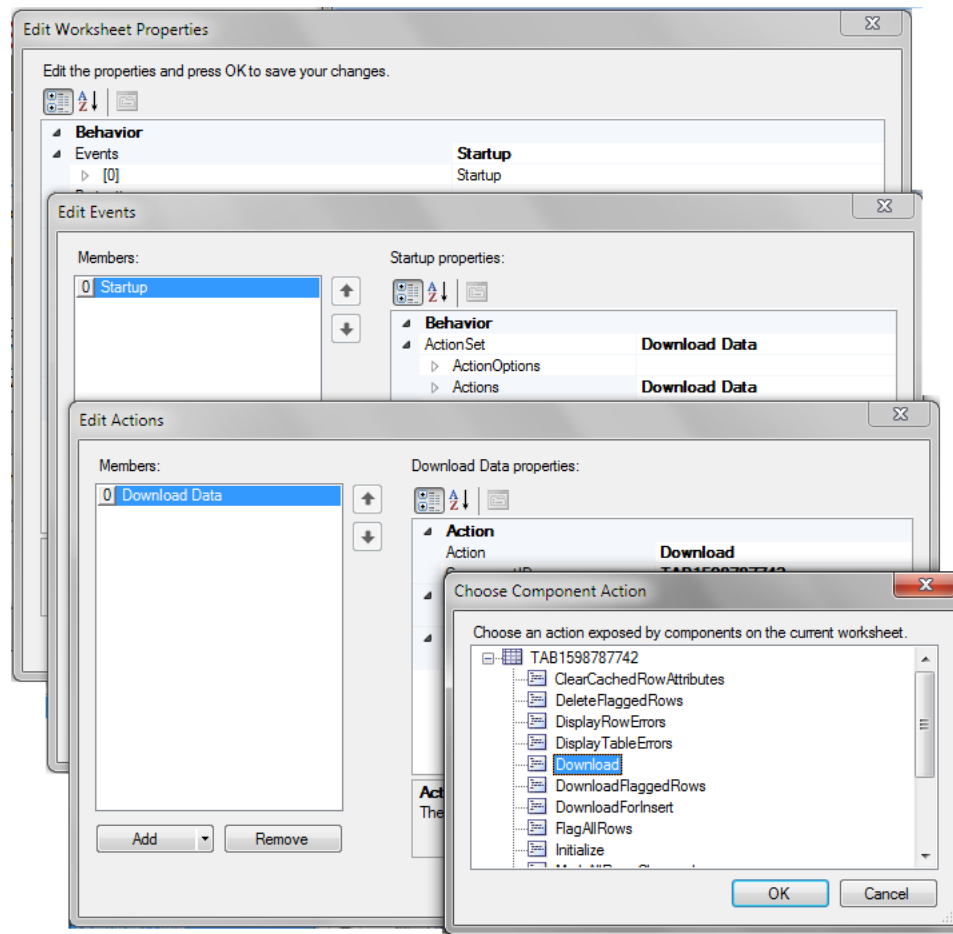
1. Open the integrated Excel workbook.
2. Click the **Worksheet Properties** button in the **Oracle ADF** tab, and add a ribbon command. For more information about adding a ribbon command in a worksheet, see [Section 8.3.1, "How to Define a Workbook Command Button for the Runtime Ribbon Tab."](#)

Note: Instead of adding a ribbon command, you can configure a worksheet event or an Oracle ADF component (a button, for example) to invoke the action set at runtime.

3. Open the Edit Action dialog to configure an action set. For more information about invoking action sets, see [Section 8.2, "Using Action Sets."](#)
4. Add the ADF Table component `Download` action to the list of actions that the action set invokes at runtime. Note that `Download` is a component action.

The ADF Table component `Download` action downloads the current state of the binding referenced by the ADF Table component `TreeID` property. To ensure that the state of this binding is up to date before download, add a query action that refreshes the binding before the action set invokes the ADF Table component `Download` action.

[Figure 7-7](#) shows the Edit Action dialog in the `EditCustomers-DT.xlsx` workbook where the action set invoked by the worksheet event `Startup` is configured.

Figure 7-7 Action Set Downloading Data to an ADF Table Component

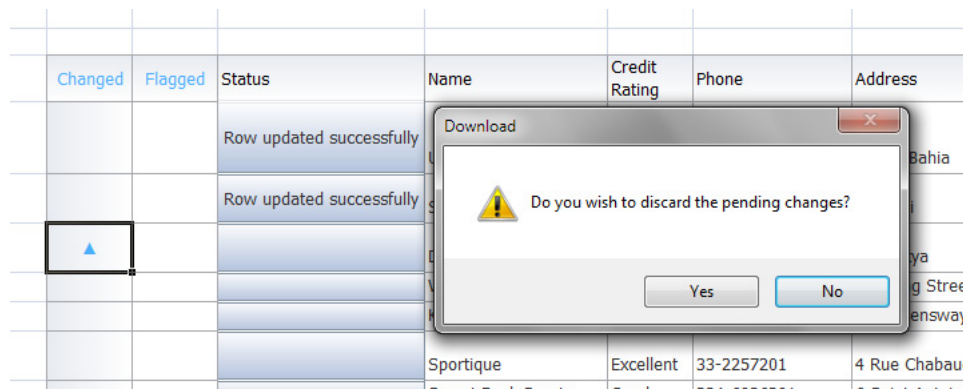
5. Click OK.

7.4.2 What Happens at Runtime: How the ADF Table Component Downloads Data

The end user invokes the action set that you configured. The action set invokes the list of actions specified in order. These include an action that invokes the `Download` action of the ADF Table component. When invoked, the `Download` action downloads all rows from the tree binding referenced by the ADF Table component `TreeID` property.

Make a note of following key points when the `Download` action is invoked at runtime:

- If there are any rows marked as changed when the `Download` action is invoked, the end user is prompted to confirm the action and to continue (see [Figure 7-8](#)). If the end user chooses **No**, the action and the action set are cancelled without error.
- All existing Excel rows are removed from the table in Excel.
- The status column is cleared of all messages.

Figure 7–8 Confirmation Prompt Before Downloading Data in ADF Table

The number of rows that the action downloads depends on the values set for the `RowLimit` group of properties in the ADF Table component. For more information, see [Section 7.20, "Limiting the Number of Rows Your Table-Type Component Downloads."](#)

Note: Any filter criteria that has been applied to the worksheet automatically gets cleared prior to the `Download` action.

7.5 Configuring a Worksheet to Download Data as Pending Insert Rows in an ADF Table component

A *Pending Insert* row is a worksheet table row with data that, on upload, is inserted as a new data row in the iterator. For example, if the end user creates a new row in the table by using the **Insert** option in the right click context menu, the new row is treated as a pending insert row and is inserted to the iterator when being uploaded.

A *Pending Update* row is a worksheet table row with data that, on upload, updates an existing data row in the iterator. For example, if the iterator of the tree binding contains some rows retrieved from the database and when these rows are downloaded to the ADF table, they are treated as pending update rows. If the end user makes changes to these rows and uploads them, the existing rows in the iterator are updated with new values from the ADF Table row.

In most cases, rows in the iterator of the tree binding are downloaded as pending update rows into the ADF Table. If you want some rows to be downloaded as pending inserts, you need to set the state of these rows to `STATUS_INITIALIZED`. For more information about how to set a row's state as `STATUS_INITIALIZED`, see the `setNewRowState` method in *Oracle Fusion Middleware Java API Reference for Oracle ADF Model*.

Note the following differences between pending insert rows and pending update rows:

- Pending insert rows are populated with the value of the EL expression for the insert component that is associated with each column in the ADF Table component, while pending update rows are populated with the value of the EL expression for the update component that is associated with each column in the ADF Table component.
- When evaluated for pending insert rows, the EL expression `#{components.componentID.currentRowMode}` returns `Insert`. In contrast, the same EL expression returns `Update` for pending update rows.

Note that the `componentID` part of the EL expression `{components.componentID.currentRowMode}` references the ID of the ADF Table component.

7.5.1 What Happens at Runtime: Download Action is Invoked

When the `Download` action is invoked, it examines the states of all rows in the iterator. Rows of state `STATUS_INITIALIZED` are downloaded as pending insert rows in the table, while rows of other states are downloaded as pending update rows.

7.5.2 Using `STATUS_INITIALIZED` Rows for Pending Inserts

You can use the `STATUS_INITIALIZED` rows to pre-populate values for some, or all, attributes of the pending insert rows. As a `STATUS_INITIALIZED` row is not validated, you can configure an action to populate the `STATUS_INITIALIZED` row partially and insert it into the iterator before the `Download` action is invoked. The `Download` action then treats this row as a pending insert row so that a new row, based on the pre-populated row, can be inserted.

Note that after uploading the pending insert, a `STATUS_NEW` row is inserted into the iterator, but the original `STATUS_INITIALIZED` row is not removed. If you want, you can configure another action to remove `STATUS_INITIALIZED` rows. For example, you can configure an action set with the following actions:

1. Create `STATUS_INITIALIZED` rows
2. Run `Download` action
3. Clean up `STATUS_INITIALIZED` rows

Notes:

- If `InsertRowEnabled` is set to `False`, any changes to the downloaded `STATUS_INITIALIZED` rows are ignored when upload is performed.
 - To avoid empty `STATUS_INITIALIZED` rows if you are using an `UpdateOnly` table, remove the custom method that inserts `STATUS_INITIALIZED` rows from the `Download` action.
-
-

7.5.3 What You May Need to Know About `DownloadForInsert` Action

ADF Desktop Integration also supports a table action called `DownloadForInsert`. `DownloadForInsert` is an obsolete action and can be replaced with the `Download` action. `DownloadForInsert` continues to work as it always has worked in previous releases. The key difference, with respect to `Download`, is that `DownloadForInsert` only considers rows in the iterator that are in the `STATUS_INITIALIZED` state.

7.6 Configuring an ADF Table Component to Update Existing Data

When you add the ADF Table component, by default, it allows end users to edit the existing data, but it does not allow them to add new data rows or to delete existing data rows.

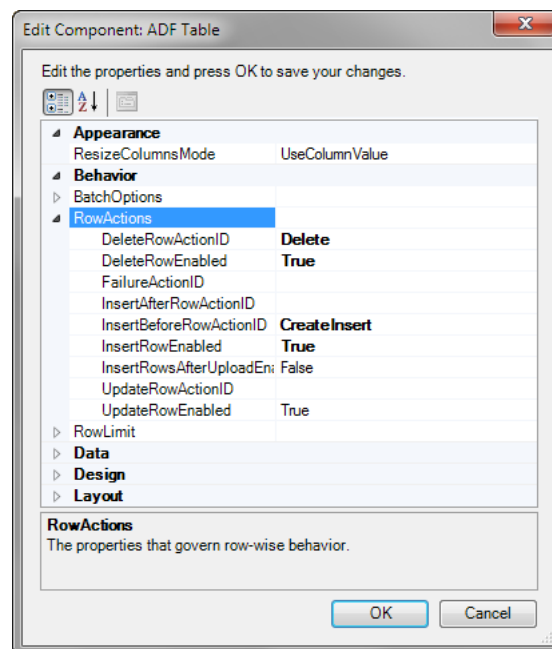
7.6.1 How to Configure an ADF Table Component to Update Data

If you want the end user to be able to edit existing data, but would like to restrict the addition or deletion of data rows, no additional configuration is required. Ensure that the ADF Table component `RowAction` properties are set, as described in [Table 7-2](#), and shown in [Figure 7-9](#).

Table 7-2 RowAction Properties of ADF Table Component

Property	Value
<code>InsertRowEnabled</code>	False
<code>DeleteRowEnabled</code>	False
<code>UpdateRowEnabled</code>	True

Figure 7-9 ADF Table RowActions Properties to Update Data



7.6.2 What Happens at Runtime: How the ADF Table Component Updates Data

When the end user changes data in a row, ADF Desktop Integration marks the row and an upward pointing triangle appears in a row of the `_ADF_ChangedColumn` column. After updating the existing data, the end user initiates the upload process to save the changes. For more information about the ADF Table component's upload process, see [Section 7.8, "Configuring an ADF Component to Upload Changes from an ADF Table Component."](#)

Excel uploads modified rows from the integrated workbook in batches rather than row by row. You can configure the size of batches and the actions an ADF Table component invokes when it uploads a batch. For more information about batch processing, see [Section 7.11, "Batch Processing in an ADF Table Component."](#)

For more information about the properties that you can set for the ADF Table component, see [Section A.12, "ADF Table Component Properties and Actions."](#)

Note: Any filter criteria that has been applied to the worksheet is automatically cleared prior to the upload action.

7.7 Configuring an ADF Table Component to Insert Data

The primary purpose of an ADF Table component is to provide end users with an interface where they can input or edit data which can then be uploaded to the database that serves your Fusion web application. For this to happen, you must expose methods on data controls, create action bindings in your page definition file, and set properties for the ADF Table component that an Excel worksheet hosts. Note that a full Excel row must be inserted for this functionality to work correctly.

7.7.1 How to Configure an ADF Table Component to Insert Data Using a View Object's Operations

If you want the changes that the end user makes in an ADF Table component to be committed invoking the ADF Table component's `Upload` action, you must configure some of the ADF Table component's properties.

Before you begin:

It may be helpful to have an understanding of how to configure ADF Table component to insert data. For more information, see [Section 7.7, "Configuring an ADF Table Component to Insert Data."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

To configure an ADF Table component to insert data using a view object's operations:

1. Open the project in JDeveloper.
2. If not present, add a `CreateInsert` and a `Commit` action binding to the page definition file that is associated with the Excel worksheet that hosts the ADF Table component.

For more information, see [Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook"](#) and [Section 7.2, "Page Definition Requirements for an ADF Table Component."](#)

3. Open the integrated Excel workbook.
4. Select the cell in the Excel worksheet that references the ADF Table component and click the **Edit Properties** button in the **Oracle ADF** tab.
5. In the Edit Component: ADF Table dialog, configure the `RowActions` properties of the ADF Table component as described in the [Table 7-3](#):

Table 7-3 *RowActions properties of ADF Table component*

Set this property to...	This value...
<code>InsertRowEnabled</code>	<code>True</code>
<code>InsertBeforeRowActionID</code>	The <code>CreateInsert</code> action binding that the page definition file exposes.

Table 7-3 (Cont.) RowActions properties of ADF Table component

Set this property to...	This value...
InsertRowsAfterUploadEnabled	True, to upload the inserted rows again regardless of whether they have been previously uploaded. By default, this property is set to False. The property is ignored if InsertRowEnabled is set to False.

6. Configure the BatchOptions properties of the ADF Table component as described in the [Table 7-4](#).

Table 7-4 BatchOptions Properties of the ADF Table Component

Set this property to...	This value...
CommitBatchActionID	The Commit action binding that the page definition file exposes.

7. Configure the Columns property of the ADF Table component as described in the [Table 7-5](#).

Table 7-5 Columns property of ADF Table component

Set this property to...	This value...
InsertUsesUpdate	True
UpdateComponent	<ul style="list-style-type: none"> Set the Value field of the UpdateComponent property to the update attribute from the page definition file. For example, <code>{row.bindings.ProductId.inputValue}</code>. Verify that ReadOnly property of UpdateComponent is set appropriately. This property only appears if you selected InputText or TreeNodeList as the subcomponent to associate with the column. Set ReadOnly to False if you do want users to edit the values in the column, set to True otherwise. For more information about the components that you can use as a subcomponent, see Chapter 6, "Working with ADF Desktop Integration Form-Type Components."
ID	Set a value in this field that uniquely identifies the column in the ADF Table component's list of columns. A value for this property is required. The ADF Table component generates an initial value that you need not modify.
CellStyleName	Set this property to a style defined in the workbook or to an EL expression that applies a style to the cells in the column at runtime. For more information about styles, see Chapter 9, "Configuring the Appearance of an Integrated Excel Workbook."
HeaderLabel	Set this property to a label or to an EL expression that evaluates to a label which is rendered in the column header at runtime. For more information about labels, see Section 9.4, "Using Labels in an Integrated Excel Workbook."
HeaderStyleName	Set this property to a style defined in the workbook or to an EL expression that applies a style to the column's header cell at runtime. For more information about styles, see Chapter 9, "Configuring the Appearance of an Integrated Excel Workbook."

8. Repeat Step 7 for each column that contains data to commit during invocation of the Upload action.

For information about ADF Table component properties, see [Section A.12, "ADF Table Component Properties and Actions."](#)

Notes:

- If you are using a polymorphic view object and want to insert a new row, the default `CreateInsert` action binding is not sufficient. You must create a custom method that also sets the discriminator value in the newly created row.

While creating the custom method, you must expose the custom method as an action binding in the page definition file. The action binding must be specified as the `InsertBeforeActionId` rather than `CreateInsert`.

- If the `InsertRowsAfterUploadEnabled` property is set to `False` and the end user tries to upload the inserted rows again, an error message in the status column is displayed indicating that the row cannot be inserted more than once.
-

7.8 Configuring an ADF Component to Upload Changes from an ADF Table Component

You configure the ADF Table component and the worksheet that hosts it so that end user can upload changes they make to data in the ADF Table component to the Fusion web application. To configure this functionality, you decide what user gesture or worksheet event invokes the action set that invokes the ADF Table component's `Upload` action.

The `Upload` action commits all successful rows even when some rows have failures. Use the `UploadAllOrNothing` action instead if you want no row changes to get committed if one, or more, row failures occur (see [Section 7.9, "Configuring an ADF Table Component to Upload Changes Using UploadAllOrNothing Action"](#)). To provide upload options to end users in a web page from the Fusion web application that differ from the default upload dialog, you must specify a `Dialog` action in the action set before the action that invokes the ADF Table Component's `Upload` action. For more information, see [Section 7.8.5, "How to Create a Custom Upload Dialog."](#)

Note: In a master-detail relationship, ADF Desktop Integration does not support editing of the `ViewLink` source attributes, as the selections in the child view object would change as a result. To prevent any accidental editing, define the `ViewLink` source attributes to be read-only, or use a model configuration that does not include a view link between master and detail.

7.8.1 How to Configure an ADF Component to Upload Data from an ADF Table Component

Configure an ADF component, such as a worksheet ribbon command, to invoke an action set that, in turn, invokes the ADF Table component `Upload` action.

Before you begin:

It may be helpful to have an understanding of how to configure ADF component to upload data from an ADF Table data component. For more information, see

Section 7.8, "Configuring an ADF Component to Upload Changes from an ADF Table Component."

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

To configure an ADF component to upload changed data from an ADF Table component:

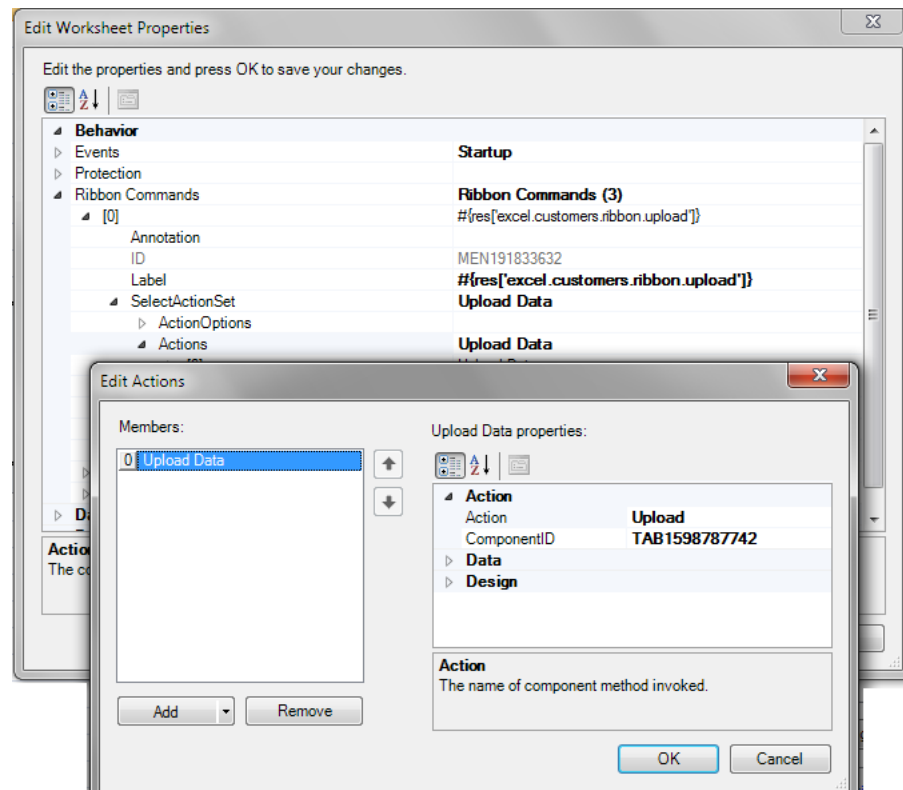
1. Open the integrated Excel workbook.
2. Open the Edit Action dialog to configure the action set that invokes the ADF Table component Upload action.

For more information about action sets, see [Section 8.2, "Using Action Sets."](#)

3. Add the ADF Table component Upload action to the list of actions that the action set invokes at runtime.

[Figure 7–10](#) shows the Edit Actions dialog in the EditCustomers-DT.xlsx workbook, where the action set invoked by the ribbon command labeled **Upload** at runtime is configured.

Figure 7–10 Action Set Uploading Data from an ADF Table Component



4. Click OK.

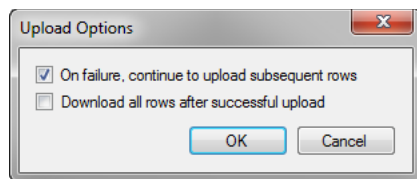
Note: The action set does not include a call to a commit-type action as the ADF Table component's batch options already include calls to Commit. For more information, see [Section 7.11.1, "How to Configure Batch Options for an ADF Table Component."](#)

7.8.2 What Happens at Runtime: How the ADF Table Component Uploads Data

At runtime, the end user invokes the action set through whatever mechanism you configured (ADF component, worksheet ribbon button, worksheet event). This triggers the following sequence of events:

1. If the ADF Table component contains dynamic columns, ADF Desktop Integration verifies whether the dynamic columns that were expanded the last time the ADF Table component's `Download` action was invoked are still present in the Fusion web application. If the columns are not present, ADF Desktop Integration prompts the end user to determine whether to continue upload process. If the end user decides not to continue, ADF Desktop Integration returns an abort code to the executing action set.
2. If the ADF Table component contains no pending changes to upload, the ADF Table component's `Upload` action returns a success code to the executing action set.
3. If you did not configure a custom upload dialog for the action set, as described in [Section 7.8.5, "How to Create a Custom Upload Dialog,"](#) ADF Desktop Integration presents the default upload dialog shown in [Figure 7-11](#).

Figure 7-11 Default Upload Dialog



If the end user clicks **Cancel**, ADF Desktop Integration returns an abort code to the executing action set. If the end user clicks **OK**, the action set continues executing with the options specified in the dialog for the upload operation.

4. The ADF Table component uploads modified rows in batches, rather than row by row. You can configure the batch options using the `BatchOptions` group of properties. For more information about batch options for the ADF Table component, see [Section 7.11, "Batch Processing in an ADF Table Component."](#)

Each row of a batch is processed in the following way, and the process continues until all changed rows of each batch are processed:

- a. For inserted rows, invoke the `InsertBeforeRowActionID` action, if specified.
 - b. Set attributes from the worksheet into the model, including any cached row attribute values.
 - c. For edited rows, invoke the `UpdateRowActionID` action; and for inserted rows, invoke the `InsertAfterRowActionID` action, if specified.
 - d. For each uploaded row, displays a status message in the Status column. For more information, see [Section 8.2.5, "How to Display a Status Message While an Action Set Runs."](#)
 - e. For any row failure, it verifies the value of `AbortOnFail`. If `AbortOnFail` is set to `False`, it continues upload process, otherwise it stops uploading data and invokes the commit action.
5. While uploading data, the ADF Table component returns a success or failure code to the executing action set based on the following:

- If the ADF Table component uploads all batches successfully, it returns the success status to the executing action set. If the end user has selected the **Download all rows after successful upload** option in Step 3, the ADF Table component then downloads all rows from the Fusion web application.
- If the ADF Table component did not upload all batches successfully, the action set invokes the action specified by the `RowActions.FailureActionID` property, if an action is specified for this property. ADF Desktop Integration returns a failure code to the action set.

If you selected **On failure, continue to upload subsequent rows** in the Upload Options dialog of Step 3, the Upload action returns a success code to the action set even if some individual rows encountered validation failures.

Notes: ■ If an ADF Table component column's `ReadOnly` property evaluates to `True`, the ADF Table component's Upload action ignores changes in the column's cells.

For more information about an ADF Table component column's properties, see [Table A-13](#).

- When the Upload action is invoked on an ADF Table that has an Excel filter applied, Excel filter's criteria is cleared to show any hidden Excel worksheet rows, but the filter is not removed.
-

7.8.3 What Happens at Runtime: How the ReadOnly EL Expression Is Evaluated During Upload

At runtime, if an ADF Table component column's `ReadOnly` property evaluates to `True`, the ADF Table component's Upload action ignores all changes in the column's cells.

For more information about change tracking, see [Section 7.23, "Evaluating EL Expression for ReadOnly Properties."](#)

7.8.4 What Happens at Runtime: Upload Failure

When the ADF Table component starts uploading data, ADF Desktop Integration creates a `DataControlFrame` savepoint before initiating the upload process (once per batch of uploaded rows). In case of any failure, ADF Desktop Integration reverts back to the same savepoint, ensuring the integrity of the server-side state of the Fusion web application.

For each row in a batch of uploaded rows, ADF Desktop Integration does the following:

1. Invokes configured actions, applies row attribute value changes, and performs data validation.
2. In case of any error, reverts back to the savepoint state.

Note: A second iteration is performed, if required, to re-upload any successfully uploaded rows whose changes were reverted due to a subsequent upload error.

For more information about savepoints, see the "Using Trees to Display Master-Detail Objects" section in the *Developing Fusion Web Applications with Oracle Application Development Framework*.

7.8.5 How to Create a Custom Upload Dialog

You display a page from Fusion web application that offers end users different options to those presented in the default upload dialog. You add a `Dialog` action before the action that invokes the ADF Table component's `Upload` action in the action set.

Before you begin:

It may be helpful to have an understanding of how to configure ADF component to upload data from an ADF Table data component. For more information, see [Section 7.8, "Configuring an ADF Component to Upload Changes from an ADF Table Component."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

To create a custom upload dialog:

1. Create a page in the JDeveloper project where you develop the Fusion web application. For information on how to create this page, see [Section 8.4, "Displaying Web Pages from a Fusion Web Application."](#)
2. In addition to the `ADFdi_CloseWindow` element (for example, a span element) described in [Section 8.4, "Displaying Web Pages from a Fusion Web Application,"](#) the page that you create in Step 1 must include the elements described in [Table 7-6.](#)

Table 7-6 *Span Elements Required for Custom Upload*

Name	Description
<code>ADFdi_AbortUploadOnFailure</code>	If you set this element to <code>True</code> , the action set stops uploading if it encounters a failure. If the element references <code>False</code> , the action set attempts to upload all rows and indicates if each row succeeded or failed to upload.
<code>ADFdi_DownloadAfterUpload</code>	Set this element to <code>True</code> so the action set downloads data from the Fusion web application to the ADF Table component after the action set uploads modified data.

Note: The page you create must include both elements to prevent ADF Desktop Integration presenting the default upload dialog to end users.

3. Add a `Dialog` action to invoke the page you created in Step 1 before the action in the action set that invokes the ADF Table component's `Upload` action.

For more information about displaying pages from a Fusion web application, see [Section 8.4, "Displaying Web Pages from a Fusion Web Application."](#)

7.8.6 What Happens at Runtime: Custom Upload Dialog

When a custom dialog appears, the page from the Fusion web application that you configure the `Dialog` action in the action set to display appears instead of the default upload dialog.

Note: If there is no server connectivity when the end user tries to upload data, the end user gets an error when the `Dialog` action fails to find the custom upload page. ADF Desktop Integration does not revert to the standard dialog when server connectivity is not available.

For more information about displaying a page from the Fusion web application, see [Section 8.4, "Displaying Web Pages from a Fusion Web Application."](#) Otherwise, the runtime behavior of the action set that you configure to upload data is as described in [Section 7.8.2, "What Happens at Runtime: How the ADF Table Component Uploads Data."](#)

7.9 Configuring an ADF Table Component to Upload Changes Using UploadAllOrNothing Action

ADF Desktop Integration commits all row changes that are successfully uploaded during a `Table.Upload` operation, even when one or more rows has failures. For example, if 100 rows are uploaded and only three rows contain failures, 97 rows are still committed to the database. For more information, see [Section 7.8, "Configuring an ADF Component to Upload Changes from an ADF Table Component."](#)

Using the `UploadAllOrNothing` action, you can configure the upload process to commit all changed rows only if all rows are successfully uploaded. For example, if 100 rows are uploaded, and if any row fails, no rows are committed to the database.

7.9.1 How to Configure an ADF Component to use UploadAllOrNothing Action

Configure an ADF component, such as a worksheet ribbon command, to invoke an action set that, in turn, invokes the ADF Table component `UploadAllOrNothing` action.

Before you begin:

It may be helpful to have an understanding of how to configure ADF component to upload data from an ADF Table data component. For more information, see [Section 7.9, "Configuring an ADF Table Component to Upload Changes Using UploadAllOrNothing Action."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

To configure an ADF component to use UploadAllOrNothing action:

1. Open the integrated Excel workbook.
2. Open the Edit Action dialog to configure the action set that invokes the ADF Table component actions.

For more information about action sets, see [Section 8.2, "Using Action Sets."](#)

3. Add the ADF Table component `UploadAllOrNothing` action to the list of actions that the action set invokes at runtime.
4. Click OK.

7.9.2 What Happens at Runtime: UploadAllOrNothing Action is Invoked

If you have chosen the `UploadAllOrNothing` action, ADF Desktop Integration commits row changes only when all rows are uploaded successfully.

Note: The `UploadAllOrNothing` action uploads data in the same way as the `Upload` action. For more information about how data gets uploaded during `Upload` as well as `UploadAllOrNothing`, see [Section 7.8.2, "What Happens at Runtime: How the ADF Table Component Uploads Data."](#)

During the `UploadAllOrNothing` action, ADF Desktop Integration uploads all changed worksheet rows prior to invoking specified by `CommitBatchActionID`. If one, or more, row-level failures occur, the action specified by `FailureActionID` is invoked and the action specified by `CommitBatchActionID` is not invoked.

In the event of a failure, all changed column values remain unchanged. The status column indicates failure for all row-level failures, but remains empty for all rows without errors. When all rows succeed and are successfully committed, the changed column values are cleared and the status column for the uploaded rows reports success.

Notes:

- The `UploadAllOrNothing` action is only supported for `DataControls` that support database transactions.
 - If `CommitBatchActionID` is not configured and an action set contains the `UploadAllOrNothing` action, a validation error is reported.
-
-

7.9.3 Limiting the Amount of Changed Data That Can Be Uploaded With UploadAllOrNothing Action

Uploading a very large number of changed worksheet rows with the `UploadAllOrNothing` action could result in significant memory consumption on the application server. To prevent end users from uploading too much data during the `UploadAllOrNothing` action, set the `UploadAllOrNothing.ChangedDataLimit` servlet parameter (specified in Kb) to limit the total amount of changed data that can get uploaded. If no parameter value is specified, a default limit of 10,240 Kb is used.

If the total amount of changed data uploaded exceeds the `UploadAllOrNothing.ChangedDataLimit` value, an error message is reported to the end user, and the `UploadAllOrNothing` action is aborted. Note that the action specified by `Table.RowActions.FailureActionID` is invoked when the changed data limit is exceeded.

7.10 Configuring an ADF Table Component to Delete Rows in the Fusion Web Application

The ADF Table component exposes an action (`DeleteFlaggedRows`) that, when invoked, deletes the rows in the Fusion web application that correspond to the flagged rows in the ADF Table component. A *flagged row* in an ADF Table component is a row where the end user has double-clicked or typed a character in the cell of the `_ADF_FlagColumn` column as described in [Section 7.11, "Batch Processing in an ADF Table](#)

Component." The `_ADF_FlagColumn` column must be present in the ADF Table component to configure it to delete rows in the Fusion web application.

In addition, the page definition file that you associate with the worksheet that hosts the ADF Table component must expose a `Delete` action binding.

7.10.1 How to Configure an ADF Table Component to Delete Rows in the Fusion Web Application

To delete rows from an ADF Table component, you must add the `Delete` action binding to the page definition file, configure `RowActions` group of properties of the ADF Table component, and configure an action set to invoke the `DeleteFlaggedRows` action.

Before you begin:

It may be helpful to have an understanding of how to configure ADF Table component to delete data rows in Fusion web application. For more information, see [Section 7.10, "Configuring an ADF Table Component to Delete Rows in the Fusion Web Application."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

To configure an ADF Table component to delete rows in a Fusion web application:

1. Open your Fusion web application in JDeveloper.
2. If not present, add a `Delete` action binding to the page definition file that is associated with the Excel worksheet that hosts the ADF Table component.

For more information, see [Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook."](#)
3. Open the property inspector for the ADF Table component and set values for the `RowActions` group of properties as described in [Table 7-7](#).

Table 7-7 RowActions Properties of ADF Table component

Set this property...	To...
<code>DeleteRowActionID</code>	The <code>Delete</code> action binding that the page definition file exposes.
<code>DeleteRowEnabled</code>	True to enable the ADF Table component to delete rows in the Fusion web application. False is the default value.

For more information about ADF Table component properties, see [Section A.12, "ADF Table Component Properties and Actions."](#)

4. Click **OK**.
5. Open the integrated Excel workbook.
6. Open the Edit Action dialog to configure an action set for the Oracle ADF component, ribbon control, or worksheet event that the end user uses to invoke the action set at runtime.
7. Add the ADF Table component's `DeleteFlaggedRows` action to the list of actions that the action set invokes at runtime.

For more information about invoking action sets, see [Section 8.2, "Using Action Sets."](#)

8. Click OK.

Note: Add the `DeleteFlaggedRows` action in a separate action set and do not include it with other table actions like `Download` or `Upload`.

7.10.2 What Happens at Runtime: How the ADF Table Component Deletes Rows in a Fusion Web Application

The end user flags rows to delete, as described in [Section 7.11.2, "Row Flagging in an ADF Table Component."](#) The end user then invokes the action set. The following sequence of events occurs:

1. If specified, the action binding referenced by the `BatchOptions.StartBatchActionID` property is invoked.

Failures from this step are treated as errors. An error stops the action set invoking. It also returns the error condition to the action set. If an action binding is specified for the `ActionSet.FailureActionID` property, the action set invokes the specified action binding.

For more information about configuring batch options, see [Section 7.11, "Batch Processing in an ADF Table Component."](#)

2. The action set invokes the `Delete` action binding specified by `RowActions.DeleteRowActionID`.

Note: Rows inserted since the last invocation of the ADF Table component's `Download` action but not uploaded to the Fusion web application are ignored even if flagged for deletion.

3. If no errors occur during the invocation of the `Delete` action binding, a success message entry appears in the `_ADF_StatusColumn` column. If a failure occurs, the ADF Table component stops invocation of the `Delete` action binding and continues to Step 4.
4. If an action binding is specified for the `BatchOptions.CommitBatchActionID` property, the action set invokes it. If this step fails, the action set stops processing batches. If no failures occur, the action set processes the next batch by invoking the action binding specified by the `BatchOptions.StartBatchActionID` property, and so on until the action set processes all batches.
5. If the action set processes all batches successfully, it invokes the action binding specified by its `ActionOptions.SuccessActionID` property if an action binding is specified for this property. It then removes the rows deleted in the Fusion web application by invocation of the `Delete` action binding specified by `RowActions.DeleteRowActionID` from the worksheet and returns a success code to the action set.

If failures occur while the action set processes the batches, the action set invokes the action binding specified by its `ActionOptions.FailureActionID` property if an action binding is specified for this property. This action binding returns a failure code to the action set.

6. If an unexpected exception occurs while the action set invokes its actions, an error code is returned to the action set. All row-level errors are displayed in the Status column, and all batch-level errors can be tracked through `Table.errors`. For more information about error handling, see [Section 12.4, "Error Reporting in an Integrated Excel Workbook."](#)

Note: When the `DeleteFlaggedRows` action is invoked on an ADF Table that has an Excel filter applied, Excel filter's criteria is cleared to show any hidden Excel worksheet rows, but the filter is not removed.

7.11 Batch Processing in an ADF Table Component

The ADF Table component uploads modified rows from the Excel workbook in batches rather than row-by-row. You can configure batch option properties that determine the size of batches and what actions the ADF Table component invokes when it uploads a batch.

7.11.1 How to Configure Batch Options for an ADF Table Component

The ADF Table component has a group of properties (`BatchOptions`) that allow you to configure how the ADF Table component manages batches of rows. Information about these properties can be found in [Section A.12, "ADF Table Component Properties and Actions."](#)

Before you begin:

It may be helpful to have an understanding of batch processing in the ADF Table component. For more information, see [Section 7.11, "Batch Processing in an ADF Table Component."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

To configure batch options for an ADF Table component:

1. Open the integrated Excel workbook.
2. Select the cell in the Excel worksheet that references the ADF Table component, and then click the **Edit Properties** button in the **Oracle ADF** tab.
3. Set values for the `BatchOptions` group of properties in the property inspector that appears.

Table 7–8 *RowData.BatchOptions Properties*

Set this property...	To...
<code>BatchSize</code>	Specify how many rows to process before an ADF Table component action (<code>Upload</code> or <code>DeleteFlaggedRows</code>) invokes the action binding specified by <code>CommitBatchActionID</code> . Any value other than a positive integer results in all rows being processed in a single batch. The default value is 100 rows.
<code>CommitBatchActionID</code>	The action binding to invoke after the ADF Table component processes each batch. Typically, this is the <code>Commit</code> action binding.

Table 7–8 (Cont.) RowData.BatchOptions Properties

Set this property...	To...
LimitBatchSize	<p>True</p> <p>When <code>True</code>, the ADF Table component processes rows in batches determined by the value of <code>BatchSize</code>. When <code>False</code>, the ADF Table component uploads all modified rows in a single batch.</p> <p><code>True</code> is the default value.</p>
StartBatchActionID	Specify the action binding to invoke at the beginning of each batch.

4. Click **OK**.

Note that a failure at the entity-level is not considered a batch failure. A failure at the commit level (for example, a wrong value for a foreign key attribute) is considered a batch failure.

7.11.2 Row Flagging in an ADF Table Component

By default, the ADF Table component includes a column, `_ADF_FlagColumn`, that facilitates the selection of rows for flagged-row processing. Double-clicking a cell of the `_ADF_FlagColumn` column flags the corresponding row for processing by actions invoked by a component action.

When the end user double clicks a cell of the `_ADF_FlagColumn` column, a solid circle appears, or disappears, in the cell to indicate that the row is flagged, or not.

[Figure 7–12](#) shows an example of a flagged column.

Figure 7–12 Flagged Column in ADF Table Component

Note: By default, the solid circle character indicates a row flagged for flagged-row processing. However, any non-empty cell in a `_ADF_FlagColumn` column flags the corresponding row for flagged-row processing.

The following component actions can be invoked on flagged rows:

- `DeleteFlaggedRows`
- `DownloadFlaggedRows`

You can use the `FlagAllRows` component action to flag all rows, and the `UnflagAllRows` component action to unflag all rows of the ADF Table component.

Notes:

- The ADF Table component's `DownloadFlaggedRows` action does not support changes in table column structure after the last invocation of the `Download` or `DownloadForInsert` action. The table column structure usually changes if you are using dynamic columns, or if the table contains columns with complex expressions in the `Visible` property.
 - The `DownloadFlaggedRows` action is not applicable to inserted rows.
-
-

Use of these component actions is dependent on the appearance of the `_ADF_FlagColumn` column in the ADF Table component. If you remove the `_ADF_FlagColumn` column from the ADF Table component, you cannot invoke any of these component actions. For more information about these component actions, see [Section A.12.3, "ADF Table Component Actions."](#)

At runtime, the end user can invoke any of the previously listed component actions from an action set. The invoked component action processes all flagged rows. For example, it downloads or deletes all flagged rows. For more information about configuring an action set to invoke a component action, see [Section 8.2.2, "How to Invoke Component Actions in an Action Set."](#)

7.11.3 Troubleshooting Errors While Uploading Data

End users might encounter unexpected reports of errors under certain circumstances while uploading data from ADF Table components. After posting changes from a batch, ADF Desktop Integration runs the action specified by the `CommitBatchActionID`. Errors that occur during the commit action might continue to be reported on subsequent batch commit actions, even though those batches of records do not contain the error. This can happen when any pending model updates are not automatically reverted when the `CommitBatchActionID` action fails.

To avoid any such error, you must explicitly revert pending model updates that exist after a commit failure. For example, you could create a custom action for the `CommitBatchActionID` that first attempts to commit the pending model changes. However, if an exception occurs during commit, the custom method should first roll back the pending model changes, so that any subsequent batch commit attempts can succeed.

Note: It is important that the commit exception gets thrown again after rollback so that the commit errors are reported as expected on the client.

7.12 Special Columns in the ADF Table Component

By default, the ADF Table component includes some columns when you insert an ADF Table component in a worksheet. You can retain or remove these columns, if required. The following list describes the columns and the purpose they serve:

- `_ADF_ChangedColumn`

The cells in this column track changes to the rows in the ADF Table component. If a change has been made to data in a row of the ADF Table component since download or the last successful upload, a character that resembles an upward

pointing arrow appears in the corresponding cell of the `_ADF_ChangedColumn` column. This character toggles (appears or disappears) when a user double-clicks a cell in this column. [Figure 7–13](#) shows an example.

Figure 7–13 *Changed Column in an ADF Table Component*



Note: If the end user does not want the ADF Table component's Upload action to upload changes in the rows flagged by this column, the user must clear the entry that appears in the corresponding cell.

A confirmation dialog appears to end users when the ADF Table component's Download action is invoked, and one or more rows in this column are flagged as changed. The end user clicks **OK** to allow the Download action to run, or **Cancel** to stop the execution of the Download action.

- `_ADF_FlagColumn`

When the end user double-clicks a cell in this column, the corresponding row is flagged for flagged-row processing. A solid circle character appears to indicate that the row is flagged for flagged-row processing. For more information about the use of this column, see [Section 7.11.2, "Row Flagging in an ADF Table Component."](#)

A confirmation dialog appears to end users when the ADF Table component's DownloadFlaggedRows action is invoked, and one or more rows in `_ADF_ChangedColumn` and `_ADF_FlagColumn` are flagged. The end user clicks **OK** to allow the action to run or **Cancel** to stop the execution of the action.

Note: By default, the solid circle character indicates a row flagged for flagged-row processing. However, any non-empty cell in a `_ADF_FlagColumn` flags the corresponding row for flagged-row processing.

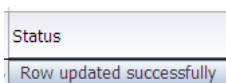
- `_ADF_StatusColumn`

This column reports the results of invocation of the following ADF Table component actions:

- DeleteFlaggedRows
- Upload

A message appears in the cell of the `_ADF_StatusColumn` to indicate the result of the invocation for the corresponding row. If the end user invokes a `DoubleClickActionSet` defined in an ADF Table column and an error occurs, the errors are also reported in the status column of the corresponding row. [Figure 7–14](#) shows an example of Status column message.

Figure 7–14 *Status Column in an ADF Table Component*



- `_ADF_RowKeyColumn`

This column, also referred to as the **Key** column, contains important information about the ADF Table component used by ADF Desktop Integration at runtime. The column appears both at runtime and design time. You can remove the column from the table at design time, but note that it automatically appears at runtime as the last column of the table.

For more information about the `_ADF_RowKeyColumn`, see [Section 7.13, "Configuring ADF Table Component Key Column."](#)

The ADF Table component treats the properties of the `_ADF_ChangedColumn`, `_ADF_FlagColumn`, `_ADF_RowKeyColumn`, and `_ADF_StatusColumn` columns differently from the properties of other columns that it references. It ignores the values set for properties such as `InsertComponent`, `InsertUsesUpdate`, and `UpdateComponent` unless it invokes the `DisplayRowErrors` action described in [Table A-14](#). It reads the values for properties related to style and appearance, for example `CellStyleName` and `HeaderStyleName`.

7.13 Configuring ADF Table Component Key Column

When you add ADF Table to your integrated Excel workbook, the **Key** column (column ID: `_ADF_RowKeyColumn`) appears automatically at design time. The **Key** column contains important information that is used by ADF Desktop Integration for proper functioning of the table. Note that you must not remove the **Key** column at runtime.

7.13.1 How to Configure the Key Column

You can configure the **Key** column's position, style properties, and header label. By default, the `_ADF_DI_TableKeyCellStyle` style is applied to it.

Before you begin:

It may be helpful to have an understanding of the **Key** column in the ADF Table component. For more information, see [Section 7.13, "Configuring ADF Table Component Key Column."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

To configure the **Key** column:

1. Open the integrated Excel workbook.
2. Select the cell in the Excel worksheet that references the ADF Table component and click the **Edit Properties** button in the **Oracle ADF** tab.
3. In the Edit Component: ADF Table dialog, click the browse (...) icon beside the input field for **Columns**.

The Edit Columns dialog appears, listing all the columns of the selected ADF Table component.

4. Select the column with ID as `_ADF_RowKeyColumn`.
5. Change the column properties as desired, but do not change the following properties:
 - `DynamicColumn`

- InsertComponent
 - InsertUsesUpdate
 - UpdateComponent
 - ID
 - Visible
6. If desired, change the position of the column using the **Up** and **Down** arrow keys.
 7. Click **OK** to close Edit Columns dialog.
 8. Click **OK** to close the Edit Component: ADF Table dialog.

7.13.2 How to Manually Add the Key Column At Design Time

If you are using the integrated Excel workbook prepared and configured using an earlier version of ADF Desktop Integration, the **Key** column will not be available at design time. It will appear only at runtime. To configure the Key column properties, you can add it in the workbook at design time.

Before you begin:

It may be helpful to have an understanding of the Key column in the ADF Table component. For more information, see [Section 7.13, "Configuring ADF Table Component Key Column."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

To manually add the Key column at design time:

1. Open the integrated Excel workbook.
2. Select the cell in the Excel worksheet that references the ADF Table component, and then click the **Edit Properties** button in the **Oracle ADF** tab.
3. Add a new column in the ADF Table, and specify the properties as described in [Table 7-9](#). For more information about adding a column, see [Section 7.3.2, "How to Add a Column in an ADF Table Component."](#)

Table 7-9 Key Column Properties

Set this property...	To ...
CellStyleName	_ADFDI_TableKeyCellStyle
HeaderStyleName	_ADFDI_HeaderStyle
DynamicColumn	False
HeaderLabel	#{_ADFDIres[COMPONENTS_TABLE_ROWKEY_COL_LABEL]}
ID	_ADF_RowKeyColumn
InsertUsesUpdate	True
UpdateComponent	OutputText The Value property must be empty.
Visible	True

If desired, you may change the position of the Key column using the **Up** and **Down** arrow keys.

4. Click **OK**.

Note: You must specify the `ID` property of the new column as `_ADF_RowKeyColumn`; otherwise, the column will not be considered to be a **Key** column, and another **Key** column will automatically appear at runtime.

7.14 Creating a List of Values in an ADF Table Component Column

Use the `TreeNodeList` subcomponent when you want to render a dropdown list of values in an ADF Table component column. The list of values can display a maximum of two hundred and fifty values at runtime. Unlike other ADF Desktop Integration components, the `TreeNodeList` subcomponent does not appear in the components palette described in [Section 5.5, "Using the Components Palette."](#) Instead, you invoke it as a subcomponent when you specify values for the `InsertComponent` or `UpdateComponent` properties of an ADF Table component column. For information about the properties of an ADF Table component column, see [Section A.12.2, "ADF Table Component Column Properties."](#)

After you invoke the `TreeNodeList` subcomponent, you must specify a tree binding attribute associated with a model-driven list as a value for the `TreeNodeList` subcomponent's `List` property. The tree binding attribute associated with a model-driven list populates the dropdown menu in the Table component's column with a list of values after invocation of the Table component's `Download` action.

Note: You can create a model-driven list of values in your ADF Table component by choosing `ModelDrivenColumnComponent` as the subcomponent type. For more information about creating a model-driven list, see [Section 7.15, "Adding a ModelDrivenColumnComponent Subcomponent to Your ADF Table Component."](#)

For information about the properties of a `TreeNodeList` subcomponent, see [Section A.9, "TreeNodeList Subcomponent Properties."](#)

7.14.1 How to Create a List of Values in an ADF Table Component Column

You add a column to the ADF Table component column and select `TreeNodeList` as the subcomponent. You then specify a tree binding attribute as the value for the `TreeNodeList` subcomponent's `List` property. A model-driven list must be associated with the tree binding attribute that you specify.

Notes:

- The `TreeNodeList` subcomponent does not support a model-driven list whose control type is `input_text_lov` or `combo_lov`.
 - ADF List of Values components using date values are not supported.
 - The `TreeNodeList` subcomponent may not support model-driven lists for EJB-based data controls in all cases.
-
-

Before you begin:

It may be helpful to have an understanding of how to create a list of values in ADF Table component. For more information, see [Section 7.14, "Creating a List of Values in an ADF Table Component Column."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

To create a list of values in an ADF Table component column:

1. Open the integrated Excel workbook.
2. Select the cell in the Excel worksheet that references the ADF Table component and click the **Edit Properties** button in the **Oracle ADF** tab.
3. In the Edit Component: ADF Table dialog, click the browse (...) icon beside the input field for **Columns**.

The Edit Columns dialog appears, listing all the columns of the selected ADF Table component.

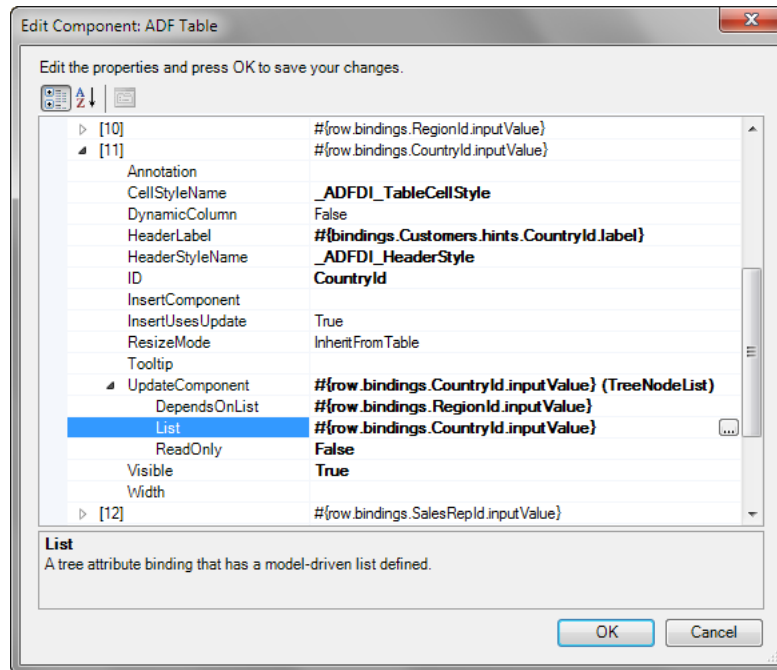
4. Click **Add** to add a new column.
5. Choose the appropriate option for the newly created column:
 - Click the browse (...) icon beside the input field for **InsertComponent** to configure the runtime list of values for insert operations.
 - Click the browse (...) icon beside the input field for **UpdateComponent** to configure the runtime list of values for update and download operations.

In both options, the Select subcomponent to create dialog appears.

6. Select **TreeNodeList** and click **OK**.
7. Expand the property that you selected in Step 5 and configure values as follows:
 - Select a tree binding attribute associated with a model-driven list for the `List` property.
 - Select a value for `DependsOnList` only if you intend to create a dependent list of values as described in [Section 8.8, "Creating Dependent Lists of Values in an Integrated Excel Workbook."](#) The tree binding attribute or list binding you select for `DependsOnList` serves as the parent list of values in a dependent list of values.
 - Configure the `ReadOnly` property as desired.
For information about these properties, see [Section A.9, "TreeNodeList Subcomponent Properties."](#)

Figure 7–15 shows the property inspector for an ADF Table component column in EditCustomers-DT.xlsx after `TreeNodeList` is selected as the subcomponent for the column's `UpdateComponent` property.

Figure 7–15 ADF Table Component Column Configured to Display a List of Values



8. Click OK.

7.14.2 What Happens at Runtime: How the ADF Table Column Renders a List of Values

At runtime, the ADF Table component invokes the `Download` action and populates each column. This action also populates the list of values in the column that you configure to render a list of values. Figure 7–16 shows an example from `EditCustomers-DT.xlsx` of the Summit sample application for ADF Desktop Integration, where **Country** is the column configured to display a list of values.

Figure 7–16 Runtime View of an ADF Table Component Column Displaying a List of Values

City	State	Zip Code	Country	Sales Rep.
Sao Paolo			Brazil	um
Osaka			Brazil	uyen
New Delhi			Japan	uyen
Seattle	Washington	98101	India	gee
Hong Kong			USA	mas
Cannes			Hong Kong	mas
			France	
			Nigeria	
			Germany	

7.15 Adding a ModelDrivenColumnComponent Subcomponent to Your ADF Table Component

You can add a ModelDrivenColumnComponent subcomponent to an ADF Table component. The value of ModelDrivenColumnComponent is determined by the Control Type hint specified for each attribute on the server.

At design time, for a column, specify the subcomponent type as ModelDrivenColumnComponent for the UpdateComponent or InsertComponent properties. At runtime, if there is a model-driven list associated with the attribute, then the column uses a dropdown list using the TreeNodeList subcomponent.

Note: If there is no model-driven list associated with the attribute, or if any non-list-based control type is specified, then the column uses an Input Text subcomponent. If there is a model-driven list whose control type is input_text_lov or combo_lov, then the column uses an Input Text subcomponent.

For more information about creating a model-driven list, see the "How to Create a Model-Driven List" section of the *Developing Fusion Web Applications with Oracle Application Development Framework*.

Support for Dependent List of Values

When multiple ModelDrivenColumnComponent list subcomponents are exposed in an ADF Table component, then for each list ADF Desktop Integration determines whether it depends on another model-driven list. It verifies that the bind variable specified for a list references an attribute bound to another list.

If the list depends on another model-driven list, the subcomponent's DependsOnList value is set automatically at runtime.

As server-side list binding dependencies are determined only for lists in the same tree node, the following tree node list bindings are not supported:

- A binding that depends on a list binding in a different tree or tree node
- A binding that depends on a list binding in the page definition file

7.16 Adding a Dynamic Column to Your ADF Table Component

You can add dynamic columns to an ADF Table component so that the ADF Table component expands or contracts at runtime depending on the available attributes returned by the view object. The DynamicColumn property of the Columns group in the TableColumn array controls this behavior. To make a column dynamic, set the DynamicColumn property to True. A dynamic column in the TableColumn array is a column that is bound to a tree binding or a tree node binding whose attribute names are not known at design time. A dynamic column can expand to more than a single worksheet column at runtime.

The ADF Table component's dynamic column supports the following subcomponent types:

- Input Text
- Output Text
- ModelDrivenColumnComponent

Note: ADF Desktop Integration does not support the subcomponent type `TreeNodeList` in a dynamic column.

Support for Model-Driven List of Values

You can also configure a dynamic column to support the List of Values subcomponent where the subcomponent type is determined from model configuration at runtime. At design time, specify the subcomponent type as `ModelDrivenColumnComponent` for the `UpdateComponent` or `InsertComponent` properties. At runtime, during dynamic column expansion, the model-driven runtime component is determined before caching the list of values. The remote servlet allows the client to retrieve Model configuration, allowing the client to choose the desired column subcomponent type. For more information about `ModelDrivenColumnComponent`, see [Section 7.15, "Adding a ModelDrivenColumnComponent Subcomponent to Your ADF Table Component."](#)

7.16.1 How to Configure a Dynamic Column

You configure a dynamic column by specifying an EL expression with the following format for the `Value` property of the component specified by the ADF Table component column's `InsertComponent` property as a subcomponent:

```
#{bindings.TreeID.[TreeNodeID].AttributeNamePrefix*.inputValue}
```

or:

```
#{bindings.TreeID.AttributeNamePrefix*.inputValue}
```

where:

- `TreeID` is the ID of the tree binding used by the ADF Table component
- `TreeNodeID` is an optional value that specifies the tree node binding ID. If you omit this value, all matching attributes from the tree binding display regardless of which tree node binding the attribute belongs to.
- `AttributeNamePrefix` identifies a subset of attributes that exist within the tree binding's underlying iterator. If you do not specify a value for `AttributeNamePrefix`, all attributes for the tree binding or tree binding node are returned. Always use the `*` character.

Note: While adding a dynamic column, ensure that tree node attribute names are not specified in the page definition file. At runtime, the tree node object returns all attribute names from the underlying iterator. If there are attribute names specified in the page definition file, the tree node object limits the list of available attribute names based on that list.

The following example returns all attributes that begin with the name "period" in the `model.EmpView` node of the `EmpTree` binding:

```
#{bindings.EmpTree.[model.EmpView].period*.inputValue}
```

7.16.2 What Happens at Runtime: How Data Is Downloaded or Uploaded In a Dynamic Column

When the ADF Table component's `Download` or `DownloadForInsert` action is invoked, the ADF Table component automatically updates the dynamic columns so that they contain an up-to-date set of matching attributes. For each invocation of `Download`, ADF Desktop Integration requires that all rows must have the same set of attributes for the dynamic column. It may generate errors if the set of attributes changes from row to row during `Download`.

If a dynamic column supports both `Insert` and `Update` operations, you should specify the same EL expression for the `Value` properties of the dynamic column's `InsertComponent` and `UpdateComponent` subcomponents. At runtime, the ADF Table component expands to include a dynamic column that displays the value of the attribute binding returned by the EL expression.

When the ADF Table component's `Upload` action is invoked, the workbook prompts the end user to determine if the end user wants to continue to upload data when the previously downloaded attributes no longer exist in the tree binding.

Note: If an Excel AutoFilter is applied on a table with dynamic columns, the AutoFilter is automatically removed every time the dynamic columns are expanded, collapsed, or adjusted.

Support for View Objects with Declarative SQL Mode

To support view objects that are configured with declarative SQL mode and customized at runtime, expose a tree binding in the page definition file that has no attributes defined. For example:

```
<tree IterBinding="DeclSQLModeIterator" id="DeclSQLModeTree">
  <nodeDefinition Name="DeclSQLModeTreeNode" />
</tree>
```

At runtime, the tree binding will return the selected attributes from the underlying declarative SQL mode view object to the integrated Excel worksheet.

7.16.3 How to Specify Header Labels for Dynamic Columns

Use the following syntax to write EL expressions for the `HeaderLabel` property of a dynamic column:

```
#{bindings.TreeID.[TreeNodeID].hints.AttributeNamePrefix*.label}
```

or:

```
#{bindings.TreeID.hints.AttributeNamePrefix*.label}
```

Specify the same tree binding ID, tree node binding ID, and attribute name prefix values in the `HeaderLabel` property of the dynamic column as the values you specify for the `Value` properties of the dynamic column's `InsertComponent` and `UpdateComponent` if the dynamic column supports `Insert` and `Update` operations.

Note: The ADF Table component ignores the value of a column's `Visible` property when you configure a column to be dynamic. For more information about ADF Table component column properties, see [Table A-13](#).

If you want the mandatory columns, where the end user must enter a value, to be marked with a character or a string, you must configure the `HeaderLabel` property. Use the following syntax to write EL expression to add a character or string to all mandatory columns:

```
=IF("#{bindings.TreeID.[TreeNodeID].hints.*.mandatory}", "<prefix_for_mandatory_cols>", "") & "#{bindings.TreeID.[TreeNodeID].hints.*.label}"
```

For example, the following EL expression adds an asterisk (*) character to the mandatory columns label:

```
=IF("#{bindings.MyTree.[myapp.model.MyChildNode].hints.*.mandatory}", "*" , "" ) & "#{bindings.MyTree.[myapp.model.MyChildNode].hints.*.label}"
```

7.16.4 How to Specify Styles for Dynamic Columns According to Attribute Data Type

You can specify different styles for each data type according to the data type of the column. Use the following syntax to write EL expressions for the `CellStyleName` property of a dynamic column:

```
=IF("#{bindings.TreeID.[TreeNodeID].hints.*.dataType}"="<data_type>" , <custom_style_expression1> , <custom_style_expression2>)
```

In the following example, the `MyDateStyle` style is applied to all date columns, and `MyDefaultStyle` is applied to other data type columns:

```
=IF("#{bindings.MyTree.[myapp.model.MyChildNode].hints.*.dataType}"="date" , "MyDateStyle" , "MyDefaultStyle")
```

The following example shows another scenario where the `MyDateStyle` style is applied to all date data type columns, `MyNumberStyle` is applied to all number data type columns, and `MyDefaultStyle` is applied to other data type columns:

```
=IF("#{bindings.MyTree.[myapp.model.MyChildNode].hints.*.dataType}"="date" , "MyDateStyle" , IF("#{bindings.MyTree.[myapp.model.MyChildNode].hints.*.dataType}"="number" , "MyNumberStyle" , "MyDefaultStyle"))
```

For more information about EL expressions, see [Appendix B, "ADF Desktop Integration EL Expressions."](#)

7.17 Configuring Column Widths in an ADF Table Component

You can configure column widths of an ADF Table component so that they are automatically resized at runtime. The columns can be resized using Excel's AutoFit column width feature, which determines the width based on the data values in the column. ADF Desktop Integration can also resize the columns using explicit width values derived from EL expressions.

The resizing behavior of ADF Table columns is configured at the table level. You can then override them at the column level.

7.17.1 How to Configure an ADF Table Component to Resize Columns at Runtime

You can use the design-time `ResizeColumnsMode` property to specify the common resizing behavior for all columns of the ADF Table component. Use the `ResizeColumns` table component method to control when the resizing occurs. To override resizing behavior of a particular column, use the column's `ResizeMode` property.

Resizing a column's width at runtime is a two-step process. First, you need to configure the table column with the desired width-related property values. Then, add `ResizeColumns` to the desired action sets.

Before you begin:

It may be helpful to have an understanding of configuring resizing behavior of ADF Table columns. For more information, see [Section 7.17, "Configuring Column Widths in an ADF Table Component."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

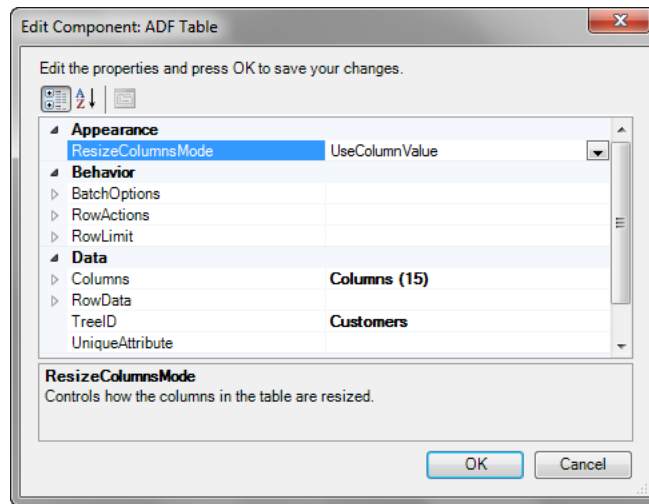
To configure resizing behavior of ADF Table columns:

1. Open the integrated Excel workbook.
2. Select any cell in the ADF Table component and click the **Edit Properties** button in the **Oracle ADF** tab.
3. In the Edit Component: ADF Table dialog, configure and set the `ResizeColumnsMode` property as described in the [Table 7-10](#):

Table 7-10 *ResizeColumnsMode Values of the ADF Table Component*

Value	Description
<code>UseColumnValue</code>	Default. The table is not automatically resized at runtime, unless individual columns have <code>Column.ResizeMode</code> properties set to a value other than <code>InheritFromTable</code> .
<code>AutoFitAllWithHeader</code>	<p>All columns within the table boundaries are resized to best fit using Excel's AutoFit support. Data values in the columns' cells, including the header cells, are used to determine the best fit.</p> <p>Individual columns that have <code>Column.ResizeMode</code> properties set to a value other than <code>InheritFromTable</code> are resized accordingly.</p> <p>Note that values in the column's cells above or below the table are not considered when finding the best fit.</p>
<code>AutoFitAllWithoutHeader</code>	<p>All columns within the table boundaries are resized to best fit using Excel's AutoFit support. Data values in the columns' cells, excluding the header cells, are used to determine the best fit.</p> <p>Individual columns that have <code>Column.ResizeMode</code> properties set to a value other than <code>InheritFromTable</code> are resized accordingly.</p> <p>Note that values in the column's cells above or below the table are not considered when finding the best fit.</p>

[Figure 7-17](#) shows the `ResizeColumnsMode` property at design-time.

Figure 7–17 *ResizeColumnsMode Property in design-time*

4. To configure the resizing behavior of a column and override the table-level resizing behavior, set the `ResizeMode` property.

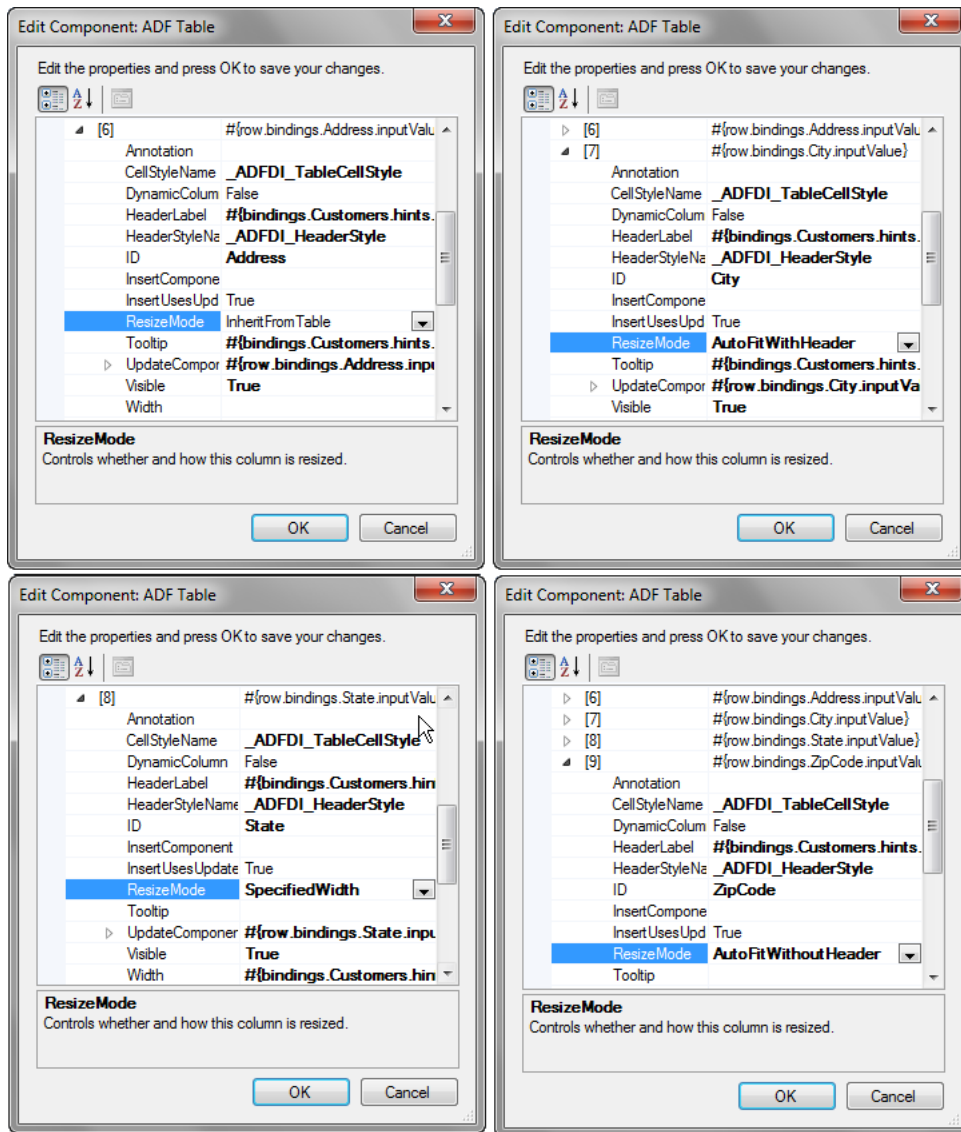
In the Edit Component: ADF Table dialog, expand the `Columns` property and set the `ResizeMode` property as described in the [Table 7–11](#):

Table 7–11 *ResizeMode Values of the ADF Table Column Property*

Value	Description
Manual	The column is not resized; column width is left at the current setting.
InheritFromTable	Default. The column is resized based on the table's <code>ResizeColumnsMode</code> setting. If <code>ResizeColumnsMode</code> is set to <code>UseColumnValue</code> , then no resizing occurs.
AutoFitWithHeader	Including the header cell, the column is resized to best fit using the Excel's AutoFit support.
AutoFitWithoutHeader	Excluding the header cell, the column is resized to best fit using the Excel's AutoFit support.
SpecifiedWidth	ADF Desktop Integration uses the <code>Width</code> property to determine the desired width of the column. You can specify a numerical value, or an EL expression.

[Figure 7–18](#) shows the `ResizeMode` property of four different columns of the ADF Table component of `EditCustomers-DT.xlsx` at design-time.

Figure 7–18 *ResizeMode Property in design-time*



5. If `ResizeColumnsMode` is set to `UseColumnValue`, and `SizeMode` is set to `SpecifiedWidth`, set `Column.Width` to the number of characters you want to display in the column.

A given column's width property may be set to a literal numerical value or an EL expression that evaluates to a number between 1 and 255, inclusive. If the expression cannot be evaluated, or if the expression evaluates to less than 1 or greater than 255, the `SizeMode` is considered to be `Manual` and the column is not resized. An example EL expression for `width` that makes use of the UI Hint `displayWidth` for an attribute is:

```
{bindings.Customers.hints.Name.displayWidth}
```

6. Click **OK**.

7.17.2 How to Configure an Action Set to Resize Columns of an ADF Table Component at Runtime

You can configure the action set in a worksheet ribbon command, ADF Button component, or a worksheet event to invoke the ADF Table component `ResizeColumns` action.

Note that resizing a table with many columns and many rows might take a noticeable amount of time.

Before you begin:

It may be helpful to have an understanding of configuring resizing behavior of ADF Table columns. For more information, see [Section 7.17, "Configuring Column Widths in an ADF Table Component"](#) and [Section 8.2, "Using Action Sets."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

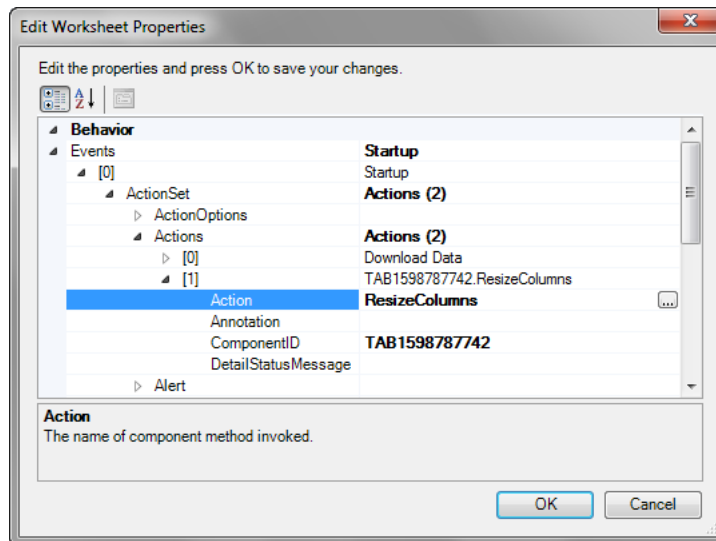
To configure an action set to resize Columns of an ADF Table component:

1. Open the integrated Excel workbook.
2. Click the **Worksheet Properties** button in the **Oracle ADF** tab, and add a ribbon command. For more information about adding a ribbon command in a worksheet, see [Section 8.3.1, "How to Define a Workbook Command Button for the Runtime Ribbon Tab."](#)

Note: Instead of adding a ribbon command, you can configure a worksheet event or an Oracle ADF component (a button, for example) to invoke the action set at runtime.

3. Open the Edit Action dialog to configure an action set. For more information about invoking action sets, see [Section 8.2, "Using Action Sets."](#)
4. Add the ADF Table component `ResizeColumns` action to the list of actions that the action set invokes at runtime. Note that `ResizeColumns` is a component action.
5. Click **OK**.

[Figure 7-19](#) shows the `ResizeColumns` action in `EditCustomers-DT.xlsx` at design-time.

Figure 7–19 *ResizeColumns Action*

7.17.3 What Happens at Runtime: How the ADF Table Columns are Resized

The ADF Table columns are resized as the result of running of `Table.ResizeColumns` component action in an action set (see [Section 7.17.2, "How to Configure an Action Set to Resize Columns of an ADF Table Component at Runtime"](#)).

7.17.4 What You May Need to Know About Resizing Columns of an ADF Table Component at Runtime

The entire worksheet columns containing the ADF Table component columns are resized depending on the values in the `Table.ResizeColumnsMode` and `Column.ResizeMode` properties. Resizing the table columns affects the contents of the cells or any other components (such as form components) located in the same Excel worksheet column outside of the table's boundaries.

If a worksheet contains two or more ADF Table components configured with action sets to be resized at runtime, each table attempts to resize the Excel worksheet columns independently, but the table's `ResizeColumns` action that runs last sets the column width.

Tip: For worksheets that contain more than one ADF Table component, call the `ResizeColumns` action only on the primary table

Notes:

- The `Column.Width` property does not support row-specific bindings.
 - A common strategy is to call `ResizeColumns` after one of the Download actions. See [Figure 7-19](#) for an example.
 - Excel internally rounds the specified `Width` values to the nearest whole pixel value. For example, a value of 8.5 characters rounds to 8.43, which equates to 64 pixels.
 - Using one of the `AutoFit` resizing modes on cells that have **Wrap Text** selected in their style definition may not resize as expected. Using `SpecifiedWidth` mode, or removing the **Wrap Text** setting from the style, may produce better results.
-
-

7.18 Displaying Tooltips in ADF Desktop Integration Table-Type Components

You can add tooltips to table column headers of ADF Table and ADF Read-only Table components. Note that the tooltip is always initially empty for headers of special columns of the ADF Table component.

To add a tooltip to an table column header:

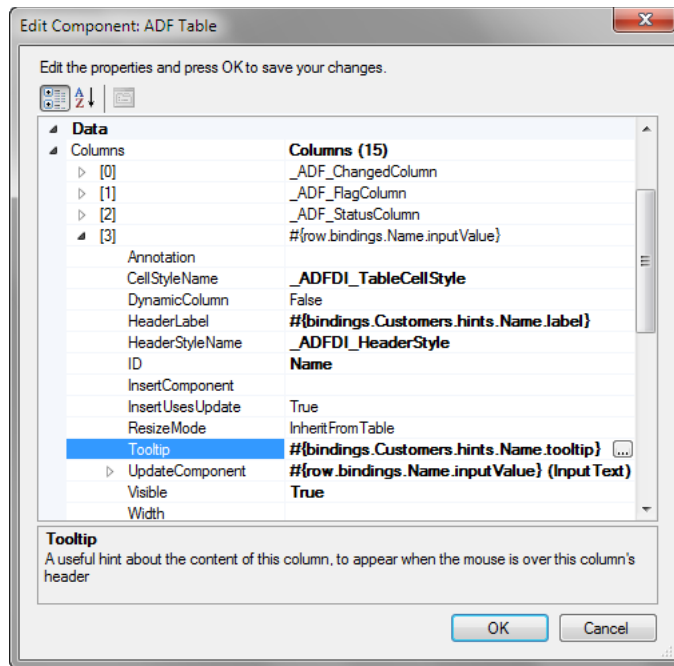
1. Open the integrated Excel workbook.
2. If the Table-type component has already been inserted in the Excel worksheet, click any cell of the table, and click **Edit Properties** in the Oracle ADF tab.

To insert a Table-type to the worksheet, select the cell where you want to anchor the component. In the components palette or the bindings palette, select the Table-type component or the binding, and click **Insert Component** or **Insert Binding**.

3. In the Edit Component: ADF Table dialog, expand the `Columns` property. Click the browse (...) icon of the `Tooltip` property of the desired column, and enter the tooltip message.

[Figure 7-20](#) shows the tooltip EL expression for an ADF Table column header in `EditCustomers-DT.xlsx`.

Figure 7–20 *Tooltip for ADF Table Column Header at Design-time*



4. Click **OK**.

Figure 7–21 shows the tooltip message at runtime. Notice the small red arrow at the top-right of the header cell. It indicates the header cell has a comment. Hover mouse pointer over the cell to see the tooltip message.

Figure 7–21 *Tooltip for ADF Table Column Header at Runtime*

Edit Customers Sheet						
Changed	Flagged	Status	Name	Rating	Phone	Address
			Unisports	Excellent	55-2066101	72 Via Bahia
			Simms Athletics	Poor	81-20101	Takashi
			Delhi Sports	Good	91-10351	Chanakya
			Womansport	Fair	1-206-104-0103	281 King Street
			Ojibway Retail	Fair	1-716-555-7171	415 Main Street

7.18.1 What You May Need to Know About Tooltips for Table Columns

The tooltips for column headers are evaluated and rendered when the table column headers are rendered including first time table initialization, `Table.Initialize`, and `Table.Download` actions.

By default, the tooltips for special columns are empty. If the `Tooltip` property of a special column is set to a non-empty expression, it is trimmed of whitespace, and inserted into the target cell as an Excel comment.

To get a unique tooltip for each expanded dynamic column at runtime, enter the expression in the following syntax in the `ToolTip` property:

```
{bindings.<TreeID>.hints.*.tooltip}
```

At runtime, the dynamic column expands to the available set of attributes in the specified tree or the node. ADF Desktop Integration also retrieves the corresponding tooltip values and applies each one to the appropriate column using the rules described above.

For more information about tooltips, see [Section 6.9.1, "What You May Need to Know About Tooltips."](#) You can also add tooltips to the headers of special columns of the table components (see [Section 7.12, "Special Columns in the ADF Table Component"](#)) and the dynamic columns (see [Section 7.16, "Adding a Dynamic Column to Your ADF Table Component"](#))

7.19 Creating an ADF Read-Only Table Component

At runtime, the ADF Read-only Table component renders a table across a continuous range of cells that displays data from the tree binding that the ADF Read-only Table component references. Use this component to display data that you do not want the end user to edit.

This component supports several properties, such as `RowLimit`, that determine how many rows the component downloads when it invokes its `Download` action. It also includes a group of properties (`Columns`) that determine what columns from the tree binding appear at runtime in the Excel worksheet. The `TreeID` property specifies the tree binding that the component references. More information about these properties and others that the ADF Read-only Table component supports can be found in [Section A.13, "ADF Read-only Table Component Properties and Actions."](#)

ADF Desktop Integration uses the first column of a worksheet (column A) to store some special information needed for the proper operation of Read-only Table components. Therefore, at runtime, column A is hidden for any worksheet that contains at least one instance of ADF Read-only Table component. The content of the worksheet are shifted by one column in the right direction to compensate for the hiding of column A, and maintain the visual layout from design time. Upon returning to design time, column A is unhidden and the worksheet contents are shifted back one column in the left direction.

[Figure 7–22](#) shows the ADF Read-only Table component at design time with the property inspector in the foreground.

Figure 7-22 ADF Read-only Table Component at Design Time

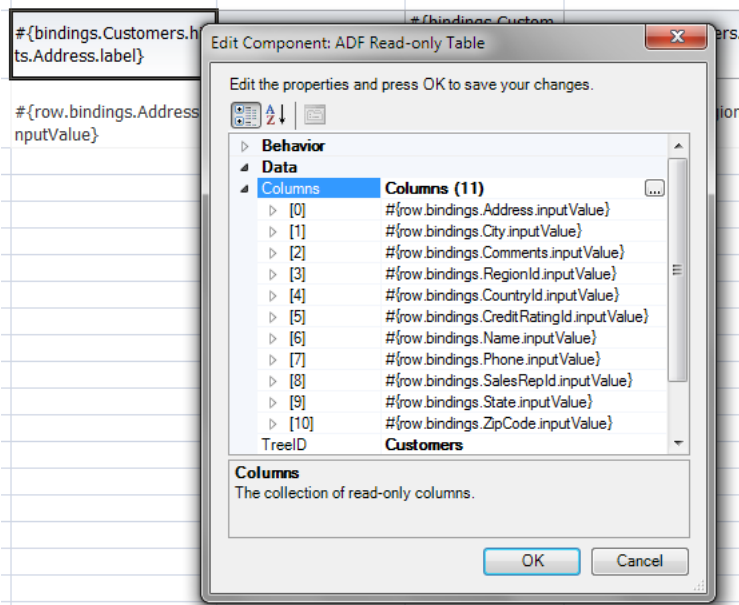


Figure 7-23 shows the columns that an ADF Read-only Table component which references the Customers tree binding at runtime.

Figure 7-23 Columns in an ADF Read-only Table Component at Runtime

Name	Credit Rating	Phone	Address	City	State	Zip Code	Country	Sales Rep.	Comments
Unisports	1	55-2066101	72 Via Bahia	Sao Paolo				1	12 Customer usua
Simms Athletic	4	81-20101	Takashi	Osaka				2	14 Customer shou
Delhi Sports	2	91-10351	Chanakya	New Delhi				3	14 Customer speci
Womansport	3	1-206-104-0103	281 King Stree	Seattle	Washington	98101		4	11
Kam's Sporting	1	852-3692888	95 Queensway	Hong Kong				5	15
Sportique	1	33-2257201	4 Rue Chabaud	Cannes				6	15 Customer speci
Sweet Rock Sp	2	234-6036201	6 Saint Antoine	Lagos				7	
Muench Sports	3	49-527454	Marktplatz	Stuttgart				8	15 Customer usua
Beisbol Sit	1	809-352689		San Pedro de Macon's				9	11

Notes:

- At runtime, inserting a row into the ADF Read-only Table component results in a new Excel row that behaves as if it is part of the downloaded data set, but the new row exists only in Excel. The data from the new row is not uploaded to the server, and does not affect the Fusion web application data.
- Read-only columns include double-click action sets. However, these actions cannot reliably position on the current row. So, the results of using row-level action sets with the ADF Read-only Table component is not consistent. If you need to use row-level action sets with reliable row positioning, use the ADF Table component instead of the ADF Read-only Table component.

7.19.1 How to Insert an ADF Read-only Table Component

You use the ADF Desktop Integration Designer task pane to insert an ADF Read-only Table component into a worksheet.

Before you begin:

It may be helpful to have an understanding of ADF Read-only Table component. For more information, see [Section 7.19, "Creating an ADF Read-Only Table Component."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

To insert an ADF Read-only Table component:

1. Open the integrated Excel workbook.
2. Select the cell in the Excel worksheet where you want to anchor the component.

When inserting a table component, you must ensure that the data of two tables does not overlap at runtime, and the selected cell is not a merged cell

3. In the bindings palette, select the binding to create the ADF Read-only Table component, and then click **Insert Binding**.
4. In the dialog that appears, select **ADF Read-only Table**.

Note: You can also insert an ADF Read-only Table component by using the components palette or **Oracle ADF** tab. Select **ADF Read-only Table** and click **Insert Component**. If you use the components palette to create the component, you would have to add each column to appear in the component at runtime.

5. Configure properties in the property inspector that appears to determine the columns to appear and the actions the component invokes at runtime.
6. Click **OK**.

Note: You can modify the properties of the component at a later time by selecting the cell in the worksheet that anchors the component and then displaying the property inspector.

To remove the table component, use the Delete ribbon command. For more information, see [Section 5.14, "Removing ADF Desktop Integration Components."](#)

7.19.2 How to Manually Add a Column to the ADF Read-only Table Component

You can manually add additional columns to an ADF Read-only Table component or re-add columns that you previously removed.

Before you begin:

It may be helpful to have an understanding of ADF Read-only Table component. For more information, see [Section 7.19, "Creating an ADF Read-Only Table Component."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

To manually add a column to the ADF Read-only Table component:

1. Open the integrated Excel workbook.

2. Select the cell in the worksheet that hosts the ADF Read-only Table component and click the **Edit Properties** button in the **Oracle ADF** tab.
3. In the Edit Component: ADF Table dialog, click the browse (...) icon beside the input field for **Columns**.

The Edit Columns dialog appears, listing all the columns of the selected ADF Table component

4. Click **Add** to add a new column to the ADF Read-only Table component.
5. Set values for the properties of the new column.

For information about the properties of an ADF Read-only Table component column, see [Table A-16](#).

6. Click **OK**.

7.20 Limiting the Number of Rows Your Table-Type Component Downloads

You can configure the number of rows that an ADF Table or ADF Read-only Table component downloads by setting values for the component's `RowLimit` group of properties. You can also display a warning message, if desired, that alerts the end user when the number of rows available to download exceeds the number of rows specified for download.

7.20.1 How to Limit the Number of Rows a Component Downloads

Specify the number of rows that the component downloads when it invokes its Download action as a value for the `RowLimit.MaxRows` property. Optionally, write an EL expression for the `RowLimit.WarningMessage` property so that the end user receives a message if the number of rows available to download exceeds the number specified by `RowLimit.MaxRows`.

Before you begin:

It may be helpful to have an understanding of how to limit the number of rows while downloading data in your ADF Table component. For more information, see [Section 7.20, "Limiting the Number of Rows Your Table-Type Component Downloads."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

To limit the number of rows a table-type component downloads:

1. Open the integrated Excel workbook.
2. Select the cell in the Excel worksheet that references the table-type component and click the **Edit Properties** button in the **Oracle ADF** tab.

For more information, see [Section 8.2, "Using Action Sets."](#)

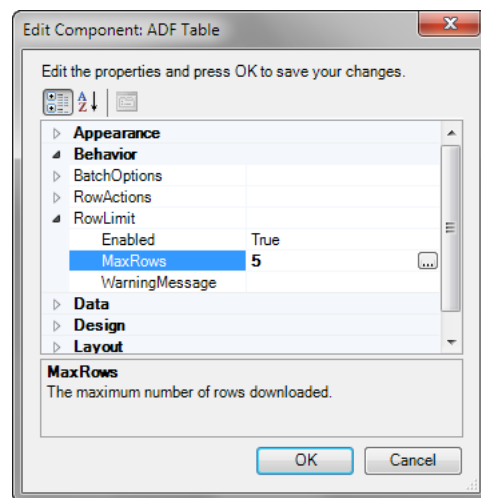
3. Configure properties for the `RowLimit` group of properties, as described in [Table 7-12](#). For more information about these properties, see [Section A.1, "Frequently Used Properties in the ADF Desktop Integration."](#)

Table 7–12 RowLimit Group of Properties

Set this property to...	This value...
<code>RowLimit.Enabled</code>	Set to <code>True</code> to limit the number of rows downloaded to the value specified by <code>RowLimit.MaxRows</code> .
<code>RowLimit.MaxRows</code>	Specify an EL expression that evaluates to the maximum number of rows to download.
<code>RowLimit.WarningMessage</code>	Write an EL expression for this property to generate a message for the end user if the number of rows available to download exceeds the number specified by <code>RowLimit.MaxRows</code> . If the value for this property is null, the <code>Download</code> action downloads the number of rows specified by <code>RowLimit.MaxRows</code> displaying the default warning message to the end user.

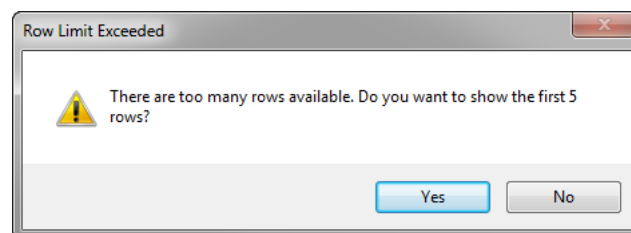
4. Click OK.

Figure 7–24 shows the Edit Component dialog in the `EditCustomers-DT.xlsx` workbook where the row limit of an ADF Table component is configured.

Figure 7–24 Limiting Number of Rows of an ADF Table Component

7.20.2 What Happens at Runtime: How the RowLimit Property Works

When invoked, the Table-type component's `Download` action downloads the number of rows that you specified as the value for `RowLimit.MaxRows` from the Fusion web application. A message dialog similar to the one in Figure 7–25 appears if you specify an EL expression for `RowLimit.MaxRows` or do not modify its default value.

Figure 7–25 Row Limit Exceeded Warning Message

7.21 Clearing the Values of Cached Attributes in an ADF Table Component

The `RowData` group of properties described in [Table A-12](#) allow you to specify data to cache in the ADF Table component. For more information about this functionality, see the following:

- [Section 12.7, "Handling Data Conflicts When Uploading Data from a Workbook"](#)
- [Chapter 15, "Using an Integrated Excel Workbook Across Multiple Web Sessions and in Disconnected Mode"](#)

The ADF Table component exposes an action (`ClearCachedRowAttributes`) that, when invoked, clears the values of cached attributes for the current row of the ADF Table component.

Do not configure a component (for example, an ADF Table component's column or an ADF Input Text component) so that an end user can view or edit an attribute binding that you have also specified for an element in the `RowData.CachedAttributes` array. The `RowData.CachedAttributes` array caches the values retrieved by the worksheet `DownSync` action. The worksheet `UpSync` action sends the values of the `RowData.CachedAttributes` array to the Fusion web application. This may override edits an end user makes to an attribute binding exposed through a component in the worksheet.

7.21.1 How to Clear the Values of Cached Attributes in an ADF Table Component

Configure a `DoubleClickActionSet` that includes an action to invoke the ADF Table component's `ClearCachedRowAttributes` action.

Before you begin:

It may be helpful to have an understanding of how to clear cached attributes in an ADF Table component. For more information, see [Section 7.21, "Clearing the Values of Cached Attributes in an ADF Table Component."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

To clear the values of cached attributes in an ADF Table component:

1. Open the integrated Excel workbook.
2. Open the Edit Action dialog for the Oracle ADF component that is going to invoke the `DoubleClickActionSet` at runtime.

For more information about invoking action sets, see [Chapter 8.2, "Using Action Sets."](#)

3. Add an action to the `DoubleClickActionSet` that invokes the ADF Table component's `ClearCachedRowAttributes` action.
4. Click OK.

7.21.2 What Happens at Runtime: How the ADF Table Component Clears Cached Values

The action set invokes the ADF Table component's `ClearCachedRowAttributes` action. This action clears the cached values specified by the `RowData.CachedAttributes` property for the current row of the ADF Table component.

7.22 Tracking Changes in an ADF Table Component

End users can create or modify data in the cells of an integrated Excel workbook that hosts an ADF Table component.

If a column is updatable and not read-only, change tracking is activated. End users can make the following changes to activate change tracking:

- Edit cell values
- Insert or delete cell values
- Paste values to cells in the ADF Table component column that they copied elsewhere

A character that resembles an upward pointing arrow appears in a row of the `_ADF_ChangedColumn` column if the end user makes a change to data in a corresponding row. [Figure 7-26](#) shows an example.

Figure 7-26 *Changed Column in an ADF Table Component*



This character appears if the end user makes a change to data hosted by a component where the component's `ReadOnly` property value is `False`. The ADF Input Text and `TreeNodeList` subcomponents both have a `ReadOnly` property. You can write an EL expression or a static string for this `ReadOnly` property that evaluates to `True` or `False`. If you write a static string or an EL expression that evaluates to `True`, no character appears in the `_ADF_ChangedColumn` column. For more information about `ReadOnly` EL expressions and change tracking, see [Section 7.23, "Evaluating EL Expression for ReadOnly Properties."](#)

7.23 Evaluating EL Expression for ReadOnly Properties

If a table column's `ReadOnly` EL expression contains a binding expression (for example, `#{row.bindings.color.inputValue}`), the runtime evaluation of that expression will be different depending on when the evaluation occurs. The evaluation happens during the following:

- downloading data (`Download`, `DownloadFlaggedRows`, `DownloadForInsert`)
- uploading data (`Upload`), and change tracking

7.23.1 What Happens at Runtime: Evaluating EL Expression While Downloading Data

During `Download`, the EL expression is evaluated with the current binding value as expected.

7.23.2 What Happens at Runtime: Evaluating EL Expression While Uploading Data or Tracking Changes

During `Upload`, or when the end user changes values in the editable table, the EL expression is evaluated differently than `Download`. Specifically, an empty string is substituted for the binding expression prior to evaluation of the EL expression.

For example, if you have the following EL expression in an editable cell:

```
=IF("#{row.bindings.color.inputValue}"="RED", True, False)
```

During Upload, or when the end user changes values in the editable table, the EL expression evaluates to `=IF("="RED", True, False)`, and always returns False.

7.23.3 What You May Need to Know About Evaluating EL Expression While Uploading Data or Tracking Changes

During Upload and change tracking, an extra round trip to the server would be required to retrieve the binding values, in order to evaluate the EL expression properly. The extra round trip to the server would impact performance negatively, and could even require a new login if the end user did not have a currently valid session.

Note: The same EL expression evaluation behavior also applies to the `CellStyleName` EL expression property when inserting new worksheet rows during table change tracking.

Due to the difference in behavior, if possible, you should avoid ReadOnly EL Expressions that contain binding expressions. However, if it is important for a given use case to use an attribute value in the ReadOnly expression, you should consider setting the worksheet protection to `Automatic`. For more information about worksheet protection, see [Section 9.7, "Using Worksheet Protection."](#)

For example, if you have the following EL expression in a cell:

```
=IF("#{row.bindings.color.inputValue}"="RED", True, False)
```

During Download, the RED cells in this column will be set to `Locked` and the end user will not be able to edit those cells.

7.24 Using Explicit Worksheet Setup Action

ADF Desktop Integration provides several features for configuring or customizing a worksheet after the binding container's metadata has been obtained from the server at runtime. However, at times, you might want to customize the data or the binding container before the client retrieves the binding container metadata. For example, at design time, you might want to add a table to the worksheet, but without specifying the View Object that will drive that table, until runtime. This would be desirable if the View Object to be used depends on some parameter values or settings that are not known until runtime. In addition, you might want to customize the View Object based on runtime parameter values (such as add attributes, or indicate which attributes to display). Similarly, you may also want to configure the binding container based on runtime parameter values. Such use cases require performing setup tasks before the binding container metadata is sent from the sever to the worksheet.

Using the Explicit Worksheet Setup Action feature of ADF Desktop Integration, you can specify a setup action that is invoked before the client retrieves the binding container metadata.

7.24.1 How to Configure Explicit Worksheet Setup Action

Using the `SetupActionID` property of the worksheet, you can specify a method that is invoked before the binding container metadata is sent to the worksheet. In the method, you can implement the logic necessary for any customization on the data and binding container.

Before you begin:

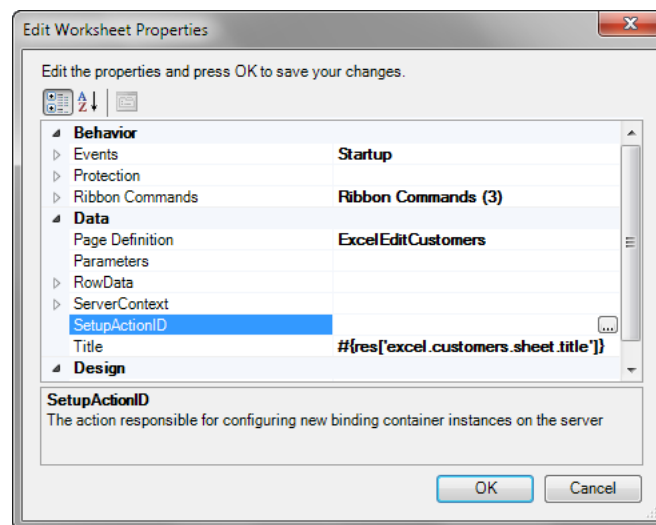
It may be helpful to have an understanding of the Explicit Worksheet Setup Action feature. For more information, see [Section 7.24, "Using Explicit Worksheet Setup Action."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 7.1.2, "Additional Functionality of Table-Type Components."](#)

To use SetupActionID property of the worksheet:

1. Open the worksheet in the integrated Excel workbook.
2. From the Excel Ribbon, click **Workbook Properties**.
3. In the Edit Worksheet Properties dialog, expand Data and click the browse icon (...) beside the input field for the **SetupActionID** property, as shown in [Figure 7-27](#).

Figure 7-27 SetupActionID Property in Edit Worksheet Properties Dialog



4. In the Select Binding dialog, select the action that you want to invoke before the binding container metadata is sent to the worksheet, and click **OK**.

Note: The `SetupActionID` property accepts ADFm actions only. Validation error is reported if an invalid method is set for the property.

5. Click **OK** to close the Edit Worksheet Properties dialog.

7.24.2 What You May Need to Know About Explicit Worksheet Setup Action

After the action specified in the `SetupActionID` property runs, the binding container metadata that is sent to worksheet reflects the customization configured in the method. ADF Desktop Integration ensures that the setup action runs only once for any binding container instance. If, for any reason, a new binding container instance becomes associated with the worksheet, the setup action will be invoked again, to ensure it is configured.

If any kind of failure occurs during the invoking of the setup action, ADF Desktop Integration is automatically disabled in the worksheet. Logging out, and then logging in, will not enable ADF Desktop Integration in the worksheet. Running **Clear All Data** command from the Excel Ribbon re-enables ADF Desktop Integration in the worksheet, the setup action runs again on subsequent requests.

Adding Interactivity to Your Integrated Excel Workbook

This chapter describes how to add interactivity options to your integrated Excel workbook, how to configure the ribbon tab, creating databound search forms and dependent list of values, and how to use EL expressions in Excel formula.

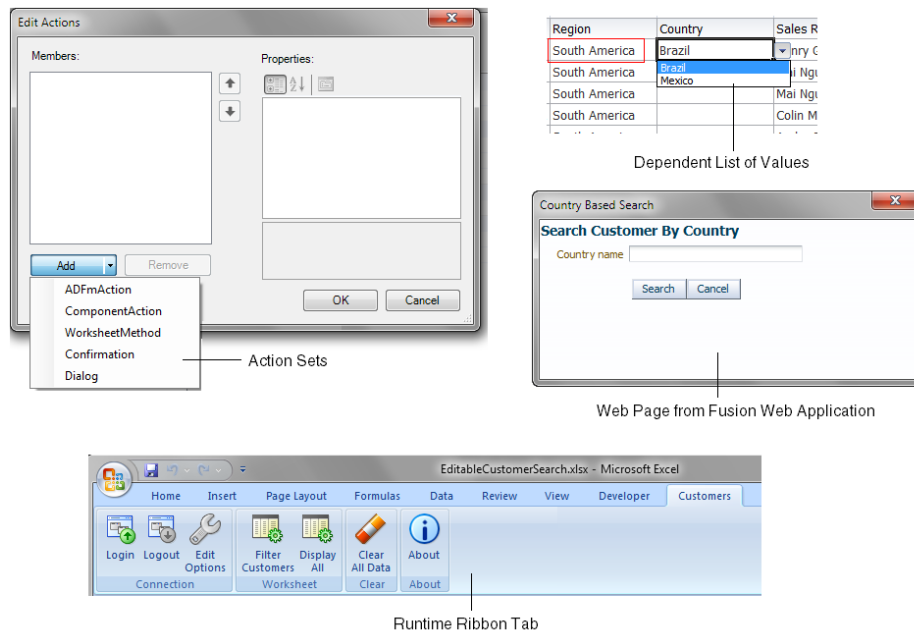
This chapter includes the following sections:

- [Section 8.1, "About Adding Interactivity to an Integrated Excel Workbook"](#)
- [Section 8.2, "Using Action Sets"](#)
- [Section 8.3, "Configuring the Runtime Ribbon Tab"](#)
- [Section 8.4, "Displaying Web Pages from a Fusion Web Application"](#)
- [Section 8.5, "Adding a Custom Popup Picker Dialog to an ADF Table Column"](#)
- [Section 8.6, "Creating ADF Databound Search Forms in an Integrated Excel Workbook"](#)
- [Section 8.7, "Creating a Form in an Integrated Excel Workbook"](#)
- [Section 8.8, "Creating Dependent Lists of Values in an Integrated Excel Workbook"](#)
- [Section 8.9, "Using EL Expression to Generate an Excel Formula"](#)
- [Section 8.10, "Using Calculated Cells in an Integrated Excel Workbook"](#)
- [Section 8.11, "Using Macros in an Integrated Excel Workbook"](#)

8.1 About Adding Interactivity to an Integrated Excel Workbook

You can make your integrated workbook interactive to the end user by using features such as action sets, configuring the runtime ribbon tab, creating dependent list of values, and so on. [Figure 8-1](#) shows some of the interactive features.

Figure 8–1 Interactivity Features in an Integrated Excel Workbook



Adding interactivity to an integrated Excel workbook permits end users to run action sets that invoke Oracle ADF functionality in the workbook. It also provides status messages, alert messages, and error handling in the integrated Excel workbook while these action sets run. In addition to end-user gestures (double-click, click, select) on the ADF Desktop Integration components that invoke action sets, you can configure workbook and worksheet ribbon buttons that end users use at runtime to invoke action sets.

8.1.1 Adding Interactivity to Integrated Excel Workbook Use Cases and Examples

To make your integrated Excel workbook interactive, you can use action sets that are invoked by the end user's gestures. For example, as shown in [Figure 8–2](#), the **Filter Customers** ribbon command in `EditableCustomerSearch-DT.xlsx` uses multiple actions sets to filter data matching the search criteria.

Figure 8–2 Action Sets of Filter Customers Ribbon Command

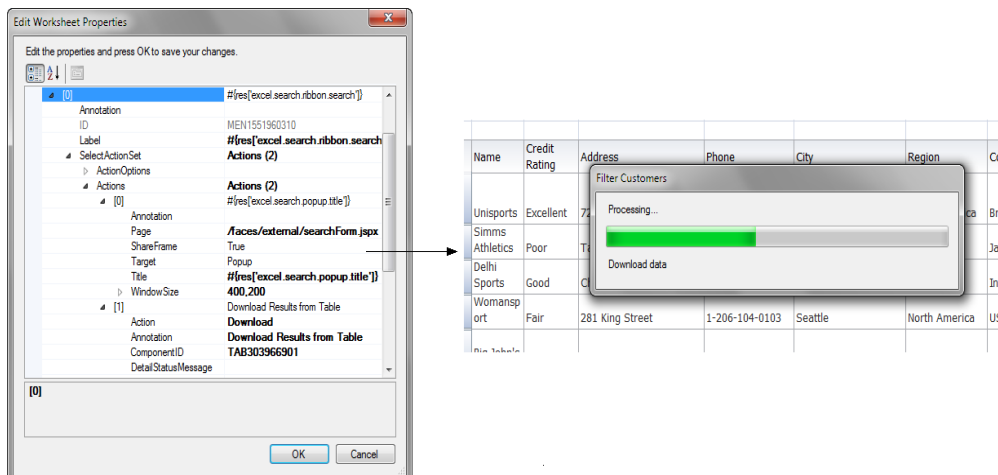
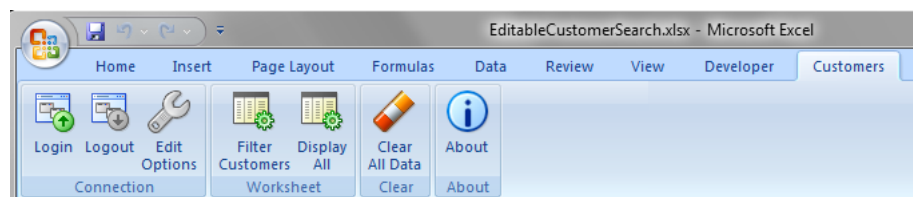


Figure 8–3 shows an example of custom runtime ribbon tab implemented in `EditCustomerSearch.xlsx`.

Figure 8–3 Runtime Ribbon Tab of `EditCustomerSearch.xlsx`



8.1.2 Additional Functionality for Adding Interactivity to an Integrated Excel Workbook

In addition to action sets and runtime ribbon tab, you can add additional functionality to configure your workbook. Following are links to other functionalities that you can use:

- **Display Web Pages:** You can display pages from the Fusion web application with which you integrate your Excel workbook. For more information, see [Section 8.4, "Displaying Web Pages from a Fusion Web Application."](#)
- **Dependent List of Values:** You can configure an ADF List of Values component as a dependent list of values component whose values are determined by another list of values component. For more information, see [Section 8.8, "Creating Dependent Lists of Values in an Integrated Excel Workbook."](#)
- **Styles:** You can configure the display of your form-type components using several predefined Excel styles. For more information, see [Section 9.2, "Working with Styles."](#)
- **Macros:** Use macros and Excel formulas to manage the data that you want to download from or upload to your Fusion web application. For more information, see [Section 8.10, "Using Calculated Cells in an Integrated Excel Workbook,"](#) and [Section 8.11, "Using Macros in an Integrated Excel Workbook."](#)

8.2 Using Action Sets

An *action set* is an ordered list of one or more actions that run in a specified order. The types of actions are as follows:

- `ADFmAction`
- `ComponentAction`
- `WorksheetMethod`
- `Confirmation`
- `Dialog`

An action set can be invoked by an end-user's gesture (for example, clicking an ADF Button) or an Excel worksheet event. Where an end-user gesture invokes an action set, the name of the action set property in the ADF component's property inspector is prefaced by the name of the gesture required. The following list describes the property names that ADF Desktop Integration displays in property inspectors, and what user gesture can invoke an action set:

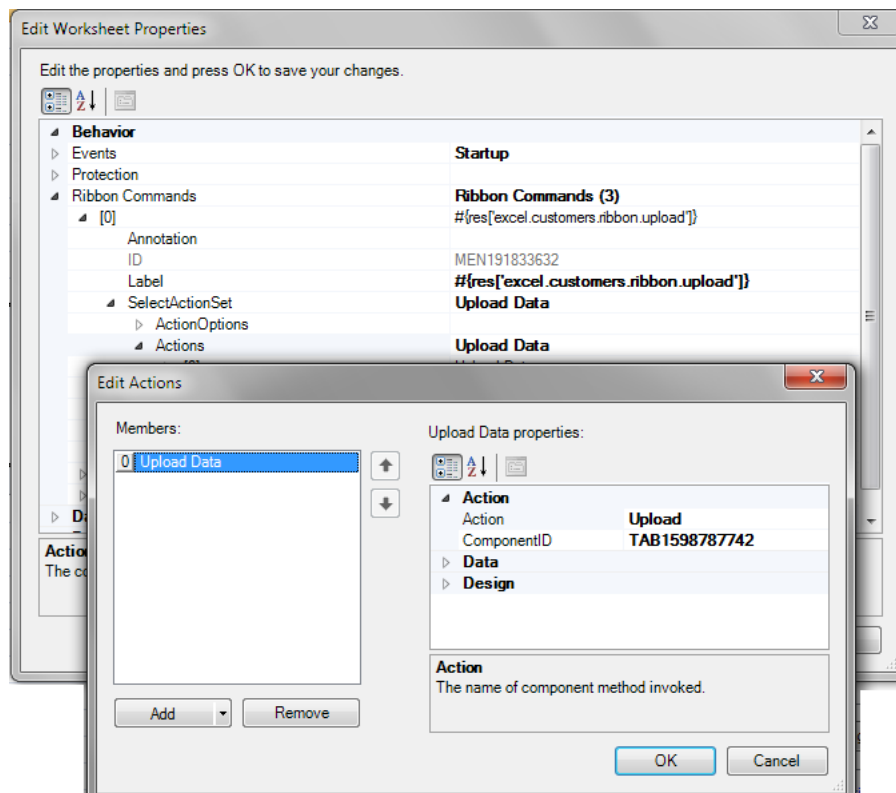
- `ClickActionSet` for an ADF Button component, as the end user clicks the button to invoke the associated action set

- DoubleClickActionSet for an ADF Input Text or ADF Output Text component, as the end user double-clicks these components to invoke the associated action set
- SelectActionSet for a worksheet ribbon button, as the end user selects a button to invoke the associated action set
- ActionSet for a worksheet event, as no explicit end-user gesture is required to invoke the action set

You invoke the Edit Action dialog from an ADF component, worksheet ribbon button, or worksheet event to define or configure an action set. In addition to defining the actions that an action set invokes, you can configure the action set's `Alert` properties to provide feedback on the result of invocation of an action set. You configure the `Status` properties for an action set to display a status message to end users while an action set runs the actions you define. For information about opening the Edit Action dialog, see [Section 5.12, "Using the Collection Editors."](#)

The Summit sample application for ADF Desktop Integration provides many examples of action sets in use. One example is the ribbon command labeled **Upload** at runtime in the `EditCustomers-DT.xlsx` workbook. An action set has been configured for this ribbon command that invokes the ADF Table component's `Upload` action illustrated by [Figure 8-4](#) which shows the Edit Action dialog in design mode.

Figure 8-4 Action Set for Upload Data Button in the `EditCustomers-DT.xlsx` Workbook



Tip: Write a description in the **Annotation** field for each action that you add to the Edit Action dialog. The description you write appears in the **Members** list view and, depending on how you write it, may be more meaningful than the default entry that ADF Desktop Integration generates.

Note: ADF Desktop Integration invokes the actions in an action set in the order that you specify in the **Members** list view.

8.2.1 How to Invoke a Method Action Binding in an Action Set

You can invoke multiple method action bindings in an action set. Page definition files define what action bindings are available to invoke in a worksheet that you integrate with your Fusion web application. For more information about page definition files and action bindings in an integrated Excel workbook, see [Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook."](#)

You use the Edit Action dialog to specify a method action binding to invoke.

Before you begin:

It may be helpful to have an understanding of action sets. For more information, see [Section 8.2, "Using Action Sets."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 8.1.2, "Additional Functionality for Adding Interactivity to an Integrated Excel Workbook."](#)

To invoke a method action binding in an action set:

1. Open the integrated Excel workbook.
2. Open the Edit Action dialog and invoke the dropdown list from the **Add** button illustrated here.



3. Select **ADFmAction** and configure its properties as described in the following list:
 - **ActionID**
Click the browse (...) icon beside the input field for **ActionID** to invoke the Binding ID picker and select the method action binding that the action set invokes.
 - **Annotation**
Optionally, enter a comment about the purpose of the action that you are configuring. The value you set for this property has no functional impact.
4. Click **OK**.

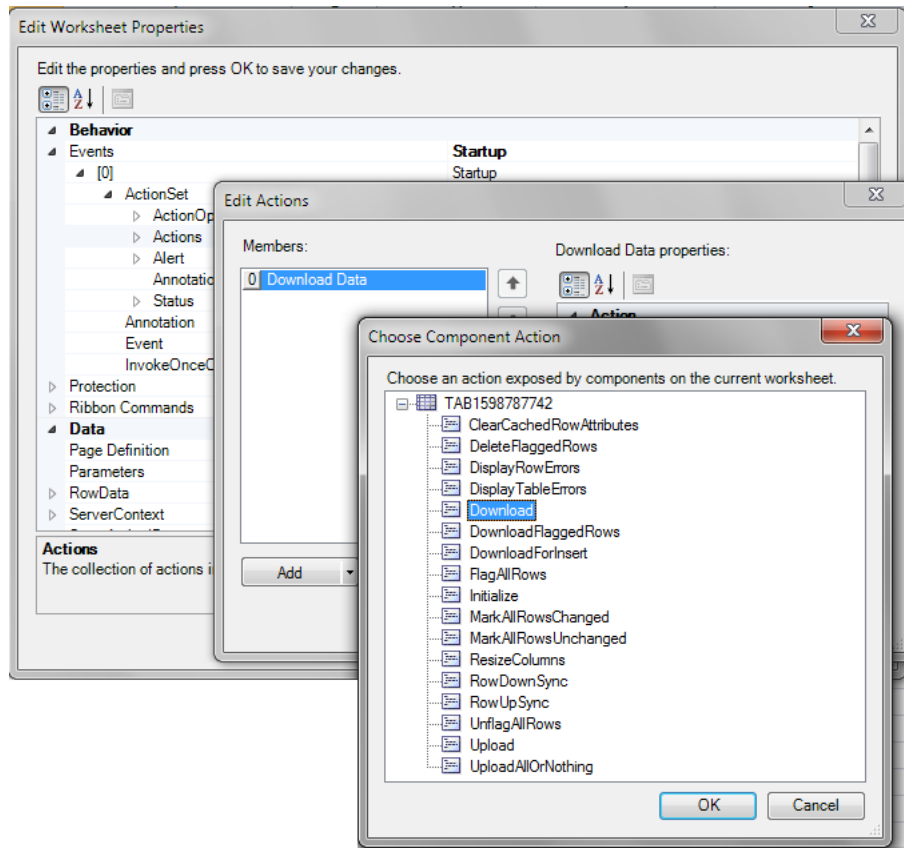
8.2.2 How to Invoke Component Actions in an Action Set

The ADF Table and the ADF Read-only Table components in ADF Desktop Integration expose actions that can be used to manage the transfer of data between Excel worksheets that you integrate with a Fusion web application. The ADF Read-only Table component exposes one component action, **Download**, while the ADF Table component exposes many other actions. More information about the actions for both components can be found in [Appendix A, "ADF Desktop Integration Component Properties and Actions."](#)

You configure action sets to invoke one or more component actions by referencing the component action in the array of actions. For example, [Figure 8–5](#) shows the Choose Component Action dialog where the actions exposed by the ADF Table and ADF

Read-only Table components present in a worksheet can be invoked by a `SelectActionSet` action set.

Figure 8–5 Choose Component Method Dialog



Note: An Excel worksheet must include an ADF Table or ADF Read-only Table component before one or more of these components' actions can be invoked by an action set.

Before you begin:

It may be helpful to have an understanding of action sets. For more information, see [Section 8.2, "Using Action Sets."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 8.1.2, "Additional Functionality for Adding Interactivity to an Integrated Excel Workbook."](#)

To invoke a component action from an action set:

1. Open the integrated Excel workbook.
2. Open the Edit Action dialog and invoke the dropdown list from the **Add** button illustrated here.



3. Select **ComponentAction** and configure its properties as described in the following list:
 - **ComponentID**
Click the browse (...) icon beside the input field for ComponentID to invoke the Choose Component Method dialog and select the component action that the action set invokes at runtime. This populates the ComponentID and Method input fields.
 - **Action**
The component's action that the action set invokes at runtime.
 - **Annotation**
Optionally, enter a comment about the purpose of the action that you are configuring. The value you set for this property has no functional impact.
 - **DetailStatusMessage**
Specify an optional literal value or EL expression that appears in the Status Message window (see [Section 8.2.5, "How to Display a Status Message While an Action Set Runs"](#)).
4. Click **OK**.

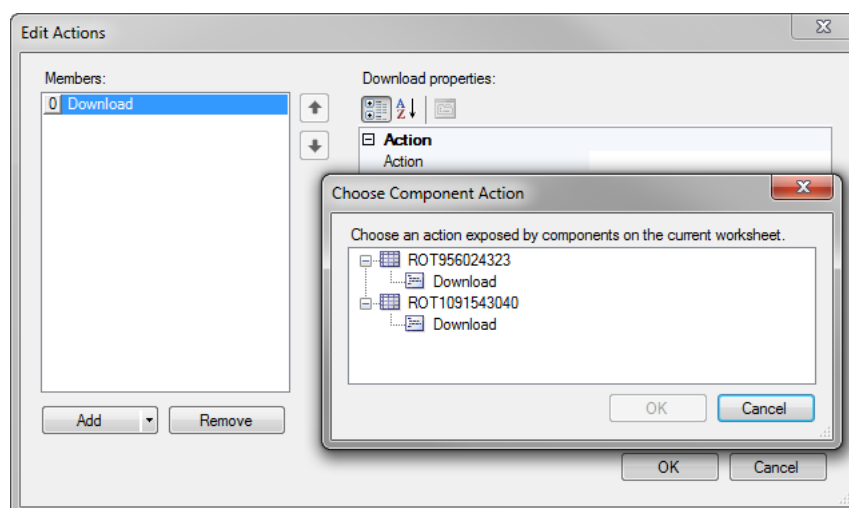
8.2.3 What You May Need to Know About an Action Set Invoking a Component Action

Note the following pieces of information about the behavior of action sets in integrated Excel workbooks.

8.2.3.1 Verifying an Action Set Invokes the Correct Component Action

When creating an action set, ensure that you invoke the component action from the correct instance of a component when a worksheet includes multiple instances of an ADF Read-only Table or ADF Table component. [Figure 8–6](#) shows the Choose Component Action dialog displaying two instances of the ADF Read-only Table component. Use the value of the ComponentID property described in [Table A–1](#) to correctly identify the instance of a component on which you want to invoke a component action.

Figure 8–6 Choose Component Action Dialog



8.2.3.2 Invoking Action Sets in a Disconnected Workbook

End users can use integrated Excel workbooks while disconnected from a Fusion web application, as described in [Chapter 15, "Using an Integrated Excel Workbook Across Multiple Web Sessions and in Disconnected Mode."](#) Some component actions, such as the Download action of the ADF Table component, require a connection to the Fusion web application to complete successfully. If the end user invokes an action set that includes such a component action, the integrated Excel workbook attempts to connect to the Fusion web application and, if necessary, invokes the authentication process described in [Section 11.2, "Authenticating the Excel Workbook User."](#)

8.2.4 How to Invoke an Action Set from a Worksheet Event

ADF Desktop Integration provides several worksheet events that, when triggered, can invoke an action set. The following worksheet events can invoke an action set:

- Startup
- Shutdown

Do not invoke a Dialog action from this event if the Dialog action's Target property is set to TaskPane.

- Activate
- Deactivate

You add an element to the array of events (`WorksheetEvent` list) referenced by the `Events` worksheet property. You specify an event and the action set that it invokes in the element that you add. For more information about the `Events` worksheet property and the worksheet events that can invoke an action set, see [Table A-22](#). See [Table A-17](#) for more information about action sets.

Use the Edit Events dialog to specify an action set to be invoked by a worksheet event.

Before you begin:

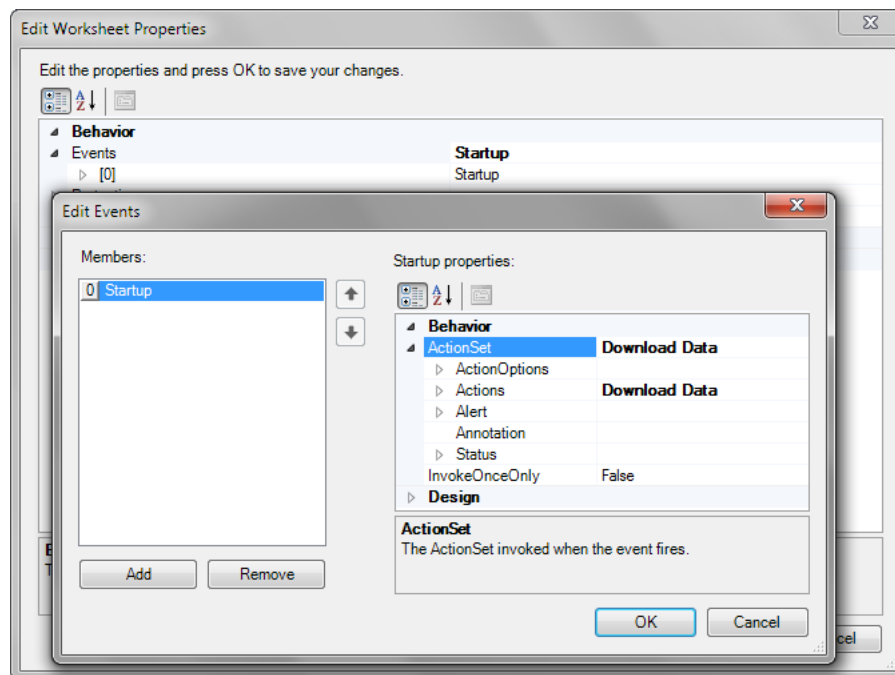
It may be helpful to have an understanding of action sets. For more information, see [Section 8.2, "Using Action Sets."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 8.1.2, "Additional Functionality for Adding Interactivity to an Integrated Excel Workbook."](#)

To invoke an action set from a worksheet event:

1. Open the integrated Excel workbook.
2. In the Workbook group of the Oracle ADF tab, click **Worksheet Properties**.
3. In the Edit Worksheet Properties dialog, click the browse (...) icon beside the input field for the `Events` property.
4. In the Edit Events dialog, click **Add** to add a new element that specifies an event and a corresponding action set that the event invokes.

[Figure 8-7](#) shows an example from the `EditCustomers-DT.xlsx` file where the worksheet event, `Startup`, invokes an action set that invokes the ADF Table component's `Download` action.

Figure 8–7 Worksheet Startup Event Invokes an Action Set

5. Click OK.

8.2.5 How to Display a Status Message While an Action Set Runs

You can display a status message and visual progress bars to end users while an action set runs by specifying values for the `Status` properties in an action set.

While using the `Status` properties in an action set, you can provide visual indication of the progress through progress bars. The `Mode` attribute of the `Status` properties enables you to choose the visual appearance of the progress bars at runtime. There are two types of progress bars available: main progress bar and detail progress bar. The *main progress bar* indicates the progress through the actions in an action set, and the *detail progress bar* indicates the progress of the current action.

You use the Edit Action dialog to configure values for the `ActionSet.Status` properties.

Before you begin:

It may be helpful to have an understanding of action sets. For more information, see [Section 8.2, "Using Action Sets."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 8.1.2, "Additional Functionality for Adding Interactivity to an Integrated Excel Workbook."](#)

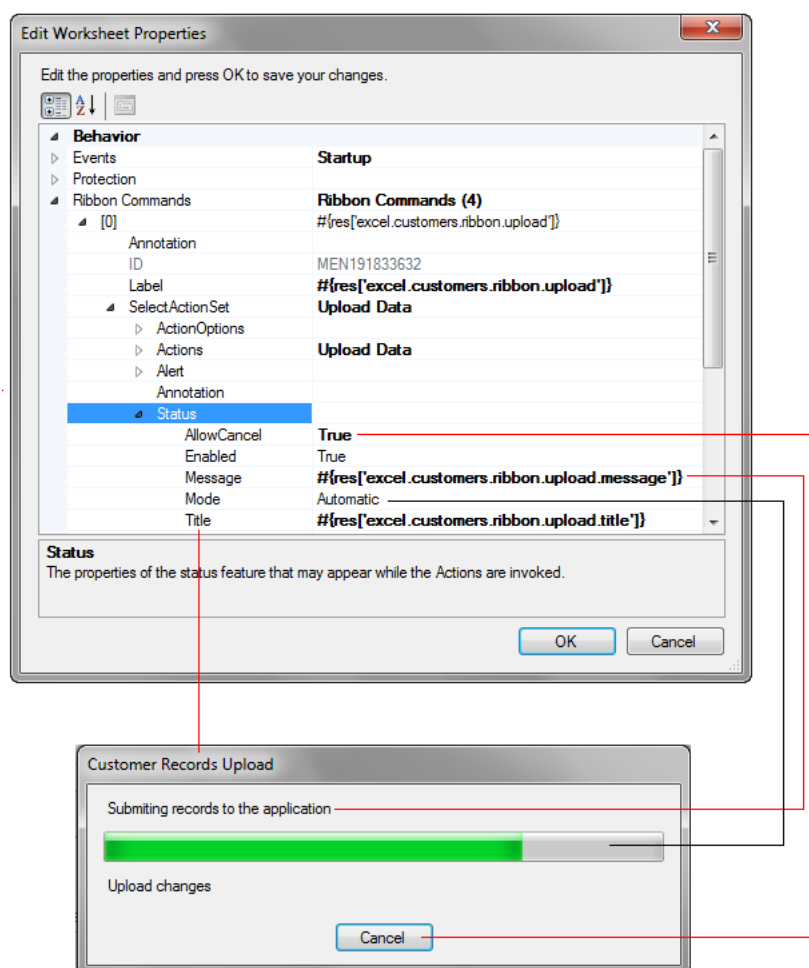
To display a status message:

1. Open the integrated Excel workbook.
2. Open the Edit Action dialog of the component.
3. Set values for the properties in the `Status` group of properties as described in the [Table 8–1](#).

Table 8–1 Status Group of Properties

For this property...	Enter or select this value...
AllowCancel	<p>True to display the Cancel button in the status dialog box.</p> <p>It indicates whether the action set execution can be canceled by the end user.</p> <p>For more information about the Cancel button, see Section 8.2.8, "What You May Need to Know About Canceling an Action."</p>
Enabled	True to display a status message. True is the default value.
Message	<p>An optional EL expression or literal value that resolves to the status message to display at runtime.</p> <p>For example, the Upload button in the <code>EditCustomers-DT.xlsx</code> file has the following EL expression configured for the <code>Message</code> property:</p> <pre><code>#{res['excel.customers.ribbon.upload.message']}</code></pre>
Title	<p>An optional EL expression or literal value that resolves to the title of the status message to display at runtime.</p> <p>For example, the Upload button in the <code>EditCustomers-DT.xlsx</code> file has the following EL expression configured for the <code>Title</code> property:</p> <pre><code>#{res['excel.customers.ribbon.upload.title']}</code></pre>
Mode	<p>Choose the visual appearance of progress bars.</p> <ul style="list-style-type: none"> ■ <code>Automatic</code>: ADF Desktop Integration analyzes the action set to determine which progress bars to display. ■ <code>BothBarsAlways</code>: Shows both main and detail progress bars. ■ <code>MainBarOnly</code>: Shows one progress bar only. The bar displays progress through the list of actions. ■ <code>DetailBarOnly</code>: Shows one progress bar only. The bar displays progress of the current action. ■ <code>MainMessageOnly</code>: None of the progress bars are shown.

[Figure 8–8](#) shows the property values, along with their corresponding visual elements, configured for the Status group of properties of the **Upload** ribbon command in the `EditCustomers-DT.xlsx` workbook.

Figure 8–8 Status Message Properties in an Action Set

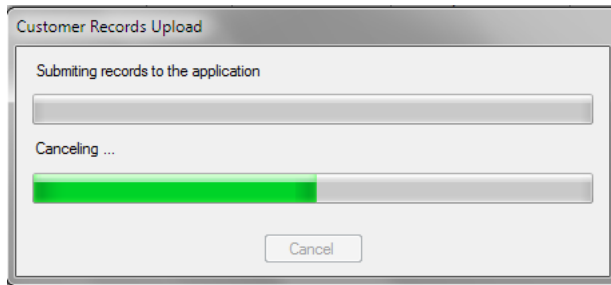
For more information about the Status group of properties, see the entry for Status in [Table A-17](#).

You can also use the optional `DetailStatusMessage` property to provide additional information to the user. For more information about the `DetailStatusMessage` property, see [Section 8.2.2, "How to Invoke Component Actions in an Action Set."](#)

4. Click OK.

8.2.6 What Happens at Runtime: How the Action Set Displays a Status Message

When an action set is invoked, a status message appears if the Status properties are configured to display a status message. [Figure 8–9](#) shows the status message that appears at runtime when the action set configured for the **Upload** ribbon command in the `EditCustomers-DT.xlsx` workbook runs.

Figure 8–9 Runtime View of Status Message

At runtime, if the value of the `Message` property is empty, ADF Desktop Integration provides a default, localized value. If the `Title` property is empty, the label from the action set container (such as button or ribbon command) is used. If the label of the container is also empty, then the default value provided by ADF Desktop Integration is used.

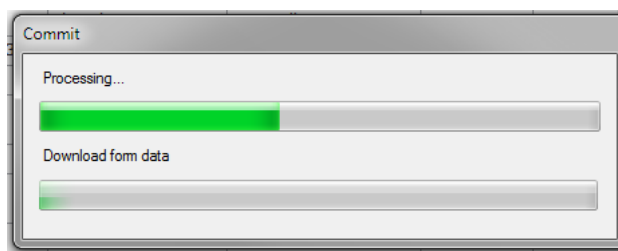
If `AllowCancel` property is set to `True`, the `Cancel` button appears allowing the end user to cancel the action set. If the end user cancels the action set, the `Cancel` button gets disabled, a warning message appears informing the user that the operation has been canceled, and the action set is aborted.

Tip: To cancel the operation of an action set, the end user can press the Space Bar key on the keyboard.

8.2.7 What You May Need to Know About Progress Bars

Note the following pieces of information about the progress bars:

- The progress bar window hides automatically when an action (such as alert, confirm, dialog, or upload options) prompts for user input.
- Some action types, such as `ADFmAction`, do not support the display of incremental progress in the detail bar. For example, [Figure 8–10](#) shows the progress bar of the `Commit` action with `Mode` set to `BothBarsAlways`. Notice that the detail bar appears, but does not show any progress.

Figure 8–10 Progress Bar for ADFmAction Type

- In the `Automatic` mode, if the action set has less than three actions, the status message dialog shows the detail progress bar only. If the action set has three or more actions, the dialog always shows the main bar, but the detail progress bar is shown only if any of the actions in the action set is capable of incremental progress. If none of the actions is capable of incremental progress, the detail bar is suppressed.
- If required, you can display the detail progress bar without the displaying the main progress bar. Such a configuration may be useful for an action set with a few

quick actions and one long action, for example, run a query and then download data.

- For very quick action sets (for example, `DisplayWorksheetErrors`), the best practice is to disable the status message.

8.2.8 What You May Need to Know About Canceling an Action

Each action in an action set can be categorized as non-interruptible, interruptible, or dialog.

The *non-interruptible* actions are atomic and cannot be canceled, or interrupted, during their operation. The following actions are non-interruptible:

- Worksheet actions: `UpSync`, `DownSync`
- ADFm action
- Table actions: `RowUpSync`, `RowDownSync`, `ClearCachedRowAttributes`, `FlagAllRows`, `UnflagAllRows`, `MarkAllRowsChanged`, `MarkAllRowsUnchanged`, `Initialize`

If the Cancel button is clicked while a non-interruptible action is running, the following happens:

1. The current action completes.
2. The action set is then aborted, and is not treated as a failure.
3. `ActionSet.Alert` is skipped.
4. The success, or failure, actions configured for the action set do not run.
5. The value in `Worksheet.Errors` is not changed.

The *interruptible* actions can be canceled during their operation. The following Table actions are interruptible:

- `Upload`, `UploadAllOrNothing`
- `Download`, `DownloadFlaggedRows`, `DownloadForInsert`
- `DeleteFlaggedRows`

If the Cancel button is clicked while an interruptible action is running, the following happens:

1. The current operation halts without completing.
2. The table is cleaned up:
 - `Upload` action: For rows that were successfully uploaded before the Cancel button was clicked, the Changed column cell flags are cleared or are left as is, and `CommitBatchActionID` action runs. If a row failed during upload, the Changed column cell is not affected and error status is displayed. The rows that did not get uploaded continue to display the changed status in the Changed column and the Status column remains untouched.
 - `UploadAllOrNothing` action: The `CommitBatchActionID` action does not run. The Changed column flags for all rows remain set. Failed rows display error message. Successfully uploaded rows have Status cells and error rows unpopulated.
 - `Download`, `DownloadForInsert` action: Rows that were downloaded before the Cancel button was clicked are left as is and are not removed. The table is then sized accordingly.

- DownloadFlaggedRows action: Flagged rows that were downloaded before the Cancel button was clicked have their flag cells cleared. The remaining flagged rows continue to display the flag status.
 - DeleteFlaggedRows action: The rows that were deleted on server before the Cancel button was clicked are removed from the worksheet. The remaining flagged rows continue to display the flag status.
3. `Table.FailureActionID` does not run.
 4. Remaining actions in the action set are skipped.
 5. `Table.Errors` and `Worksheet.Errors` are left as they were.

The *dialog* actions show modal dialogs which can be canceled or closed. The Action Set Status Message dialog is not displayed during the execution of one of these actions. The following actions are dialog type:

- Confirmation
- Dialog
- `DisplayWorksheetErrors`, `DisplayRowErrors`, `DisplayTableErrors`

8.2.9 How to Provide an Alert After the Invocation of an Action Set

You can display an alert message to end users that notifies them when an action set operation completes successfully or fails. For example, you can display a message when all actions in an action set succeed or when there was at least one failure. The `ActionSet.Alert` group of properties configures this behavior.

Note: An alert message does not appear if the end user cancels the execution of an action set. For example, you configure an alert message to appear after an action set that invokes a web page in a popup dialog completes execution. At runtime, the end user cancels execution of the action set by closing the popup dialog using the close button of the Excel web browser control that hosts the popup dialog. In this scenario, no alert message appears. For more information about displaying web pages, see [Section 8.4, "Displaying Web Pages from a Fusion Web Application."](#)

You use the Edit Action dialog to configure values for the `ActionSet.Alert` group of properties.

Before you begin:

It may be helpful to have an understanding of action sets. For more information, see [Section 8.2, "Using Action Sets."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 8.1.2, "Additional Functionality for Adding Interactivity to an Integrated Excel Workbook."](#)

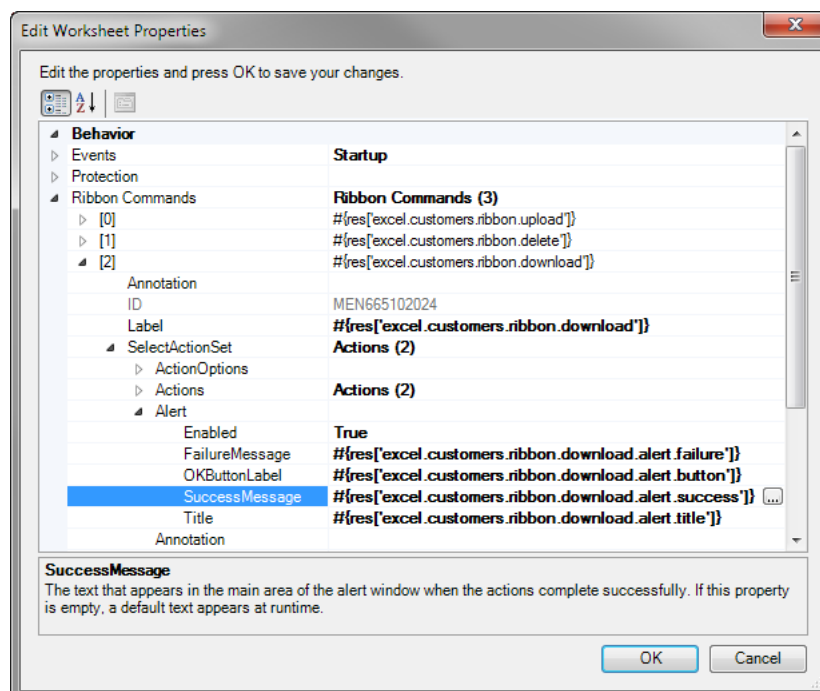
To add an alert to an action set:

1. Open the integrated Excel workbook.
2. Open the Edit Action dialog.
3. Set values for the properties in the `ActionSet.Alert` group of properties as described in [Table 8-2](#).

Table 8–2 ActionSet.Alert Group of Properties

For this property...	Enter or select this value...
Enabled	Select True from the dropdown list to display an alert message once the action set completes. The default value is False .
FailureMessage	Specify an optional EL expression or literal value that evaluates to a message to appear in the dialog if errors occur during execution of the action set. For example, the Download button in the <code>EditCustomers-DT.xlsx</code> workbook has the following value configured for the <code>FailureMessage</code> property: <pre>#{res['excel.customers.ribbon.download.alert.failure']}</pre> The Download button invokes an action set that, in turn, invokes the ADF Table component's <code>Download</code> action. The EL expression specified for <code>FailureMessage</code> retrieves error messages if the <code>Download</code> action encounters errors. For more information about error handling, see Section 12.4, "Error Reporting in an Integrated Excel Workbook."
OKButtonLabel	Specify an optional EL expression or literal value that evaluates to a message to appear in the OK button of the dialog.
SuccessMessage	Specify an optional EL expression or literal value that evaluates to a message to appear in the dialog if no errors occur during the execution of the action set. For example, the Download button in the <code>EditCustomers-DT.xlsx</code> workbook has the following value configured for the <code>SuccessMessage</code> property: <pre>#{res['excel.customers.ribbon.download.alert.success']}</pre>

[Figure 8–11](#) shows the values configured for a ribbon command's `Alert` group of properties in the `EditCustomers-DT.xlsx` workbook. This ribbon command is labeled **Download** at runtime.

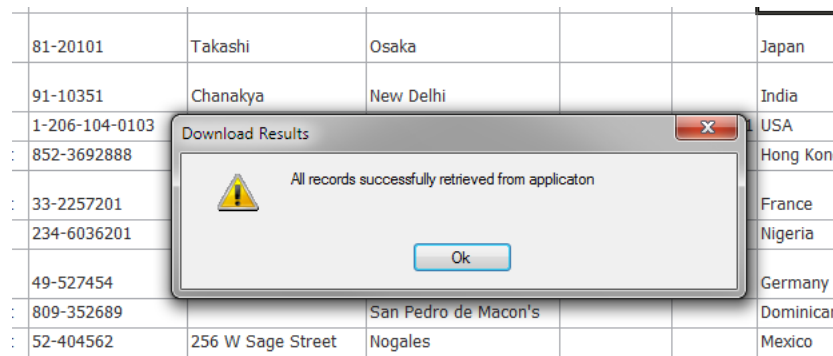
Figure 8–11 Alert Message Properties in an Action Set

4. Click OK.

8.2.10 What Happens at Runtime: How the Action Set Provides an Alert

Figure 8–12 shows the alert message that appears at runtime when the action set invoked by the ADF Button component labeled **Download** successfully completes execution.

Figure 8–12 Runtime View of an Alert Message



At runtime, if the value of the `FailureMessage`, `OKButtonLabel`, or `SuccessMessage` property is empty, ADF Desktop Integration provides a default, localized value.

8.2.11 How to Configure Error Handling for an Action Set

You specify values for an action set's `ActionOptions` properties to determine what an action set does if one of the following events occurs:

- An action in the action set fails
- All actions in the action set complete successfully

For information about how to invoke these editors, or about an ADF component's property inspector, see [Chapter 5, "Getting Started with the Development Tools."](#) More information about action set properties can be found in [Table A.14](#).

Before you begin:

It may be helpful to have an understanding of action sets. For more information, see [Section 8.2, "Using Action Sets."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 8.1.2, "Additional Functionality for Adding Interactivity to an Integrated Excel Workbook."](#)

To configure error handling for an action set:

1. Open the integrated Excel workbook.
2. Open the appropriate editor or property inspector and configure values for the action set's `ActionOptions` properties as described in the [Table 8–3](#).

Table 8–3 ActionOptions Properties

Set this property...	To...
AbortOnFailure	True (default value) so that the action set does not run any further actions if the current action fails. When set to <code>False</code> , the action set runs all actions regardless of the success or failure of previous actions.
FailureActionID	Specify an ADF Model action to invoke if an action set does not complete successfully. For example, you can specify an ADF Model action that rolls back changes made during the unsuccessful invocation of the action set. Note that calling an action set that changes a record set's currency during the execution of <code>FailureActionID</code> methods is not supported. The <code>Rollback</code> method also should not be specified as the <code>FailureActionID</code> in an action set.
SuccessActionID	Specify an ADF Model action to invoke if an action set completes successfully. For example, you can specify an action binding that runs a commit action. A value for this property is optional and you can specify a final action, such as an action binding that runs a commit action, in the action set itself. Note that calling an action set that changes a record set's currency during the execution of <code>SuccessActionID</code> methods is not supported.

3. Click **OK**.

8.2.12 How to Invoke a Confirmation Action in an Action Set

The Confirmation action presents the end user with a simple message dialog that displays the title and prompt message specified in the Confirmation action properties.

The execution of the action set pauses until the end user clicks one of the two buttons provided. If the user clicks **OK**, the action sets proceed with the remaining actions in the Action Set. If the user clicks **Cancel**, the action set is aborted at that point and the remaining actions are not invoked. As there is no error or success, the `FailureActionID` or `SuccessActionID` action is not invoked.

Before you begin:

It may be helpful to have an understanding of action sets. For more information, see [Section 8.2, "Using Action Sets."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 8.1.2, "Additional Functionality for Adding Interactivity to an Integrated Excel Workbook."](#)

To invoke a Confirmation action from a component

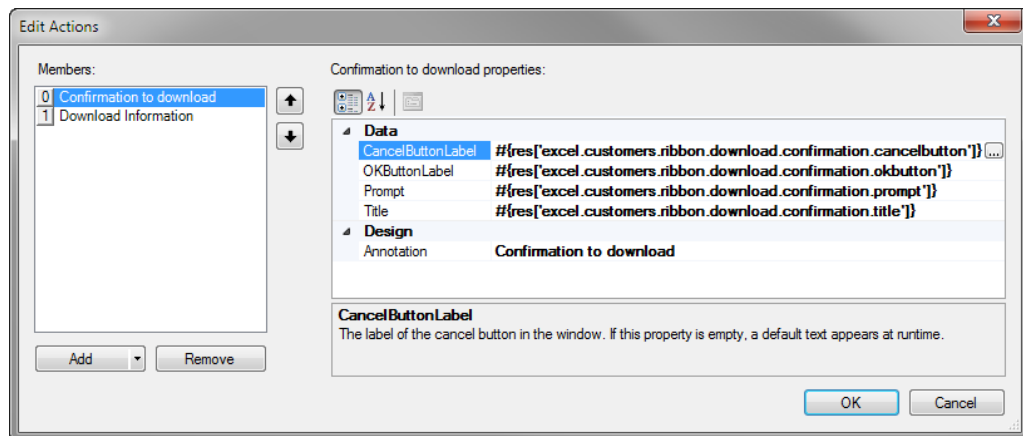
1. Open the integrated Excel workbook.
2. Open the Edit Action dialog and click the down arrow in the **Add** button to open a dropdown list, as illustrated here.



3. Select **Confirmation** and configure its Data properties as described in the following list:
 - `CancelButtonLabel`
Specify an optional EL expression or literal value that evaluates to a message to appear in the **Cancel** button of the dialog.
 - `OKButtonLabel`
Specify an optional EL expression or literal value that evaluates to a message to appear in the **OK** button of the dialog.
 - `Prompt`
Specify an optional EL expression or literal value that evaluates to a message to appear as the prompt of the dialog.
 - `Title`
Specify an optional EL expression or literal value that evaluates to a title of the confirmation dialog to display at runtime.
4. Optionally, enter a comment in the `Annotation` property about the purpose of the action that you are configuring. The value you set for this property has no functional impact.
5. Click **OK**.

Figure 8–13 shows the Edit Action dialog with default attribute values for the **Download** ribbon command.

Figure 8–13 Confirmation Action Attributes

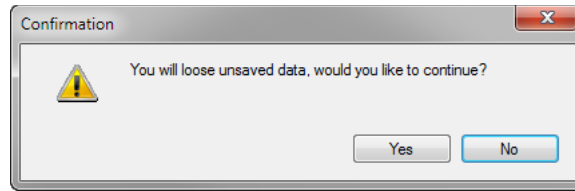


8.2.13 What Happens at Runtime: How the Action Set Provides a Confirmation

Once the action set is invoked, the user is prompted with a confirmation dialog. If the user clicks **OK**, the next action operation is performed; and if the user clicks **Cancel**, the Action Set execution terminates without an error.

Note: If the user cancels a Confirmation action, the `FailureActionID` binding does not run.

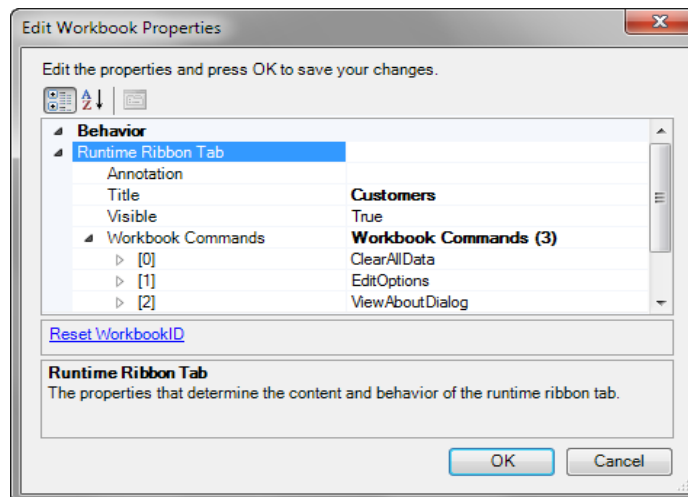
Figure 8–14 shows a default Confirmation dialog with Yes and No buttons.

Figure 8–14 Confirmation Dialog

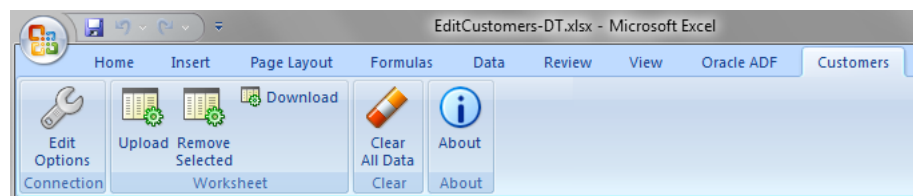
At runtime, if the value of the `CancelButtonLabel`, `OKButtonLabel`, or `Prompt` property is empty, ADF Desktop Integration provides a default, localized value. If the `Title` property is empty, the label from the action set container (such as button or ribbon command) is used. If the label of the container is also empty, then the default value provided by ADF Desktop Integration is used.

8.3 Configuring the Runtime Ribbon Tab

You can configure the runtime ribbon tab in the Excel Ribbon with items that invoke Oracle ADF functionality in your integrated Excel workbook. In the Runtime Ribbon Tab group, setting the `Visible` workbook property to `True` makes this tab appear at runtime. The `Title` property determines the title of the tab that the end user sees at runtime, as illustrated in [Figure 8–15](#).

Figure 8–15 Workbook Properties for Runtime Ribbon Tab

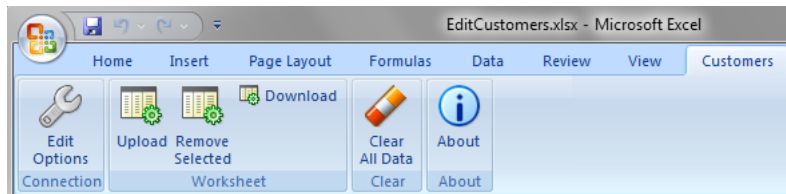
At runtime, the tab appears as the last tab in the Ribbon and all your configured commands appear in various groups of the tab, as illustrated by [Figure 8–16](#).

Figure 8–16 Runtime View of the Ribbon Tab

[Figure 8–17](#) illustrates the runtime ribbon tab in `EditCustomers.xlsx` with two commands configured for worksheet. At runtime, the commands are divided into four

groups: items that invoke commands on the workbook, items that invoke commands on the current worksheet, a command group to clear all data, and a command workgroup to display ADF Desktop Integration version information.

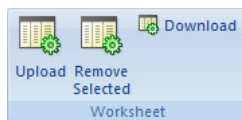
Figure 8–17 Runtime View of Ribbon Tab in EditCustomers.xlsx



You configure the `Workbook Commands` property in the properties of the workbook so that the runtime ribbon tab contains commands that allow the end user to invoke workbook actions. You configure the `Ribbon Commands` property in the properties of the worksheet so that the ADF Desktop Integration tab contains items allowing a user to invoke an action set. Worksheet command items appear when the worksheet is active. If you remove a workbook command, it does not appear in the runtime tab for that workbook. If you remove all the commands for a given group, the group does not appear when that workbook is active.

Figure 8–18 shows the Worksheet group at runtime where the worksheet actions, that invoke `SelectActionSet` action sets, appear.

Figure 8–18 Runtime Worksheet Group



8.3.1 How to Define a Workbook Command Button for the Runtime Ribbon Tab

To define a workbook command button for the runtime ribbon tab, you configure some workbook properties. The following procedure shows how to create or remove an item in the Workbook group by using the workbook action, `Login`, as an example.

Before you begin:

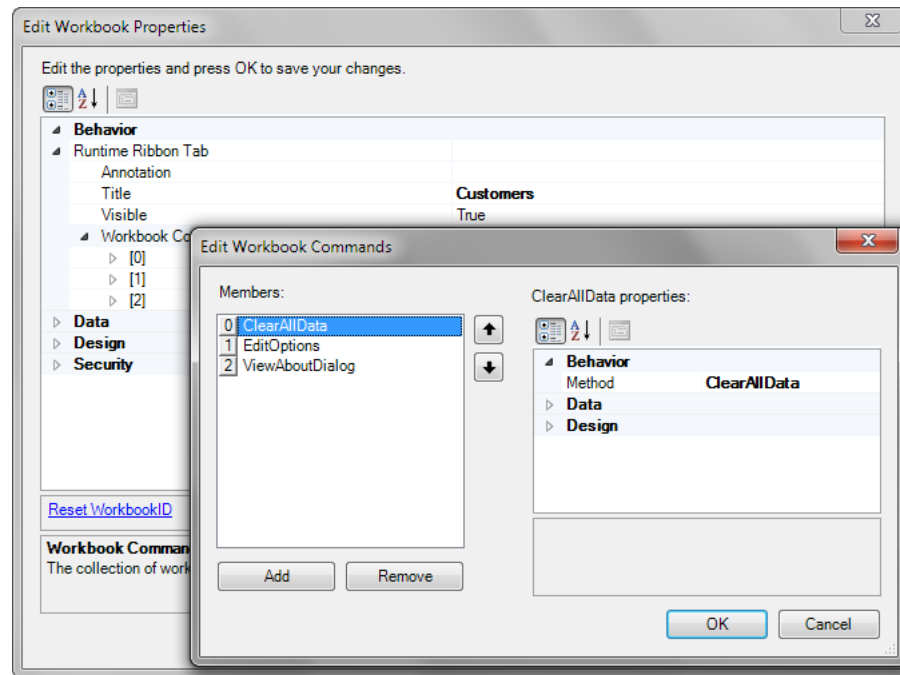
It may be helpful to have an understanding of the runtime ribbon tab in Excel. For more information, see [Section 8.3, "Configuring the Runtime Ribbon Tab."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 8.1.2, "Additional Functionality for Adding Interactivity to an Integrated Excel Workbook."](#)

To define a workbook command button:

1. Open the integrated Excel workbook.
2. In the Workbook group of the Oracle ADF tab, click `Workbook Properties`.
3. In the `Edit Workbook Properties` dialog, expand **Runtime Ribbon Tab**, and select **Workbook Commands**. Click the browse (...) icon beside the `Workbook Commands` to display the dialog as illustrated in [Figure 8–19](#).

Figure 8–19 Edit Workbook Commands Dialog



4. Click **Add** and specify values for the properties of the workbook command buttons as follows:

- Method

Specify the workbook action that you want the workbook command button to invoke.

- Label

Enter a value in the input field that appears as the label at runtime.

Alternatively, invoke the expression builder by clicking the browse (...) icon and write an EL expression that resolves to a string value in a resource bundle.

Note that the runtime value that appears in the label cannot exceed 1024 characters. A runtime value that exceeds 1024 characters is truncated so that only 1024 characters appear.

For more information about using resource bundles, see [Section 10.2, "Using Resource Bundles in an Integrated Excel Workbook."](#)

For more information about labels, see [Section 9.4, "Using Labels in an Integrated Excel Workbook."](#)

5. Click **OK**.

Note: The order of workbook commands in the Edit Workbook Commands dialog is ignored at runtime. The order and grouping of the workbook-level commands is always the same.

8.3.2 How to Configure a Worksheet Command for the Runtime Ribbon Tab

To define a worksheet command, you configure properties for the worksheet using the property inspector. By default, no command buttons are defined for the Worksheet

group in the worksheet properties. You add members to the list that is referenced by the `Ribbon Commands` property in the properties of the worksheet.

CAUTION: Set the `Runtime Ribbon Tab.Visible` workbook property to `TRUE` to display command buttons. If the `Runtime Ribbon Tab.Visible` is set to `FALSE`, no command buttons appear. For more information about workbook properties, see [Table A-21](#).

Before you begin:

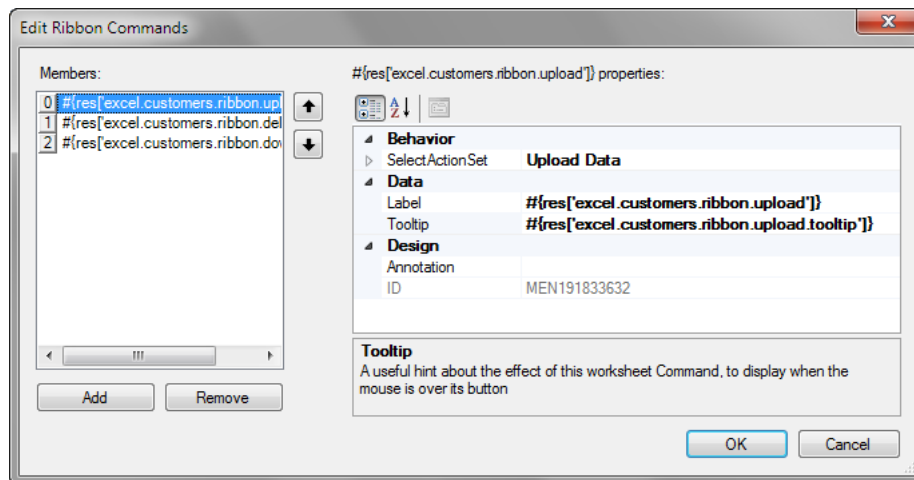
It may be helpful to have an understanding of the runtime ribbon tab in Excel. For more information, see [Section 8.3, "Configuring the Runtime Ribbon Tab."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 8.1.2, "Additional Functionality for Adding Interactivity to an Integrated Excel Workbook."](#)

To define a worksheet command button:

1. Open the integrated Excel workbook.
2. In the Workbook group of the Oracle ADF tab, click **Worksheet Properties**.
3. In the Edit Worksheet Properties dialog, click the browse (...) icon beside the input field for the `Ribbon Commands` property to invoke the editor, as illustrated in [Figure 8-20](#).

Figure 8-20 Edit Ribbon Commands Dialog



4. Click **Add** to add a new ribbon button in the **Members** list of the collection editor.
5. Configure the properties of `SelectActionSet` to specify the type of action(s) that the ribbon button invokes.
6. To add a tooltip, enter the EL expression in the **Tooltip** property.

Note that Ribbon command tooltips have a maximum size of 1024 characters. If a tooltip value exceeds that limit, only the first 1024 characters are shown.

7. Click **OK**.

[Figure 8-18](#) displays how the commands appear at runtime.

Notes:

- At runtime, the worksheet commands appear in the same order as they are defined in the Edit Ribbon Commands dialog.
- The ribbon controls of the toolbar are shared among all open integrated workbooks. If you open two, or more, workbooks using different ribbon buttons occupying the same location in the toolbar, Excel always shows the key tip of the first opened workbook in all open workbooks. This is an Excel limitation.
- When the user hovers the mouse over the ribbon command with the tooltip, a multi-part tooltip appears. The command label appears first in bold followed by the text from the `Tooltip` property. Below this text, the add-in name appears.

8.4 Displaying Web Pages from a Fusion Web Application

You configure a `Dialog` action in an action set to display pages from the Fusion web application with which you integrate your Excel workbook. These pages provide additional functionality for your integrated Excel workbook. Examples of additional functionality that you can provide include search dialogs and display pick dialogs that interact with your Fusion web application. You can also configure upload options.

The `Dialog` action in an action set can be configured to display in one of the following two types of dialog:

- Popup dialog
- Runtime task pane

The value for the `Dialog.Target` property (`Popup` or `TaskPane`) of the component's action set determines where a web page is rendered.

The value for the `Dialog.Page` property specifies the web page to display when the action is invoked. Valid values include a URL relative to the value of the `WebAppRoot` property or an absolute URL.

For example, the `EditableCustomerSearch-DT.xlsx` workbook specifies the following relative URL as a value for the page to invoke when a user clicks the **Filter Customers** ribbon command at runtime:

```
/faces/external/searchForm.jspx
```

Absolute URLs such as the following are also valid:

```
http://www.oracle.com/technetwork/middleware/fusion-middleware/overview/index.html
```

Note: The `Dialog` action does not support ADF task flows.

8.4.1 How to Display a Web Page in a Popup Dialog

You can configure a `Dialog` action in an action set to invoke a web page from your Fusion web application in a modal popup dialog hosted by Excel's web browser control. This feature provides end users with functionality that allows them to, for example, input values displayed by a page from the Fusion web application into the integrated Excel workbook.

The web page that the action set invokes must contain a reserved HTML `` element that has a case-sensitive ID attribute set to `ADFdi_CloseWindow`.

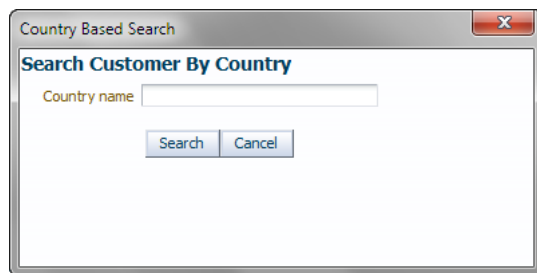
[Example 8–1](#) shows how you can automatically set the value of the span element using the rendered property of the `f:verbatim` tag.

Example 8–1 Use of Reserved HTML `` Element

```
<f:verbatim rendered="#{requestScope.searchAction eq 'search'}">
<span id="ADFdi_CloseWindow">Continue</span>
</f:verbatim>
<f:verbatim rendered="#{requestScope.searchAction eq 'cancel'}">
<span id="ADFdi_CloseWindow">Abort</span>
</f:verbatim>
```

[Figure 8–21](#) shows the `searchForm.jspx` page hosted by the `EditableCustomerSearch-DT.xlsx` workbook's browser control.

Figure 8–21 Search Popup Dialog



In scenarios where you cannot use the rendered property of the `f:verbatim` tag as outlined in [Example 8–1](#), you may need to:

1. Create a backing bean that exposes the `Dialog` action's result value as a property
2. Use an action listener to invoke the backing bean, and an EL expression in the `` element to set the value `ADFdi_CloseWindow` to the bean property value.

Whichever approach you take, ADF Desktop Integration monitors the value of `ADFdi_CloseWindow` to determine when to close the popup dialog. If the content of the `ADFdi_CloseWindow` `` element is:

- An empty string or is not present, the popup dialog remains open.
- Continue, the popup dialog closes and the action set invokes its next action.

The following example shows `ADFdi_CloseWindow` assigned a value of "Continue":

```
var closeWindowSpan = document.getElementById("ADFdi_CloseWindow");
closeWindowSpan.innerHTML = "Continue";
```

- Abort, the popup dialog closes and the action set stops running. No additional actions are invoked.
- Some other string value, the popup dialog remains open.

You set the `Target` property for a `Dialog` action to `Popup` to display a web page from the Fusion web application in a modal popup dialog hosted by Excel's web browser control. Displaying a web page in a modal popup dialog differs from displaying a web page in Excel's task pane, because the `Dialog` action that the action set invokes cannot continue execution until it receives user input. While the popup dialog is open, the

end user cannot interact with any other part of the integrated Excel workbook, as the popup dialog retains focus.

End users can navigate between multiple web pages from the Fusion web application within the browser control until they close the browser control, or ADF Desktop Integration closes it.

To immediately synchronize the changes that the end user makes to a data control through a popup dialog, specify the next action in the action set after the `Dialog` action to download all modified bindings to the worksheet (use the `DownSync` worksheet action) or ADF Table component (use the `RowDownSync` action). This scenario assumes that you specify `Continue` as the value for `ADFdi_CloseWindow`.

Notes:

- If you configure the web page that appears in the popup dialog so that the end user can download an integrated Excel workbook, the Oracle ADF functionality in the integrated Excel workbook is disabled when the end user opens the workbook after download.
- If you are using the HTML `<select>` components, such as list box or dropdown list, note that `<select>` components do not follow z-order configuration when the page is displayed through `Dialog` actions. In the .NET Web Browser control, on a web page with layered and overlapping components, the `<select>` components might appear on top of other components.
- If the **Title** property is left blank, the web page's title will be used as the dialog's window title.
- The value of the `ADFdi_CloseWindow` `` is monitored on every page transition in the browser control. When the value is `Continue`, the popup dialog closes and the action set continues to run. When the value is `Abort`, the popup dialog closes and no further actions in the action set run. If the `` element is not present, or the value is other than `Continue` or `Abort`, the popup dialog will remain open.

On each page transition, if the reserved `` element is present, client side Javascript can run and change the value of the element. If the value changes to `Continue` or `Abort`, the popup dialog also closes and has the same effect on the action set.

8.4.2 How to Display a Web Page in ADF Desktop Integration Runtime Task Pane

You can set the `Dialog.Target` property for an action to `TaskPane` to display a web page specified by the `Dialog.Page` property in the ADF Desktop Integration task pane. In contrast to displaying a web page in a popup dialog, displaying a web page in the task pane allows an action set to continue executing actions while the web page displays. End users can access and interact with other parts of the integrated Excel workbook while the web page displays.

Notes:

- If the **Title** property is left blank, the task pane's title will also remain blank.
 - If the **Target** property of a `Dialog` action is set to `TaskPane`, ADF Desktop Integration ignores the value of `ADFdi_CloseWindow` (and other elements).
-
-

8.4.3 What You May Need to Know About Displaying Pages from a Fusion Web Application

You can keep the data an integrated Excel workbook contains synchronized with a Fusion web application by specifying additional actions in the action set that invokes the `Dialog` action. You can ensure that the Fusion web application page and the integrated Excel worksheet both use the same data control frame by setting the `ShareFrame` property of the `Dialog` action.

Notes:

- If your custom web page is based on ADF Faces and opens a popup window, the web page must be configured in a certain way to work properly. On the command component, set the `windowEmbedStyle` to **inlineDocument**. For more information, see *Developing Web User Interfaces with Oracle ADF Faces*.
 - The `Dialog.Page` property does not accept EL expressions.
-
-

8.4.3.1 Keeping an Integrated Excel Workbook and a Fusion Web Application Synchronized

To ensure that data in the integrated Excel workbook and the Fusion web application remains synchronized while end users use pages from the Fusion web application, configure the action set that invokes the `Dialog` action to:

- Send changes from the integrated Excel workbook to the Fusion web application before invoking the `Dialog` action.

Invoke the `RowUpSync` worksheet action to synchronize changes from the current row in the ADF Table component. You may also invoke `UpSync` to synchronize changes in form components.

- Send changes from the Fusion web application to the integrated Excel workbook after invoking the `Dialog` action.

Invoke the `RowDownSync` worksheet action to send changes from the Fusion web application to the current row in the ADF Table component. You may also invoke `DownSync` to synchronize changes in form components.

For `DoubleClickActionSet`, you must ensure that the server-side model is in the same state after executing the action set as it was before executing the action set. In most cases, it is sufficient to roll back any and all uncommitted changes at the end of each `DoubleClickActionSet`, as there are no pending uncommitted changes when the action set execution begins.

For more information about synchronizing data between an integrated Excel workbook and a Fusion web application, see [Chapter 15, "Using an Integrated Excel Workbook Across Multiple Web Sessions and in Disconnected Mode."](#) For information

about worksheet actions and ADF Table component actions, see [Chapter A, "ADF Desktop Integration Component Properties and Actions."](#)

8.4.3.2 Sharing Data Control Frames Between Integrated Excel Worksheets and Fusion Web Application Pages

Fusion web applications and integrated Excel workbooks both use data control frames to manage the transactions and state of view objects and, by extension, the bindings exposed in a page definition file. When you invoke a Fusion web application's page from an integrated Excel worksheet, you can ensure that the page and the integrated Excel worksheet both use the same data control frame by setting the `ShareFrame` property of the `Dialog` action that invokes the page to `True`.

The `Page` property in the `Dialog` action specifies the page that the `Dialog` action invokes. If the `Dialog` action invokes an absolute URL or a page that is not part of your Fusion web application, ADF Desktop Integration ignores the value of `ShareFrame` if `ShareFrame` is set to `True`.

Set `ShareFrame` to `False` in the following scenarios:

- The `Dialog.Page` property in the action set references an absolute URL or a page that is not part of your Fusion web application.
- The `Dialog.Page` property in the action set references a page that is part of your Fusion web application, but that does not need to share information with the integrated Excel worksheet. For example, a page that displays online help information.

For more information about data control frames in a Fusion web application, see the "Sharing Data Controls Between Task Flows" section of the *Developing Fusion Web Applications with Oracle Application Development Framework*.

8.4.3.3 Configuring a Fusion Web Application for ADF Desktop Integration Frame Sharing

When you add the ADF Desktop Integration feature to your Fusion web application, the application is automatically configured to support ADF Desktop Integration frame sharing. Frame sharing allows each worksheet of an integrated Excel workbook to use a dedicated `DataControl` frame. Web pages displayed in dialogs invoked from each worksheet can then share the same `DataControl` frame as the integrated Excel worksheet.

To verify that your Fusion web application is configured to support frame sharing:

1. Open your Fusion web application project in JDeveloper.
2. In the Applications window, expand the Application Resources panel.
3. Open the `adf-config.xml` file available in **Descriptors** > **ADF META-INF** node.
4. Click the **Source** tab to open the source editor.
5. Confirm that the following `adf-desktopintegration-servlet-config` element is present in the file before the `</adf-config>` tag:

```
<adf-desktopintegration-servlet-config
xmlns="http://xmlns.oracle.com/adf/desktopintegration/servlet/config">
  <controller-state-manager-class>
    oracle.adf.desktopintegration.controller.impl.ADFcControllerStateManager
  </controller-state-manager-class>
</adf-desktopintegration-servlet-config>
```

6. Save the `adf-config.xml` file and close JDeveloper.

8.5 Adding a Custom Popup Picker Dialog to an ADF Table Column

You can configure the `DoubleClickActionSet` of an ADF Table component's column subcomponent (`UpdateComponent` or `InsertComponent`) to invoke a Fusion web application page that renders a pick dialog where the end user selects a value to insert in the ADF Table component column.

This functionality is useful when you want to constrain the values that end users can enter in an ADF Table component. For example, you may want a runtime ADF Table component column to be read-only in the Excel worksheet so that end users cannot manually modify values to prevent them from introducing errors. Invoking a pick dialog rendered by a Fusion web application page allows the end user to change values in the ADF Table component without entering incorrect data.

In addition to configuring the `DoubleClickActionSet`, you may configure the ADF Table component's `RowData.CachedAttributes` property to reference attribute binding values if you want:

- End users to modify values in the Fusion web application's page that you do not want to appear in the ADF Table component of the integrated Excel workbook
- An ADF Table component's column to be read-only in the integrated Excel workbook
- Cache data in an ADF Table component over one or more user sessions that is not visible to end users but is modified by a pick dialog

For example, an ADF Table component displays a list of product names to end users. A pick dialog is invoked that refreshes the list of product names in the ADF Table component and, as part of the process, sets the value of product IDs. In this scenario, you specify the attribute binding value for the product ID in the ADF Table component's `RowData.CachedAttributes` property. After the action set runs, the ADF Table component displays the refreshed list of product names in the rows of the Excel worksheet and references the associated product IDs in its `RowData.CachedAttributes` property.

For information about populating values in the pick dialog, see the "Creating Databound Selection Lists and Shuttles" chapter in the *Developing Fusion Web Applications with Oracle Application Development Framework*.

To invoke a custom pick dialog from an ADF Table component:

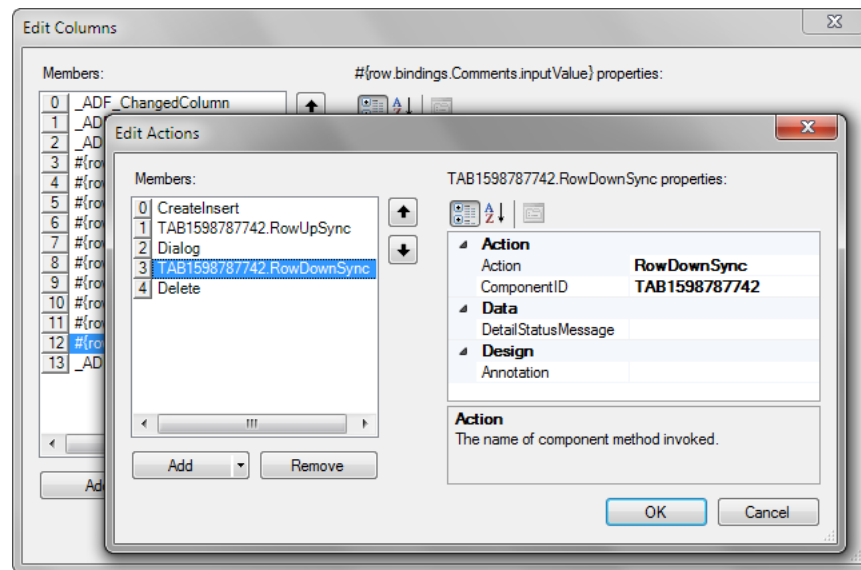
1. Open the integrated Excel workbook.
2. Select the cell in the Excel worksheet that anchors the ADF Table component and click the Edit Properties button in the **Oracle ADF** tab to display the property inspector.
3. Configure the ADF Table component's `RowData.CachedAttributes` property to reference attribute binding values.
4. Click the browse (...) icon beside the input field for **Columns** to display the Edit Columns dialog.
5. In the **Members** list, select the column from which the end user invokes the pick dialog at runtime.
6. Configure the `Actions` attribute of `DoubleClickActionSet` of the column subcomponent (`UpdateComponent` or `InsertComponent`), as described in [Table 8-4](#).

Table 8–4 DoubleClickActionSet Properties

Add this action...	To...
ADFmAction	(InsertComponent only) Invoke the CreateInsert action binding if the end user invokes the DoubleClickActionSet from a newly created row in the Excel worksheet's ADF Table component. In this scenario, the ADF Table component's RowUpSync action (invoked in the next action) fails if the Fusion web application does not contain a placeholder row.
ComponentAction	Invoke the ADF Table component's Table.RowUpSync action to synchronize any pending changes in the current row of the ADF Table component to the Fusion web application.
Dialog	Configure the Dialog action to invoke the pick dialog page from the Fusion web application. Set the Dialog action's ShareFrame property to True. For more information, see Section 8.4, "Displaying Web Pages from a Fusion Web Application."
ComponentAction	Invoke the ADF Table component's Table.RowDownSync action to synchronize data from the row in the ADF Table component's iterator in the Fusion web application that corresponds to the current ADF Table component row in the worksheet.
ADFmAction	(InsertComponent only) If you added a CreateInsert action binding, you should also invoke the Delete action binding to remove the placeholder row.

Figure 8–22 shows the Edit Action dialog of DoubleClickActionSet properties.

Figure 8–22 DoubleClickActionSet Properties



7. Click OK.

8.6 Creating ADF Databound Search Forms in an Integrated Excel Workbook

You can create forms in your integrated Excel workbooks using ADF Input Text and ADF Button components. End users can use the forms you create to insert data or

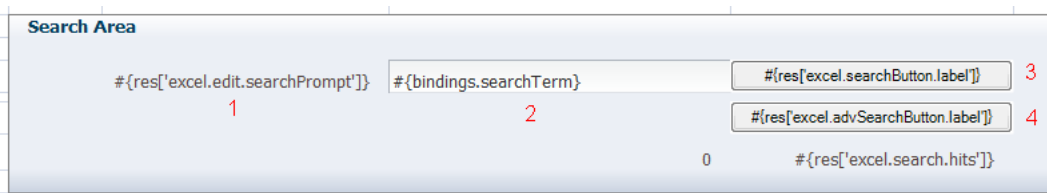
query for information. This section uses the latter example to demonstrate how you create forms.

End users can enter a search term in the ADF Input Text component and retrieve matching results by clicking an ADF Button component. To present a more sophisticated user interface to end users for a search operation, you can invoke search forms from your Fusion web application. Results from these search operations can be downloaded to the ADF Table or ADF Read-only Table components in your integrated Excel workbook.

Figure 8–23 shows an example of design time view of the Oracle ADF components used to configure search options where:

1. ADF Label component is used in a simple search form
2. ADF Input Text component is used in a simple search form
3. ADF Button component is used in a simple search form
4. ADF Button component is used to invoke an advanced search form

Figure 8–23 Oracle ADF Components Used for Search



Note: ADF Desktop Integration does not support usage of the `FindMode` attribute in page definition files. For more information about the `FindMode` attribute, see the "*pageNamePageDef.xml*" section of the *Developing Fusion Web Applications with Oracle Application Development Framework*.

8.6.1 How to Create a Search Form in an Integrated Excel Workbook

You insert an ADF Input Text component and configure it so that the end user can enter a search term. Insert an ADF Button component and configure its action set to:

1. Take the value the end user enters in the ADF Input Text component.
2. Query for the value.
3. Download the results to an ADF Table or ADF Read-only Table component in the integrated Excel workbook.

Before you begin:

It may be helpful to have an understanding of ADF databound search forms. For more information, see [Section 8.6, "Creating ADF Databound Search Forms in an Integrated Excel Workbook."](#)

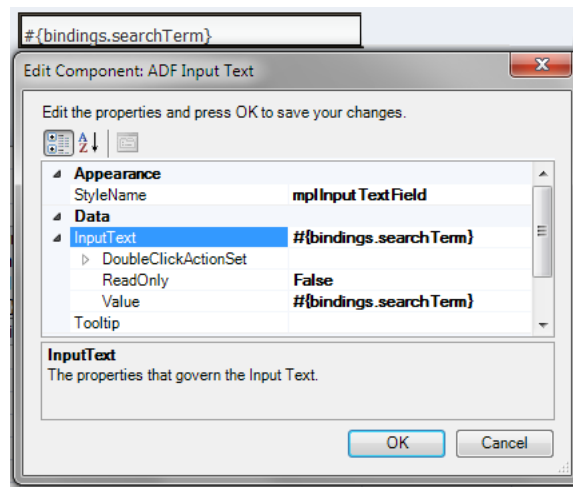
You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 8.1.2, "Additional Functionality for Adding Interactivity to an Integrated Excel Workbook."](#)

To create a simple search form in an integrated Excel workbook:

1. Open the integrated Excel workbook.
2. Insert an ADF Input Text component in the Excel worksheet cell where you want the end user to enter the search criteria.
3. Configure the ADF Input Text component so that it assigns the search term, that a user enters, to an attribute binding.

Figure 8–24 shows an example where an ADF Input Text component assigns the user-entered value to the `searchTerm` attribute binding. The `searchTerm`, which is a part of variable iterator, is then passed as a `NamedData` argument to the `executeSimpleProductQuery` method.

Figure 8–24 ADF Input Text Component for a Simple Search Form



4. Optionally, apply a style to the ADF Input Text component to indicate to end users that they can enter a search term in the cell.
5. Optionally, create an ADF Label component in an adjoining cell to indicate to end users that they can enter a search term in the ADF Input Text component you created in Step 2.
6. Create an ADF Button component in the Excel worksheet.
7. Set the `Label` property of the ADF Button component so that it displays a string at runtime to indicate to end users that they can start a search operation by clicking the button.
8. Optionally, add a tooltip.

For more information about adding tooltips, see [Section 6.9, "Displaying Tooltips in ADF Desktop Integration Form-Type Components."](#)

9. Open the Edit Action dialog to configure the array of actions (`Actions` list) in the `ClickActionSet` properties of the ADF Button component. [Table 8–5](#) describes the actions to invoke in sequence.

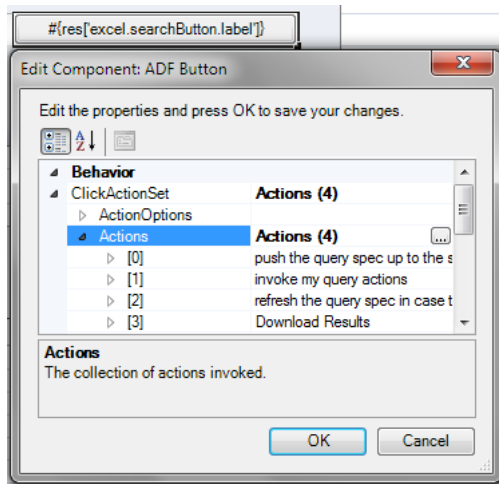
Table 8–5 ClickActionSet Properties of the ADF Button Component

Add this action...	To...
Worksheet	Invoke the UpSync worksheet action to copy the value entered in the cell that hosts an ADF Input Text or ADF List of Values component to the Fusion web application. For more information about worksheet actions, see Section A.16, "Worksheet Actions and Properties."
ADFmAction	Invoke an ADF Model action that is bound to the attribute binding you specified in Step 3. The ADF Model action queries for the end user's search term value referenced by the attribute binding.
Worksheet	Invoke the DownSync worksheet action to synchronize any pending changes from the Fusion web application to the ADF Input Text, ADF Output Text, and ADF List of Values components in the worksheet. For more information about worksheet actions, see Section A.16, "Worksheet Actions and Properties."
Component	Invoke a Download action from the ADF Table or ADF Read-only Table components to download the results that match the search criteria specified.

10. Click OK.

Figure 8–25 shows an example where an ADF Button component invokes the executeSimpleProductQuery action binding using the search term the end user entered in the ADF Input Text component.

Figure 8–25 ADF Button Component for Simple Search Form



8.6.2 How to Create a Search Form using a Web Page in an Integrated Excel Workbook

You can use the ADF Button component, or the ribbon command, to invoke a page from the Fusion web application that displays a search form to the end user. Configure the action set for the ADF Button component to invoke the Download action for the ADF Table or ADF Read-only Table component so that the search results from the search operation are downloaded to the integrated Excel workbook.

For information about creating a search form in a Fusion web application, see the "Creating ADF Databound Search Forms" chapter in the *Developing Fusion Web Applications with Oracle Application Development Framework*.

Before you begin:

It may be helpful to have an understanding of ADF databound search forms. For more information, see [Section 8.6, "Creating ADF Databound Search Forms in an Integrated Excel Workbook."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 8.1.2, "Additional Functionality for Adding Interactivity to an Integrated Excel Workbook."](#)

To invoke a web page from an integrated Excel workbook:

1. Open the integrated Excel workbook.
2. Create an ADF Button component, or a ribbon command, in the Excel worksheet.
3. Set the `Label` property of the component so that it displays a string at runtime to indicate to end users that they can start a search operation by clicking the button.
4. Use the Edit Action dialog to configure the array of actions (`Action list`) in the `ClickActionSet` properties (`SelectActionSet` properties if you are configuring a ribbon command) of the component. [Table 8–6](#) describes the actions to invoke in sequence.

Table 8–6 Actions to Invoke an Advanced Search Form

Add this action...	To...
Dialog	Display the page from your Fusion web application that contains the search form. For more information about displaying pages from a Fusion web application, see Section 8.4, "Displaying Web Pages from a Fusion Web Application."
ComponentAction	Invoke a <code>Download</code> action from the ADF Table or ADF Read-only Table components to download the results that match the search criteria specified.

5. Click OK.

[Figure 8–26](#) shows an example from the `EditableCustomerSearch-DT.xlsx` workbook where the ribbon command's `SelectActionSet` contains a `Dialog` action followed by the ADF Table component's `Download` action. When the end user invokes the ribbon command, the `Dialog` action will show the search page (`searchForm.jspx`) in a browser window. After the end user specifies search criteria in the search page and selects the **Search** button there, the ADF Table component's `Download` action runs. This will retrieve the rows matching the specified search criteria into the integrated worksheet.

Figure 8–26 Ribbon Command Configured to open a Web Page

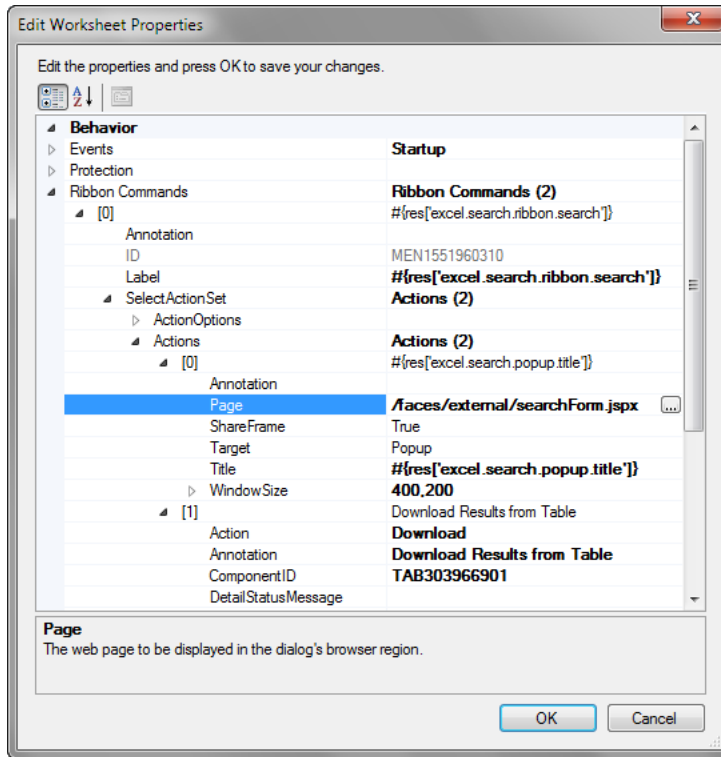
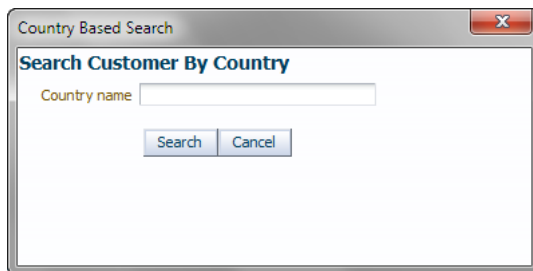


Figure 8–27 shows the web page search form at runtime.

Figure 8–27 Web Page Search Form



8.7 Creating a Form in an Integrated Excel Workbook

You can use the ADF Desktop Integration components described in [Chapter 6, "Working with ADF Desktop Integration Form-Type Components,"](#) to create forms in your integrated Excel workbook. These components can be useful when you want to provide end users with functionality that allows them to view and edit individual fields rather than use the functionality provided by the table-type components to download rows of data from the Fusion web application. Use one or more of the following components to create a form:

- ADF Button

Use this component to provide end users with a button that can invoke a ClickActionSet. [Figure 8–28](#) shows an ADF Button labeled **Search** that invokes a search operation using the search term entered by the end user in the ADF Input Text component.

- **ADF Input Text**
Use this component to provide end users with a read/write field where the current value of a binding appears. This component can also be used to input a value, as in the example illustrated in [Figure 8–28](#), where users enter a search term in the ADF Input Text component.
- **ADF Output Text**
Use this component to provide end users with a read-only field where the current value of a binding appears.
- **ADF List of Values**
Use this component to provide end users with a dropdown menu from which a user can select a value from a list binding.
- **ADF Label**
Use this component to provide end users with instructions or other information on how to use the form you create. For example, the `EditWarehouses-DT.xlsx` workbook uses ADF Label components to display the title of the form. [Figure 8–28](#) shows the label with the title text `Edit Summit Warehouses`.

Figure 8–28 Runtime View of a Form in an Integrated Excel Workbook

The screenshot shows a form titled "Edit Summit Warehouses" with the following fields and values:

Warehouse Id.	201	Manager	Roberta Menchu
Region	South America	Country	Brazil
Address	68 Via Centrale		
City	Sao Paulo	State	
Zip Code		Phone	

Navigation buttons: First, Next, Previous, Last. A "Worksheet Errors" button is also present.

You use the ADF Desktop Integration task pane to insert the components you require into a worksheet.

To create a form in an integrated Excel workbook:

1. Decide which ADF form components you require for the finalized form and insert them in the Excel worksheet.
For more information about these components, see [Chapter 6, "Working with ADF Desktop Integration Form-Type Components."](#)
2. Configure the layout and appearance of the components you insert.
For more information about configuring the appearance of components, see [Chapter 9, "Configuring the Appearance of an Integrated Excel Workbook."](#)
3. Test your form.
For more information about testing an integrated Excel workbook, see [Chapter 13, "Testing Your Integrated Excel Workbook."](#)

8.8 Creating Dependent Lists of Values in an Integrated Excel Workbook

ADF Desktop Integration provides the following components that you use to create lists of values in an integrated Excel workbook:

- ADF List of Values
You configure properties for this component when you want to create a list of values in the Excel worksheet.
- TreeNodeList subcomponent
You configure properties for this component when you want to create a list of values in an ADF Table component column.

Using these two components, you can create a dependent list of values in your integrated Excel workbook. A *dependent list of values* is a list of values component (referred to as a *child list of values*) whose values are determined by another list of values component (referred to as a *parent list of values*).

The server-side list bindings must be defined such that when the selected item of the parent list of values is changed, the available child list of values items are updated properly. Figure 8–29 shows an example with two illustrations from the EditWarehouses-DT.xlsx workbook, where the **Country** field (child list of values) changes when the value in the **Region** field (parent list of values) changes.

Figure 8–29 List of Values and Dependent List of Values

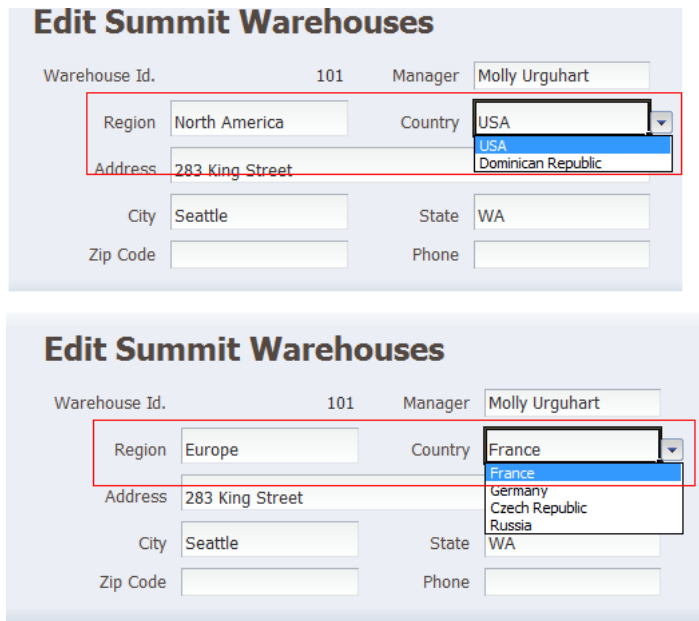


Table 8–7 describes the dependent list of values implementations you can create using the previously listed components and the requirements to achieve each implementation.

Some of the implementations described in Table 8–7 require model-driven lists. For information about creating a model-driven list, see the "How to Create a Model-Driven List" section of the *Developing Fusion Web Applications with Oracle Application Development Framework*.

Table 8–7 *Dependent List of Values Configuration Options*

Configuration	Requirements
Render both the parent and child list of values in the Excel worksheet using ADF List of Values components.	<p>Both instances of the ADF List of Values component must reference a list binding. One or both of the list bindings that you reference can be model-driven lists.</p> <p>Both list bindings can reference model-driven lists only if the underlying iterator has at least one row of data. At runtime, if the underlying iterator has zero rows of data and the end user selects a value from the parent list of values (list binding referenced by the ADF List of Values component's <code>DependsOnListID</code> property), the child list of values (list binding referenced by the ADF List of Values component's <code>ListID</code> property) does not get filtered based on the value the end user selects.</p> <p>To work around this scenario, choose one of the following options:</p> <ul style="list-style-type: none"> ■ Ensure that the underlying iterator has at least one row of data ■ Use an alternative list binding configuration where you expose multiple iterators and all necessary iterators get refreshed <p>For more information, see Section 8.8.1, "How to Create a Dependent List of Values in an Excel Worksheet."</p>
Render both the parent and child list of values in ADF Table component columns using <code>TreeNodeList</code> subcomponents.	<p>Both the parent and child list of values (<code>TreeNodeList</code> subcomponents) must reference tree binding attributes associated with model-driven lists.</p> <p>For more information, see Section 8.8.3, "How to Create a Dependent List of Values in an ADF Table Component's Columns."</p>
Render the parent list of values in an ADF List of Values component and the child list of values in an ADF Table component column using the <code>TreeNodeList</code> subcomponent.	<p>The child list of values (<code>TreeNodeList</code> subcomponent) must reference a tree binding attribute associated with a model-driven list. The parent list of values (ADF List of Values component) must reference a list binding.</p> <p>For more information, see Section 8.8.5, "Creating a Dependent List of Values in an Excel Worksheet and an ADF Table Component Column."</p>

Note the following points if you plan to create a dependent list of values:

- When the cell value referenced by `DependsOnList` or `DependsOnListID` is changed, ADF Desktop Integration overrides any previous changes to the child component list of values without warning the end user.
- The dependent list of values does not work unless the list specified in the `DependsOnList` (or `DependsOnListID`) property is referenced by a component in the Excel worksheet.
- If a circular dependency is defined (List A depends on List B, and List B depends on List A), the first dependency (List A depends on List B) triggers the expected behavior. ADF Desktop Integration considers other dependencies to be misconfigurations.
- You can create a chain of dependencies as follows:
 - List A depends on List B
 - List B depends on List C

In this scenario, a change in List C (grandparent list of values) updates both Lists A (grandchild list of values) and B (child list of values). If you create a similar scenario, you must ensure that both the grandchild list of values and the child list of values, get refreshed whenever the parent list of values selection is changed. You can do this by specifying the two bind variables on the grandchild list of values to set up an implicit dependency between the view attributes. Another way

is to declare explicit attribute dependencies between each of the view attributes that have model-driven lists configured. For example, specify that attribute A depends on attribute B and attribute C, and attribute B depends on attribute C.

- Caching in a dependent list of values is discussed in [Section 15.4, "Caching Lists of Values for Use in Disconnected Mode."](#)
- ADF Desktop Integration caches the values that appear in a dependent list of values. Hence, the dependent list item values for a given parent list selection must remain constant across all rows of an ADF Table component.
- ADF List of Values components using date values are not supported.

8.8.1 How to Create a Dependent List of Values in an Excel Worksheet

Use two instances of the ADF List of Values component to create a dependent list of values in an Excel worksheet.

Specify the list binding referenced by the parent ADF List of Values component as a value for the child ADF List of Values component's `ListOfValues.DependsOnListID` property.

For more information about ADF List of Values, see [Section A.7, "ADF List of Values Component Properties."](#)

Before you begin:

It may be helpful to have an understanding of dependent list of values. For more information, see [Section 8.8, "Creating Dependent Lists of Values in an Integrated Excel Workbook."](#)

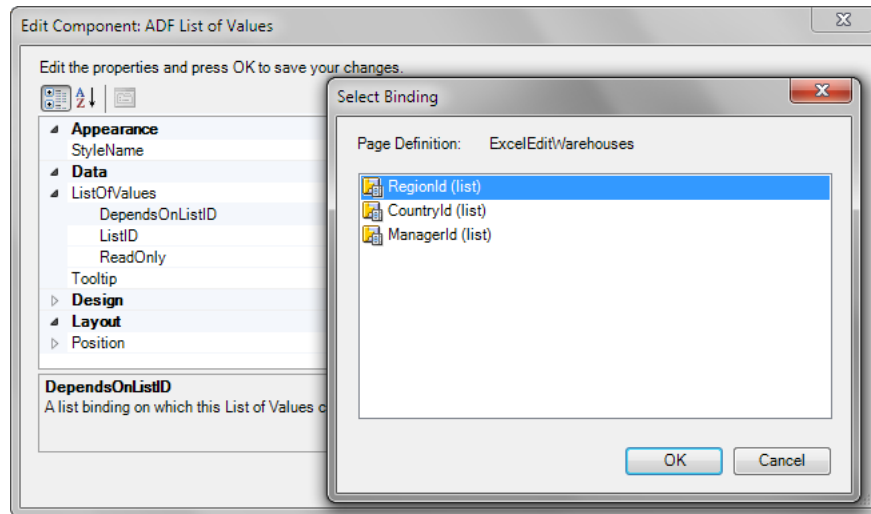
You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 8.1.2, "Additional Functionality for Adding Interactivity to an Integrated Excel Workbook."](#)

To create a dependent list of values in an Excel worksheet:

1. If not present, add the required list bindings to your page definition file.
For more information about adding bindings to page definition files, see [Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook."](#)
2. Open the integrated Excel workbook.
3. Insert two ADF List of Values components into your integrated Excel workbook, as described in [Section 6.6, "Inserting an ADF List of Values Component."](#)
4. In the property inspector for the ADF List of Values component that is to serve as the parent in the dependent list of values, set the value of the `ListOfValues.ListID` property to the list binding that is the parent.
5. In the property inspector for the ADF List of Values component that is to serve as the child in the dependent list of values, set the following properties:
 - `ListOfValues.ListID`
Specify the list binding that is the child in the dependent list of values.
 - `ListOfValues.DependsOnListID`
Select the list binding that you specified for the ADF List of Values component that serves as a parent in Step 4.

Figure 8–30 shows the property inspector for the child ADF List of Values where the RegionId list binding is specified as the parent list of values and CountryId list is the dependent list of values.

Figure 8–30 Design Time Dependent List of Values in an Excel Worksheet



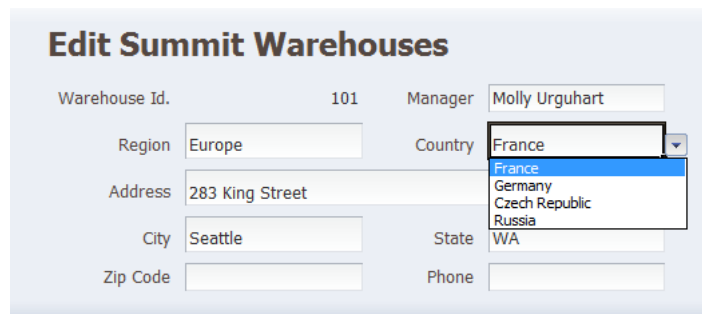
6. Click OK.

8.8.2 What Happens at Runtime: How the Excel Worksheet Renders a Dependent List of Values

At runtime, ADF Desktop Integration renders both instances of the ADF List of Values component. When the end user selects a value from the parent list of values, the selected value determines the list of values in the child list.

Figure 8–31 shows an example where **Country**, a dependent list value, displays only the states from the selected **Region** list value.

Figure 8–31 Runtime Dependent List of Values in an Excel Worksheet



8.8.3 How to Create a Dependent List of Values in an ADF Table Component's Columns

Use instances of the `TreeNodeList` subcomponent to render both lists of values in a dependent list of values in ADF Table component columns at runtime.

Specify a tree binding attribute as a value for the parent `TreeNodeList` subcomponent's `List` property. You also specify a tree binding attribute as a value for the child `TreeNodeList` subcomponent's `List` property and the same tree binding attribute

referenced by the parent `TreeNodeList` subcomponent as a value for its `DependsOnList` property.

Ensure that both tree binding attributes are associated with model-driven lists before you add the tree binding to your page definition file. For information about creating a model-driven list, see the "How to Create a Model-Driven List" section of the *Developing Fusion Web Applications with Oracle Application Development Framework*. For information about adding a tree binding to your page definition file, see [Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook."](#)

For information about the `TreeNodeList` subcomponent, see [Section A.9, "TreeNodeList Subcomponent Properties."](#)

Before you begin:

It may be helpful to have an understanding of dependent list of values. For more information, see [Section 8.8, "Creating Dependent Lists of Values in an Integrated Excel Workbook."](#)

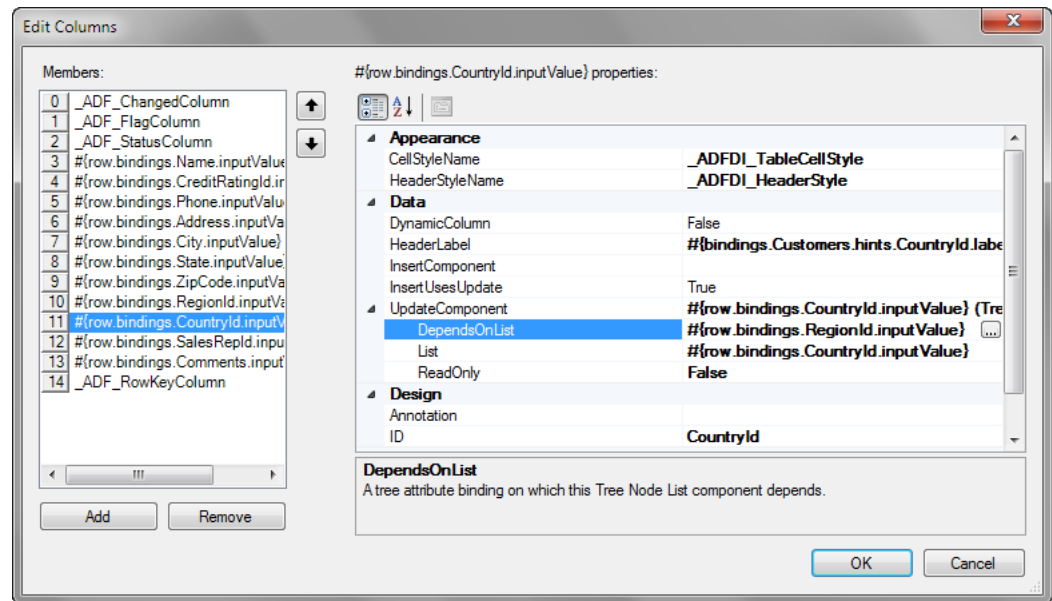
You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 8.1.2, "Additional Functionality for Adding Interactivity to an Integrated Excel Workbook."](#)

To create a dependent list of values in an ADF Table component:

1. Open the integrated Excel workbook.
2. If not present, insert an ADF Table component.
For more information, see [Section 7.3, "Inserting ADF Table Component into Excel Worksheet."](#)
3. In the property inspector for the ADF Table component, invoke the Edit Columns dialog by clicking the browse (...) icon beside the input field for **Columns**.
4. Add a new column (or modify an existing column) to serve as the parent list of values. Specify `TreeNodeList` as the column's subcomponent type. For more information about creating a list of values, see [Section 7.14, "Creating a List of Values in an ADF Table Component Column."](#)
5. Add a new column (or modify an existing column) to serve as the child list of values. Specify `TreeNodeList` as the column's subcomponent type. For more information about creating a list of values, see [Section 7.14, "Creating a List of Values in an ADF Table Component Column."](#)
6. Specify the tree binding attribute of the parent list of values as a value for the `DependsOnList` property in the child list of values column.

[Figure 8–32](#) shows the property inspector for a dependent `TreeNodeList` subcomponent, where the `RegionId` tree binding attribute is specified for the parent list of values and the `CountryId` tree binding attribute is specified for the child list of values.

Figure 8–32 Design Time Dependent List of Values in an ADF Table Component's Column



7. Click OK.

8.8.4 What Happens at Runtime: How the ADF Table Component Column Renders a Dependent List of Values

At runtime, the ADF Table component renders both instances of the `TreeNodeList` subcomponent in the columns that you configured to display these instances. When the end user selects a value from the parent list of values, the selected value determines the list of values in the child list.

Figure 8–33 shows an example where the value that the end user selects in the **Region** column list of values results in the corresponding values for sub-category appearing in the **Country** column list of values.

Figure 8–33 Runtime Dependent List of Values in an ADF Table Component's Columns

Region	Country	Sales R
South America	Brazil	Inry C
South America	Brazil	Li Ng
South America	Mexico	Mai Ng
South America		Colin M
South America		Andre t

Region	Country	Sales
Asia	Japan	Inry
South America	Japan	i N
South America	India	i N
South America	Hong Kong	i N
South America		Colin
South America		Andre

Note: If the child list and the parent list are bound to columns in the same ADF Table component, the child list items are changed for the current row only, when the end user changes the parent list selection.

8.8.5 Creating a Dependent List of Values in an Excel Worksheet and an ADF Table Component Column

Use an instance of the ADF List of Values component and an instance of the `TreeNodeList` subcomponent to create a dependent list of values where you render the parent and the child list of values.

- Parent list of values in the Excel worksheet

An instance of the ADF List of Values component renders the parent list of values in the Excel worksheet.

- Child list of values in an ADF Table component column

An instance of the `TreeNodeList` subcomponent renders the child list of values in the ADF Table component column.

Specify a list binding as a value for the parent ADF List of Values component's `ListID` property. You specify a tree binding attribute as a value for the child `TreeNodeList` subcomponent's `List` property, and the same list binding referenced by the parent ADF List of Values component as a value for its `DependsOnList` property.

Ensure that the tree binding attribute is associated with a model-driven list before you add the tree binding to your page definition file. For information about creating a model-driven list, see the "How to Create a Model-Driven List" section of the *Developing Fusion Web Applications with Oracle Application Development Framework*. For information about adding a list and tree binding to your page definition file, see [Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook."](#)

For more information about the ADF List of Values component, see [Section A.7, "ADF List of Values Component Properties."](#) For information about the `TreeNodeList` subcomponent, see [Section A.9, "TreeNodeList Subcomponent Properties."](#)

Before you begin:

It may be helpful to have an understanding of dependent list of values. For more information, see [Section 8.8, "Creating Dependent Lists of Values in an Integrated Excel Workbook."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 8.1.2, "Additional Functionality for Adding Interactivity to an Integrated Excel Workbook."](#)

To create a dependent list of values in an Excel worksheet and an ADF Table component column:

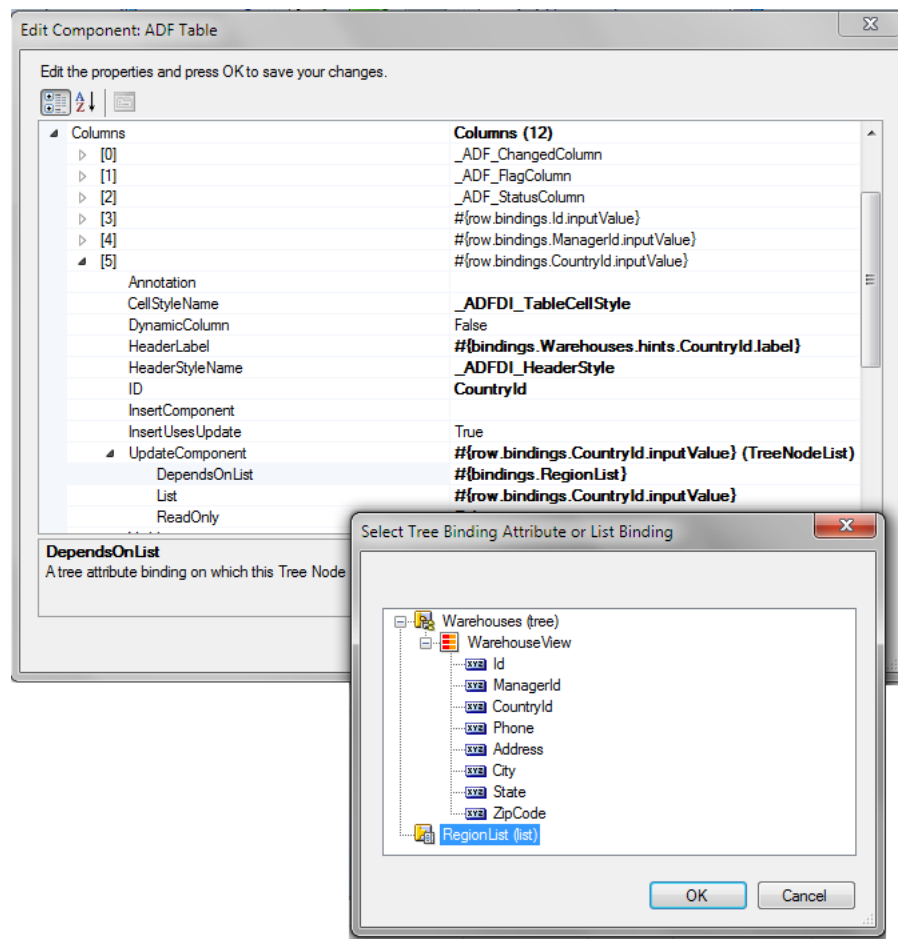
1. Open the integrated Excel workbook.
2. Insert an ADF List of Values component into your integrated Excel workbook, as described in [Section 6.6, "Inserting an ADF List of Values Component."](#)
3. In the property inspector for the ADF List of Values component, set the value of the `ListID` property to the list binding that is to serve as the parent list of values in the dependent list of values.
4. Click **OK**.
5. Open the property inspector for the ADF Table component and invoke the Edit Columns dialog by clicking the browse (...) icon beside the input field for **Columns**.
6. Click **Add** to add a new column to the ADF Table component to serve as the child list of values in the runtime-dependent list of values.

7. Choose the appropriate option for the newly created column:
 - Click the browse (...) icon beside the input field for **InsertComponent** to configure the runtime list of values for insert operations.
 - Click the browse (...) icon beside the input field for **UpdateComponent** to configure the runtime list of values for update and download operations.

In both options, the Select subcomponent to create dialog appears.
8. Select **TreeNodeList** and click **OK**.
9. Expand the property that you selected in Step 7 and configure values as follows:
 - Select the same list binding that you specified as a value for the ADF List of Values component's `ListID` property in Step 3 as a value for the `DependsOnList` property.
 - Select a tree binding attribute associated with a model-driven list for the `List` property.
 - Configure the `ReadOnly` property as desired.
10. Click **OK**.

Figure 8–34 shows the ADF Table property inspector for a child ADF Desktop Integration Tree Node component of `WarehouseLocations-DT.xlsx` where the `RegionList` list binding is specified as the parent list of values.

Figure 8–34 Design Time Dependent List of Values in an Excel Worksheet and an ADF Table Component's Column



8.8.6 What Happens at Runtime: How the Excel Worksheet and the ADF Table Component Column Render a Dependent List of Values

At runtime, the ADF List of Values component renders the parent list of values and the ADF Table component renders the child list of values in the column that you configured to display the `TreeNodeList` subcomponent. When the end user selects a value from the parent list of values, the selected value determines the list of values in the child list.

Figure 8–35 shows an example from `WarehouseLocations-DT.xlsx` where the value that the end user selects in the **Region** list of values determines the list of values that appears in the **Country** column of the ADF Table component.

Figure 8–35 Runtime-Dependent List of Values in an Excel Worksheet and an ADF Table Component's Column

Region	North America			
Status	Warehouse Id.	Manager	Country	Phon
	101	Molly Urguhart	USA	
	501	Marta Havel	USA Dominican Republic	

Region	Asia			
Status	Warehouse Id.	Manager	Country	Phoi
	101	Molly Urguhart	Japan	
	501	Marta Havel	Japan India Hong Kong	

Note: When the parent list is bound to a cell in the worksheet and the child list is bound to an ADF Table Component column, the child list items are updated for all rows in the table when the end user changes the parent list selection.

8.9 Using EL Expression to Generate an Excel Formula

You can use an EL expression to generate an Excel formula as the value of an ADF component. For example, you can use an Excel `HYPERLINK` function in an EL expression. If you use the Excel `HYPERLINK` function in an EL expression, you must enclose the `HYPERLINK` function within an Excel `T` function if you want an Oracle ADF component, such as an ADF Output Text component, to display a hyperlink at runtime.

You enclose the `HYPERLINK` function because ADF Desktop Integration interprets the Excel formula. To work around this, you wrap the `T` function around the `HYPERLINK` function so that the value of the `HYPERLINK` function is evaluated by the `T` function. The resulting value is inserted into the Excel cell that the ADF component references. Use the following syntax when writing an EL expression that invokes the `HYPERLINK` Excel function:

```
=T(=HYPERLINK("link_location","friendly_name"))
```

The EL expression in [Example 8–2](#) uses `HYPERLINK` function to navigate to <http://www.oracle.com/technetwork/developer-tools/adf/overview/index-085534.html> when end user clicks the component.

Example 8–2 `HYPERLINK` Function

```
=T(=HYPERLINK("http://www.oracle.com/technetwork/developer-tools/adf/overview/index-085534.html", "#{res['excel.workbook.powerby']}"))
```

If you write an EL expression using the `HYPERLINK` function, you should select the **Locked** checkbox in the **Protection** tab of the Format Cells dialog for the custom style that you apply to prevent error messages appearing.

Note: When using EL expressions in formulas, ensure that after the EL expression is evaluated, the resulting Excel formula has no more than 255 characters. This applies to formulas used to set conditional values to component properties in the editor.

8.9.1 How to Configure a Cell to Display a Hyperlink Using EL Expression

You write an EL expression that uses the Excel `T` function to evaluate the output of the Excel `HYPERLINK` function. The following task illustrates how you configure an ADF Output Text component to display a hyperlink that opens the Oracle ADF Desktop Integration home page.

Before you begin:

It may be helpful to have an understanding of dynamic hyperlink. For more information, see [Section 8.9, "Using EL Expression to Generate an Excel Formula."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 8.1.2, "Additional Functionality for Adding Interactivity to an Integrated Excel Workbook."](#)

To configure a cell to display a hyperlink using EL expression:

1. Open the integrated Excel workbook.
2. Insert an ADF Output Text component into the Excel worksheet.
3. Write an EL expression for the `Value` property of the ADF Output Text component.

The EL expression that you write invokes the Excel `HYPERLINK` function and uses the Excel `T` function to evaluate the output. In [Example 8–2](#), you entered the following EL expression for the `Value` property:

```
=T( "=HYPERLINK( \"http://www.oracle.com/technetwork/developer-tools/adf/overview/index-085534.html\" , \"#{res['excel.workbook.powerby']}\" )" )
```

Note: Excel requires that you write double double quotes (for example, `"#{res['excel.workbook.powerby']}"`) in the EL expression so that it can evaluate the expression correctly.

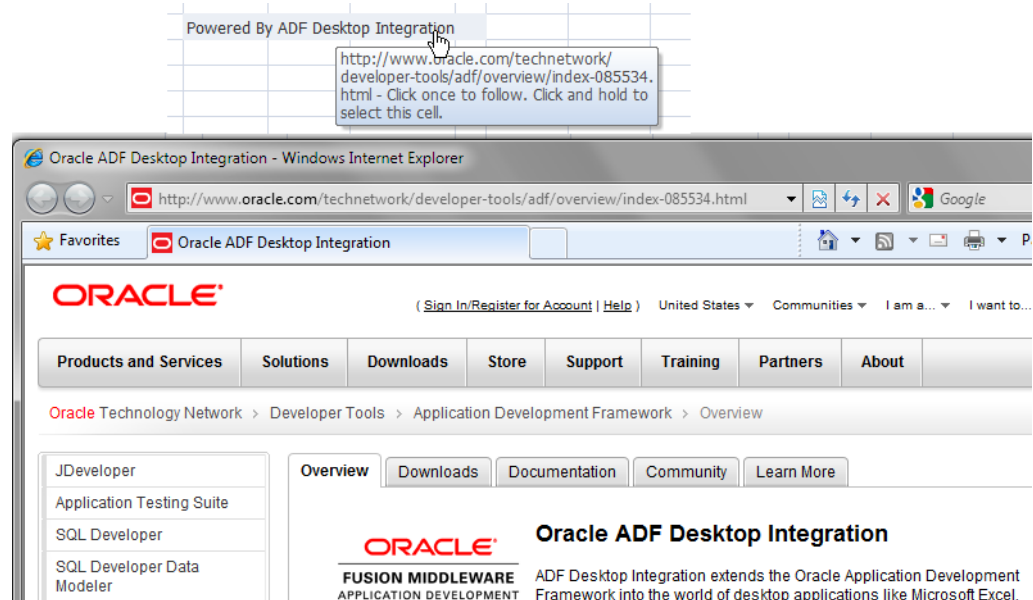
4. Click **OK**.

8.9.2 What Happens at Runtime: How a Cell Displays a Hyperlink using an EL Expression

ADF Desktop Integration evaluates the EL expression that you write at runtime. In the following example, ADF Desktop Integration:

- Retrieves the value of the `excel.workbook.powerby` from the resource file
- Inserts the result into a hyperlinked cell that a user can click

[Figure 8–36](#) shows the runtime view of the example configured in [Section 8.9.1, "How to Configure a Cell to Display a Hyperlink Using EL Expression."](#) When the end user clicks the cell that hosts the ADF Output Text component, the Oracle ADF Desktop Integration home page opens in the web browser.

Figure 8–36 ADF Output Text Component Configured to Display a Hyperlink

8.10 Using Calculated Cells in an Integrated Excel Workbook

You can write Excel formulas that perform calculations on values in an integrated Excel workbook. Before you write an Excel formula that calculates values in an integrated Excel workbook, note the following points:

- Formulas can be entered in cells that reference Oracle ADF bindings and cells that do not reference Oracle ADF bindings
- End users of an integrated Excel workbook can enter formulas at runtime
- You (developer of the integrated Excel workbook) can enter formulas at design time
- During invocation, the ADF Table component actions `Upload` and `RowUpSync` send the results of a formula calculation to the Fusion web application and not the formula itself
- Excel recalculates formulas in cells that reference Oracle ADF bindings when these cells are modified by:
 - Invocation of the ADF Table component `RowDownSync` and `Download` actions
 - Rendering of Oracle ADF components
- The ADF Table and ADF Read-only Table components insert or remove rows as they expand or contract to accommodate data downloaded from the Fusion web application. Formulas are replicated according to Excel's own rules.
- You can enter formulas above or below a cell that references an ADF Table or ADF Read-only Table component. A formula that you enter below one of these components maintains its position relative to the component as the component expands or contracts to accommodate the number of rows displayed.

For more information about Excel functions, see the Function reference section in Excel's online help documentation.

8.10.1 How to Calculate the Sum of a Table-Type Component Column

The following task illustrates how you use the Excel functions `SUM` and `OFFSET` to calculate the total of the column labeled **Difference** at runtime. You use the `OFFSET` function in an Excel formula that you write where you want to reference a range of cells that expands or contracts based on the number of rows that an ADF Table or ADF Read-only Table component downloads. The `SUM` function calculates the total in a range of Excel cells.

Before you begin:

It may be helpful to have an understanding of how to use calculated cells in an integrated Excel workbook. For more information, see [Section 8.10, "Using Calculated Cells in an Integrated Excel Workbook."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 8.1.2, "Additional Functionality for Adding Interactivity to an Integrated Excel Workbook."](#)

To calculate the sum of a column in an ADF Table component:

1. In design mode, select the cell in which you want to write the Excel formula. For example, H16.
2. Write the Excel formula that performs a calculation on a range of cells at runtime. For example:

```
=SUM(OFFSET(G14,1,0):OFFSET(G15,-1,0))
```

where `SUM` calculates the total of values in the range of cells currently referenced by G14 and G15.

[Figure 8-37](#) shows the design time view of the Excel formula in the integrated Excel workbook.

Figure 8-37 Design Time View of Excel Formula in an Integrated Excel Workbook

	G	H	I
hints.ListPrice.label		difference.label	#(res['excel.mar nin Label']
#(row.bindings.Lis		#VALUE!	gs.ListPrice.input
DT Total		0	

3. Save your changes and switch to runtime mode to test that the Excel formula you entered evaluates correctly.

8.10.2 What Happens at Runtime: How Excel Calculates the Sum of a Table-Type Component Column

[Figure 8-38](#) shows the runtime view in the integrated Excel workbook when the Excel formula shown in [Figure 8-37](#) is evaluated. The Excel formula calculates the total of the values in the range of cells that you specified in design mode.

Figure 8–38 Runtime View of Excel Formula in an Integrated Excel Workbook

fx =SUM(OFFSET(H14,1,0):OFFSET(H61,-1,0))		
H	I	J
Site Price	Difference	Current Margin
\$16.99	\$7.99	88.8%
\$16.99	\$9.99	142.7%
DT Total	21120.38	

8.11 Using Macros in an Integrated Excel Workbook

You can define and run macros based on Excel events in an integrated Excel workbook.

Note the following points:

- Macros triggered by an Excel event do not get triggered if the Excel event is invoked by ADF Desktop Integration.
- ADF Desktop Integration code invoked by running of an Excel event when the Excel event is triggered by a macro.

Configuring the Appearance of an Integrated Excel Workbook

This chapter describes how to configure the appearance of an integrated Excel workbook using predefined and custom styles in Excel, how to use EL expressions to dynamically apply styles to Oracle ADF components in a workbook at runtime, how to use labels and brand the Excel workbook, and how to use Worksheet Protection feature.

This chapter includes the following sections:

- [Section 9.1, "About Configuring the Appearance of an Integrated Excel Workbook"](#)
- [Section 9.2, "Working with Styles"](#)
- [Section 9.3, "Applying Styles Dynamically Using EL Expressions"](#)
- [Section 9.4, "Using Labels in an Integrated Excel Workbook"](#)
- [Section 9.5, "Using Styles to Improve the User Experience"](#)
- [Section 9.6, "Branding Your Integrated Excel Workbook"](#)
- [Section 9.7, "Using Worksheet Protection"](#)

9.1 About Configuring the Appearance of an Integrated Excel Workbook

You can configure the appearance of an integrated Excel workbook using both Excel functionality and Oracle ADF functionality. Configuring the appearance of a workbook may make the workbook more usable for end users. For example, applying a particular style to cells that render ADF Output Text components at runtime may indicate to end users that the cell is read-only. You may also want to configure the appearance of an integrated Excel workbook so that it aligns with your company's style sheet or the color scheme of the Fusion web application that the Excel workbook integrates with.

Using styles to configure your data in your integrated Excel workbook gives you many benefits. For example, you can use a particular style for ADF Output Text components, and a different style for ADF Input Text components.

ADF Desktop Integration provides several predefined Excel styles to apply to the ADF Desktop Integration components you configure in a workbook. You may want to define additional styles to meet the needs of your desktop integration project. If you do, familiarize yourself with the formats in an Excel workbook that render differently depending on the locale, region, and language.

9.1.1 Integrated Excel Workbook Configuration Use Cases and Examples

You can customize the appearance of ADF Desktop Integration components using styles. For example, [Figure 9–1](#) shows various styles applied to the columns of ADF Table in `EditCustomers-DT.xlsx`. Note that some styles shown in [Figure 9–1](#) are predefined styles (for example, `_ADFDT_TableCellStyle`) and others are created by Excel (for example, `Explanatory Text`).

Figure 9–1 Styles Applied to Columns of ADF Table in `EditCustomers-DT.xlsx`

Changed	Flagged	Status	Name	Credit Rating	Phone	Address	City	State	Zip Code	Region	Country	Sales Rep.	Comments	Key
			Unisports	Excellent	55-2066101	72 Via Bahia	Sao Paulo			South America	Brazil	Henry Giljum	<i>usually orders</i>	
			Simms Athletics	Poor	81-20101	Takashi	Osaka			Asia	Japan	Mai Nguyen	<i>always pay by</i>	
			Delhi Sports	Good	91-10351	Chanakya	New Delhi			Asia	India	Mai Nguyen	<i>specializes in</i>	
			Womansport	Fair	1-206-104-0103	281 King Street	Seattle	WA	98101	North America	USA	Colin Magee		

9.1.2 Additional Functionality for Configuring the Appearance of an Integrated Excel Workbook

After you have applied styles to configure the appearance of your integrated Excel workbook, you may find that you need to add additional functionality to configure your workbook. Following are links to other functionalities that you can use:

- Branding:** In addition to styles, ADF Desktop Integration provides a collection of properties (`BrandingItems`) that enable you to brand your integrated Excel workbook with application name, application version details, and copyright information. For more information, see [Section 9.6, "Branding Your Integrated Excel Workbook."](#)
- Localization:** You can customize the integrated Excel workbook as part of the process to internationalize and localize with the Fusion web application. For more information, see [Chapter 10, "Internationalizing Your Integrated Excel Workbook."](#)

9.2 Working with Styles

ADF Desktop Integration provides a mechanism to apply Excel-defined styles to some Oracle ADF components at runtime. The Oracle ADF components that support the application of styles have properties with `StyleName` in their name. For example, the column properties of the ADF Table and ADF Read-only Table components both support the properties `HeaderStyleName` and `CellStyleName` that determine styles to apply at runtime.

9.2.1 Predefined Styles in ADF Desktop Integration

Many properties have default values that are drawn from a predefined list of ADF Desktop Integration module styles. For example, the `HeaderStyleName` property's default value is `_ADFDT_HeaderStyle`, one of the predefined styles in ADF Desktop Integration. ADF Desktop Integration automatically adds these predefined styles to the Excel workbook once when it is enabled for use with ADF Desktop Integration.

The following is the list of predefined styles:

- `_ADFDT_FormBottomStyle`

- `_ADFDDI_FormDoubleClickCellStyle`
- `_ADFDDI_FormTopStyle`
- `_ADFDDI_HeaderStyle`
- `_ADFDDI_InputTextStyle`
- `_ADFDDI_LabelStyle`
- `_ADFDDI_OutputTextStyle`
- `_ADFDDI_ReadOnlyTableStyle`
- `_ADFDDI_TableCellROStyle`
- `_ADFDDI_TableCellStyle`
- `_ADFDDI_TableChangedColumnStyle`
- `_ADFDDI_TableDoubleClickCellStyle`
- `_ADFDDI_TableFlagColumnStyle`
- `_ADFDDI_TableKeyCellStyle`
- `_ADFDDI_TriangleHeaderStyle`

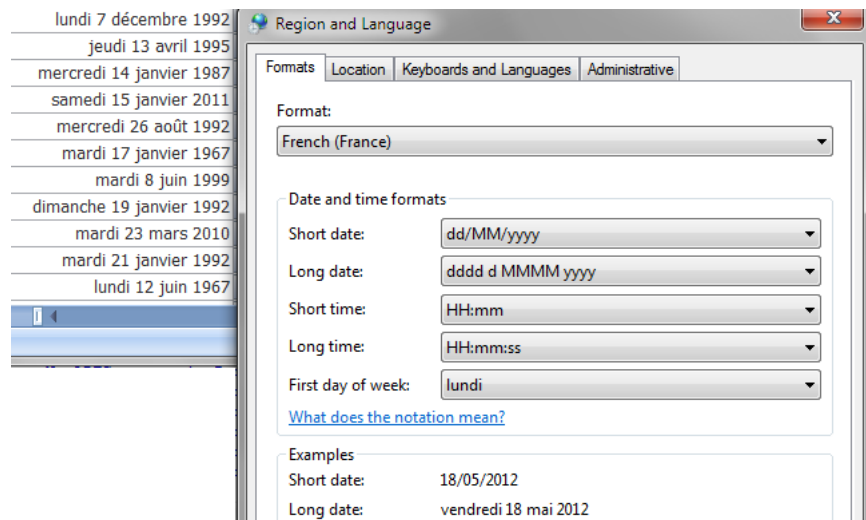
You can merge these styles and other styles that you define yourself from an integrated Excel workbook into another Excel workbook that you intend to integrate with a Fusion web application. You may create additional styles for use in your Excel workbook. For example, to add a date-specific formatting, you can duplicate `_ADFDDI_TableCellStyle`, call it `MyTableCellDateStyle`, and add your date-specific formatting.

Once you have decided what styles to apply to the ADF Desktop Integration components at runtime, you can write EL expressions to associate a style with a component. The ADF Desktop Integration component properties that include *StyleName* in their name take an EL expression as a value. The ADF Label component and the `Label` property of other ADF components also support EL expressions. These EL expressions can retrieve the values of string keys defined in resource bundles or the values of attribute control hints defined in your Fusion web application.

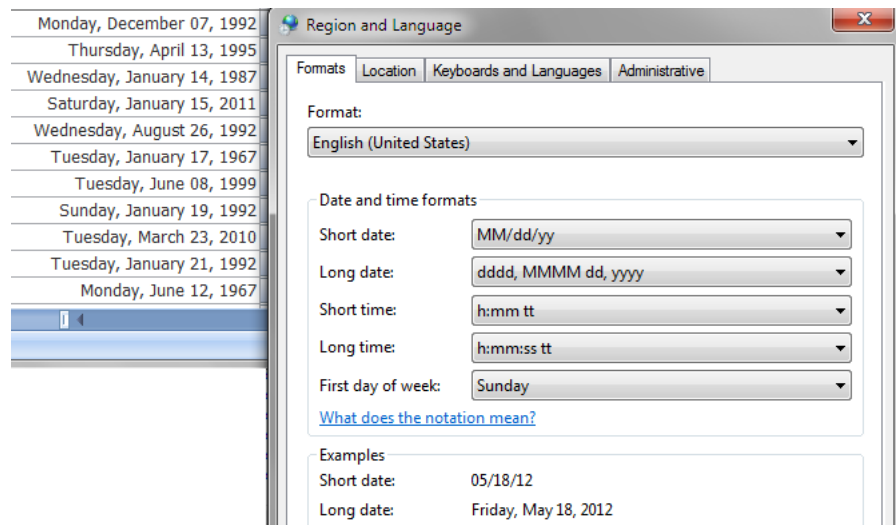
For more information about creating new styles and merging styles into a workbook, see Excel's documentation.

9.2.2 Excel's Date Formats and Microsoft Windows' Regional and Language Options

Some formats in the **Date** category of the **Number** styles that Excel can apply to cells change if a user changes the locale of the local system using the Regional and Language Options dialog that is accessible from the Microsoft Windows Control Panel. The * character precedes these formats in the **Type** list. [Figure 9-2](#) shows an example of a Date type that formats dates in a cell using French (France) conventions.

Figure 9–2 French Date Formats in Excel

If the end user changes the regional options of a system to use **English (United States)**, as illustrated in Figure 9–3, the cells that are formatted with the style in Figure 9–2 use the **English (United States)** conventions.

Figure 9–3 US English Date Formats in Excel

Note: In order for Excel to properly format and manipulate date values with no time component, the form or table attributes must use the `java.sql.Date` data type in the application's model definition.

9.2.3 How to Apply a Style to an Oracle ADF Component

To apply a style to an Oracle ADF component, use the property inspector to set values for properties with *StyleName* in their name.

Before you begin:

It may be helpful to have an understanding of styles. For more information, see [Section 9.2, "Working with Styles."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 9.1.2, "Additional Functionality for Configuring the Appearance of an Integrated Excel Workbook."](#)

To apply a style:

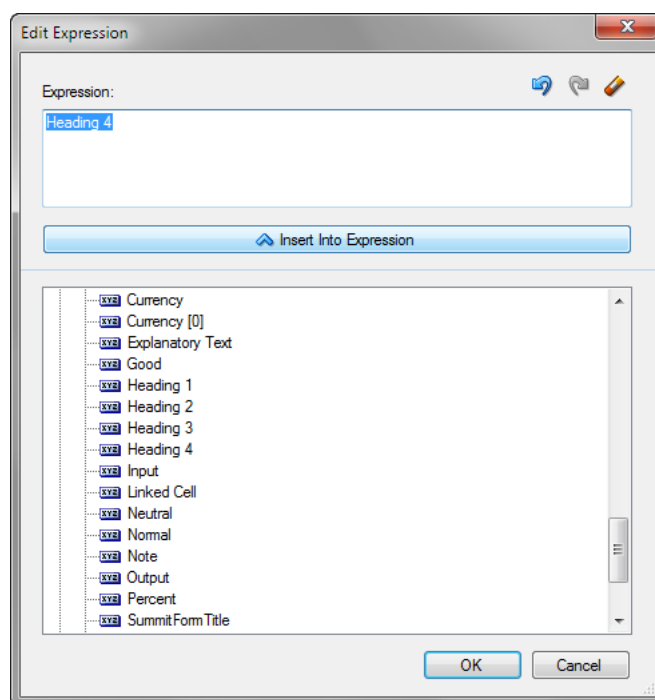
1. In the integrated Excel workbook, select the cell that references the Oracle ADF component you want to modify and then click the **Edit Properties** button in the **Oracle ADF** tab.
2. Select the **StyleName** property and click the browse (...) icon to display the Edit Expression dialog.
3. Expand the **Styles** node and select the style to apply to cell at runtime.

For example, apply a heading style (Heading 4 style) to the Binding Warehouse ID output text field. Applying the Heading 4 style rather than a general style to the field results in data appearing as bold and in blue color.

4. Click **Insert Into Expression** to insert the selected style into the Expression field.

[Figure 9-4](#) shows the Edit Expression dialog.

Figure 9-4 Edit Expression Dialog Applying a Style



5. Click **OK**.

9.2.4 What Happens at Runtime: How Style Is Applied to an ADF Component

The EL expression that you entered as a value for the property with *StyleName* in its name is evaluated at runtime. If it corresponds to one of the predefined styles or one that you defined, the style is applied to the ADF component that you set the property for.

If a cell that references an ADF component has a style applied to it that differs from the style defined in the properties of the ADF component, the ADF component overwrites the existing style at runtime and applies the style defined by its properties.

For example, [Figure 9–5](#) shows the runtime values of Binding Warehouse ID after the Heading 4 style is applied, overriding the default `_ADFDFI_OutputTextStyle` style.

Figure 9–5 Runtime Values After Applying Another Style

9.3 Applying Styles Dynamically Using EL Expressions

Oracle ADF component properties that include `StyleName` in their name can take an EL expression as a value. The EL expressions that you write can resolve to a named Excel style at runtime that is applied to the ADF component. The EL expressions that you write are Excel formulas that may include ADF data binding expressions. ADF Desktop Integration does not evaluate or apply results when a user navigates between cells or during upload.

The following examples show different contexts where you can use EL expressions to determine the behavior and appearance of ADF components at runtime. [Example 9–1](#) applies a style dynamically during download. If the status value for binding is `Closed`, apply a read-only style (`MyReadOnlyStyle`). Otherwise apply another style (`MyReadWriteStyle`).

Example 9–1 Applying a Style Dynamically During Download

```
=IF("#{bindings.Status}" = "Closed", "MyReadOnlyStyle", "MyReadWriteStyle")
```

[Example 9–2](#) uses a mixture of Excel formulas and ADF binding expressions to handle errors and type conversion.

Example 9–2 EL Expressions to Handle Errors and Type Conversion

```
=IF(ISERROR(VALUE("#{bindings.DealSize}")), "BlackStyle",  
IF(VALUE("#{bindings.DealSize}") > 300, "RedStyle", "BlackStyle"))
```

9.3.1 What Happens at Runtime: How an EL Expression Is Evaluated

When evaluating EL expressions at runtime, ADF Desktop Integration determines the value that the EL expression references. It then replaces the EL expression in the Excel formula with the value. In [Example 9–1](#), ADF Desktop Integration first determines that value of the EL expression, `#{bindings.Status}`, in the following Excel formula:

```
=IF("#{bindings.Status}" = "Closed", "MyReadOnlyStyle", "MyReadWriteStyle")
```

It then replaces the EL expression with the runtime value, as in the following example, where the expression evaluated to `Closed`:

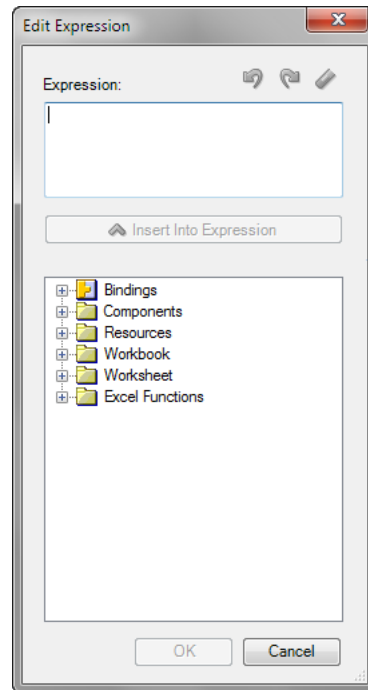
```
=IF("Closed" = "Closed", "MyReadOnlyStyle", "MyReadWriteStyle")
```

Excel evaluates the formula and, in this example, applies the `MyReadOnlyStyle` style.

9.3.2 How to Write an EL Expression That Applies a Style at Runtime

You write EL expressions for the Oracle ADF component properties that support EL expressions in the Edit Expression dialog that is accessible from the Oracle ADF component's property inspector. [Figure 9-6](#) displays an Edit Expression dialog launched from the property inspector window of an ADF Button component.

Figure 9-6 Edit Expression Dialog



Before you begin:

It may be helpful to have an understanding of how to apply styles dynamically. For more information, see [Section 9.3, "Applying Styles Dynamically Using EL Expressions."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 9.1.2, "Additional Functionality for Configuring the Appearance of an Integrated Excel Workbook."](#)

To write an EL expression that applies a style at runtime:

1. Open the integrated Excel workbook.
2. Select a cell in the Excel worksheet that references the Oracle ADF component for which you want to write an EL expression.
3. Click the **Edit Properties** button in the **Oracle ADF** tab to display the property inspector.
4. Select the property in the property inspector with which you want to associate an EL expression and click the browse (...) icon to display the Edit Expression dialog.

Note: The Edit Expression dialog appears only if the Oracle ADF component that you selected in Step 2 supports EL expressions. Depending on the context, the browse (...) icon can launch other editors such as the Edit Action dialog.

The Edit Expression dialog, as illustrated in [Figure 9–6](#), displays a hierarchical list of the Oracle ADF components, bindings, styles, resources, and Excel functions that you can reference in EL expressions. For more information about the syntax of EL expressions that you enter in this dialog, see [Appendix B, "ADF Desktop Integration EL Expressions."](#)

9.3.3 What You May Need to Know About EL Expressions That Apply Styles

Note the following points about EL expressions that apply styles:

- EL expressions that evaluate to styles are applied when:
 - An ADF Table component invokes its `Download` or `DownloadForInsert` actions
 - Rows are inserted into an ADF Table component
 - A worksheet invokes its `DownSync` action

- EL expressions that evaluate to styles are not applied when:
 - An ADF Table component invokes its `RowDownSync` action
 - The end user edits the format properties of a cell

Note also that an EL expression that evaluates to a style is not reevaluated when the end user edits a cell's value.

- The runtime value of an EL expression does not match a style defined in the end user's integrated Excel workbook

In this scenario the style formats of the targeted cells do not change. Instead, they retain their existing style formats. If you configured client-side logging, ADF Desktop Integration generates an entry in the log file when an EL expression evaluates to a style that is not defined in the end user's integrated Excel workbook. For more information about client-side logging, see [Section C.3, "Generating Log Files for an Integrated Excel Workbook."](#)

- In Excel, given a workbook with various custom named styles, if you save a copy of that workbook from Excel, Excel automatically (and silently) deletes any custom named style that is not applied to any cell.

If you have styles that are only used in EL expressions and not applied to any cell, Excel may delete them.

- The ADF Desktop Integration Publish feature creates a copy of the workbook. Hence, unused styles can disappear. The workaround is to apply each style once to an unused cell on an used worksheet.

9.4 Using Labels in an Integrated Excel Workbook

Use labels to provide end users with information about how they use the functionality in an integrated Excel workbook. You can write EL expressions that retrieve the value of string keys defined in a resource bundle or that retrieve the values of attribute control hints. An integrated Excel workbook evaluates the value of a `Label` property only when the workbook is initialized.

9.4.1 Retrieving the Values of String Keys from a Resource Bundle

Figure 9–7 shows a portion of the design time view of the `EditWarehouses-DT.xlsx` workbook in the Summit sample application for ADF Desktop Integration. It shows examples of ADF Label components and ADF Input Text components that have EL expressions specified for their `Label` properties.

Figure 9–7 Design Time View of an ADF Label Component and an ADF Input Text Component with Label Property



At runtime, these EL expressions resolve to string keys defined in the `res` resource bundle that is registered with the Summit sample application for ADF Desktop Integration. You define resource bundles in the workbook properties dialog. For information about referencing string keys from a resource bundle, see [Section 10.2, "Using Resource Bundles in an Integrated Excel Workbook."](#)

Figure 9–8 shows the corresponding runtime view of the ADF Label component and ADF Input Text component illustrated in design mode in [Figure 9–7](#).

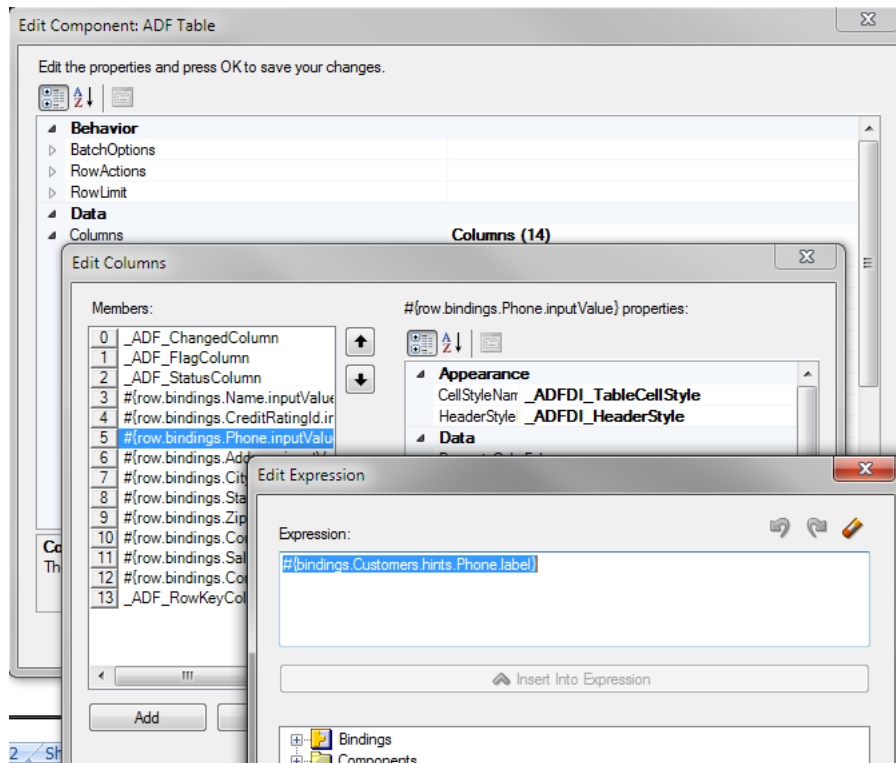
Figure 9–8 Runtime View of an ADF Label Component and an ADF Button Component with Label Property



9.4.2 Retrieving the Values of Attribute Control Hints

In addition to string keys from resource bundles, the ADF Label component and the `Label` property of other ADF components can reference attribute control hints that you define for entity objects and view objects in your JDeveloper project. [Figure 9–9](#) shows the expression builder for the Phone column in the `EditCustomers-DT.xlsx` workbook's ADF Table component. The expression builder contains an EL expression for the `HeaderLabel` property of the Phone column that retrieves the value (`Phone`) defined for an attribute control hint at runtime.

Figure 9–9 EL Expression That Retrieves the Value of an Attribute Control Hint for a Label Property



Attribute control hints can be configured for both view objects and entity objects. Information about how to add an attribute control hint to an entity object can be found in the "Defining Attribute Control Hints for Entity Objects" section of the *Developing Fusion Web Applications with Oracle Application Development Framework*. Information about how to define a UI hint for a view object can be found in the "Defining UI Hints for View Objects" section of the *Developing Fusion Web Applications with Oracle Application Development Framework*.

9.4.3 How an Integrated Excel Workbook Evaluates a Label Property

An integrated Excel workbook evaluates the `Label` properties of ADF components when the workbook is initialized after you or the end user opens the workbook for the first time. The integrated Excel workbook saves the retrieved values for the `Label` properties when the workbook itself is saved to a directory on the system.

The retrieved values for the `Label` properties do not get refreshed during invocation of actions such as the worksheet's `DownSync` action or the ADF Table component's `Download` action. You indirectly refresh the retrieved values of the `Label` properties if you invoke the workbook actions `ClearAllData` or `EditOptions` described in [Table A–20](#).

9.5 Using Styles to Improve the User Experience

It is good practice to provide end users of integrated Excel workbooks with information that helps them understand how to use the ADF components that you provide to integrate with a Fusion web application. You can do this by:

- Providing end users with instructions on how to use Oracle ADF components such as ADF Button and ADF Input Text components.

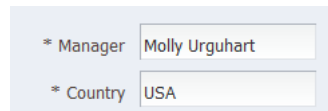
The ADF Label component and the `Label` property of other ADF components is useful for this task, as you can write labels that instruct the end user on how to use the component.

- Apply styles that indicate if an ADF component is read-only or read-write.

9.5.1 Using ADF Label Components to improve the User Experience

You can use ADF Label components to provide end users of an integrated Excel workbook information about how to use other ADF components in the workbook. For example, many forms, by convention, use an `*` to indicate to end users that they must enter a value in an input field. [Figure 9–10](#) shows two ADF Input Text components with ADF Label components in adjoining cells. Each ADF Label component references an EL expression that retrieves the value of a string key from a resource bundle at runtime. Each string key includes the `*` character to indicate to end users that they must supply a value.

Figure 9–10 ADF Label Components Providing End-User Instruction



For information about using resource bundles, see [Section 10.2, "Using Resource Bundles in an Integrated Excel Workbook."](#)

9.5.2 What You May Need to Know About the Read-Only Property in an Integrated Excel Workbook

Note the following points about the read-only property in an integrated Excel workbook:

- ADF Output Text, ADF Label, and ADF Table header row do not have read-only properties. However, these components have implied read-only behavior. In addition, end users can enter values in the cells that host these components and temporarily change the values that appear in these cells. ADF Desktop Integration ignores these changes when uploading from the worksheet and overwrites them when it downloads data from the Fusion web application.
- The ADF Input Text component, ADF List of Values component, and `TreeNodeList` subcomponent each have a read-only property (`ReadOnly`).

Note: If you specify an Excel formula in the `Value` property of an ADF Input Text component, the component behaves as if its `ReadOnly` property were `True`. The component ignores the actual value of the `ReadOnly` property.

- To protect the values of read-only cells at runtime, set the worksheet protection to automatic. When an attempt is made to edit a read-only cell after enabling worksheet protection, Excel displays a warning message and the edit is blocked. For more information about worksheet protection, see [Section 9.7, "Using Worksheet Protection."](#)
- Do not use the Excel's Protect Sheet or Protect Workbook features directly in an integrated Excel workbook. Also, ensure that end users do not use these features.

To prevent end-user confusion, apply styles to components, such as the ADF Output Text component, that indicate to end users whether a component is read-only or can be edited. By default, the ADF Output Text component uses the predefined style, `_ADFDI_OutputTextStyle`. You can define your own styles and apply them to components as described in this chapter.

For more information about the properties that ADF Desktop Integration components support, see [Appendix A, "ADF Desktop Integration Component Properties and Actions."](#)

9.6 Branding Your Integrated Excel Workbook

ADF Desktop Integration provides several features that you can configure to brand your integrated Excel workbook with information such as application name, version information, and copyright information. You can use the workbook `BrandingItems` group of properties to associate this information with an integrated Excel workbook. You must configure a Ribbon tab as described in [Section 8.3, "Configuring the Runtime Ribbon Tab"](#) so that the end user can view this branding information by clicking a ribbon button that invokes the `ViewAboutDialog` workbook action at runtime. For more information about workbook actions, see [Table A-20](#).

You can also define string keys in a resource bundle to define information, such as titles, in one location that can then be used in multiple locations in an integrated Excel workbook at runtime when EL expressions retrieve the values of these string keys. For information about defining string keys, see [Section 10.2, "Using Resource Bundles in an Integrated Excel Workbook."](#)

9.6.1 How to Brand an Integrated Excel Workbook

You define values for the workbook `BrandingItems` group of properties.

Before you begin:

It may be helpful to have an understanding of how to customize brand of your integrated Excel workbook. For more information, see [Section 9.6, "Branding Your Integrated Excel Workbook."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 9.1.2, "Additional Functionality for Configuring the Appearance of an Integrated Excel Workbook."](#)

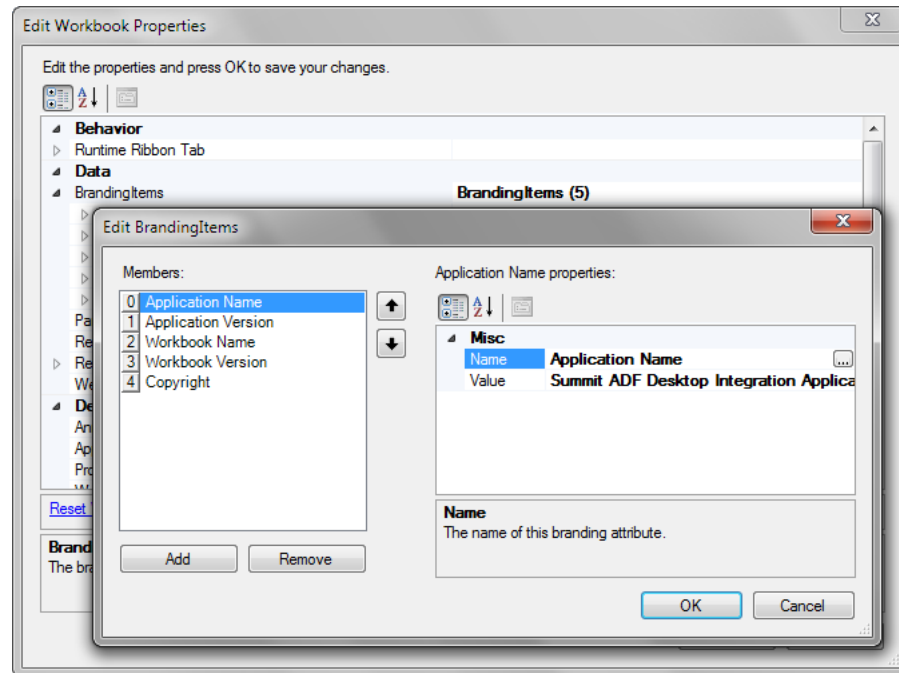
To brand an integrated Excel workbook:

1. Open the integrated Excel workbook.
2. In the Workbook group of the Oracle ADF tab, click Workbook Properties.
3. In the Edit Workbook Properties dialog, click the browse (...) icon beside the input field for **BrandingItems**.
4. In the Edit BrandingItems dialog, click **Add** and specify values for the new element as follows:
 - **Name**
Specify the name, or the EL expression, of the branding item to define.
 - **Value**
Specify a literal string or click the browse (...) icon to invoke the expression builder and write an EL expression that retrieves a value at runtime.

BrandingItems must use literal strings or resource expressions, and must not contain any binding expression.

Figure 9–11 shows the design time view of branding items in the Summit sample application for ADF Desktop Integration.

Figure 9–11 Design Time View of Branding Items in the Summit Sample Application for ADF Desktop Integration

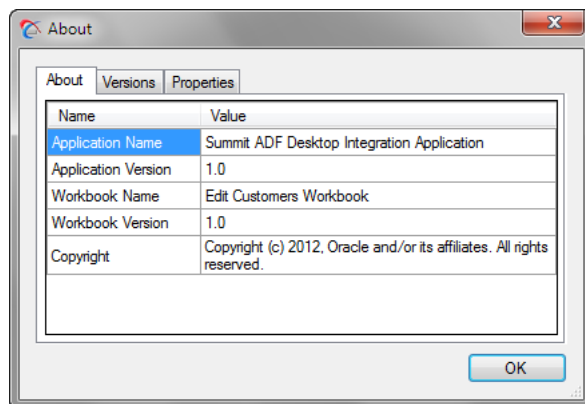


5. Click OK.

9.6.2 What Happens at Runtime: the BrandingItems Group of Properties

At runtime, the name-value pairs that you define for the BrandingItems group of properties appear in a dialog that the end user invokes from the **About** button of the **Oracle ADF** tab, which you configured to appear, as described in [Section 8.3, "Configuring the Runtime Ribbon Tab."](#) Figure 9–12 shows the runtime view of branding items in the `EditCustomers.xlsx` workbook.

Figure 9–12 Runtime View of Branding Items in the Summit Sample Application for ADF Desktop Integration



9.7 Using Worksheet Protection

By default, the end user can edit the values of locked cells and ADF Desktop Integration read-only components, such as ADF Label and ADF Output Text, at runtime. While uploading data, ADF Desktop Integration ignores these changes and overwrites them when it downloads data from the Fusion web application.

To prevent editing of locked cells at runtime, you must enable worksheet protection. Optionally, you can also provide a password to prevent the end user from turning off worksheet protection.

9.7.1 How to Enable Worksheet Protection

Worksheet protection enables true read-only mode for locked and read-only cells, and prevents any editing at runtime.

Before you begin:

It may be helpful to have an understanding worksheet protection. For more information, see [Section 9.7, "Using Worksheet Protection."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 9.1.2, "Additional Functionality for Configuring the Appearance of an Integrated Excel Workbook."](#)

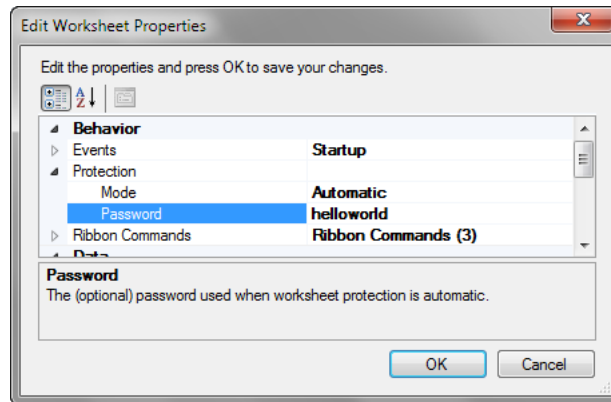
To enable Worksheet Protection:

1. Open the integrated Excel workbook.
2. In the Workbook group of the Oracle ADF tab, click **Workbook Properties**.
3. In the Edit Worksheet Properties dialog, expand the **Protection** property and configure values as follows:
 - To enable worksheet protection at runtime, set the **Mode** to **Automatic**.
 - If desired, provide a value in the **Password** field. The end user cannot turn off sheet protection at runtime without knowing this value.

Note that the password is not encrypted and that the maximum password length allowed by Excel is 255 characters. If you specify a longer password, it will be truncated silently at runtime when sheet protection is toggled.

Figure 9–13 shows the design time view of worksheet protection in the Summit sample application for ADF Desktop Integration.

Figure 9–13 Design Time View of Worksheet Protection in the Summit Sample Application for ADF Desktop Integration

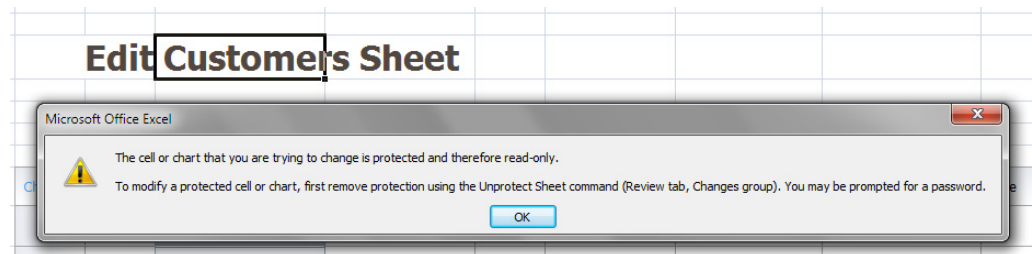


4. Click OK.

9.7.2 What Happens at Runtime: How the Locked Property Works

At runtime, if the end user tries to edit a read-only cell or a ADF Desktop Integration read-only component, Excel displays the warning message, as shown in Figure 9–14.

Figure 9–14 Worksheet Protection Warning at Runtime



When worksheet protection is enabled, ADF Desktop Integration controls the **Locked** property for cells that are within the bounds of ADF Desktop Integration components. The **Locked** property of cells outside the bounds of ADF Desktop Integration components is not affected.

At runtime, ADF Desktop Integration evaluates the read-only behavior of its components. Some components such as ADF Label and ADF Output Text, are always read-only, and other components, such as ADF Input Text, have a read-only property. At runtime, the **Locked** property is set to true when read-only for the component evaluates to true. The header labels of ADF Table components are always read-only, but column subcomponents might be ADF Output Text or ADF Input Text. At runtime, each components read-only behavior is evaluated and the corresponding cell's **Locked** property is set to the appropriate value.

9.7.3 What You May Need to Know About Worksheet Protection

Worksheet protection is not enabled by default, you need to enable it at design time if you want to use it for a particular worksheet. Also, after worksheet protection is enabled, the **Locked** property for cells is set at runtime, not at design time.

It is important to note that the password used for worksheet protection is itself not encrypted or stored in a safe location. Worksheet protection is used to improve worksheet usability, not to protect sensitive data.

After worksheet protection is enabled, Excel behaves differently. Here are some differences that you can expect:

- The ADF Table components cannot be sorted, as they include read-only cells in the Key column.
- The end user can insert a full row or column. However, once inserted, they cannot be deleted.
- The end user cannot insert partial rows or columns.

Internationalizing Your Integrated Excel Workbook

This chapter describes internationalization issues to consider when developing an integrated Excel workbook, how to use resource bundles, and how to localize the integrated Excel workbook.

This chapter includes the following sections:

- [Section 10.1, "About Internationalizing Your Integrated Excel Workbook"](#)
- [Section 10.2, "Using Resource Bundles in an Integrated Excel Workbook"](#)
- [Section 10.3, "Localization in ADF Desktop Integration"](#)

10.1 About Internationalizing Your Integrated Excel Workbook

ADF Desktop Integration provides several features that allow you to deliver integrated Excel workbooks as part of an internationalized Fusion web application. One of the principal features is the use of resource bundles to manage the localization of user-visible strings that appear in Oracle ADF components at design time. It also uses resource bundles to manage the user-visible strings that appear in these components at runtime. This chapter also describes the use of resource bundles.

Note the following points about internationalization and localization in an integrated Excel workbook:

- **Internationalized Data**

ADF Desktop Integration supports both single- and double-byte character sets. It marshals data transmitted between an Excel worksheet and a Fusion web application into XML payloads. These XML payloads use UTF-8 encoding with dates, times, and numbers in canonical formats.
- **Locale**

The locale of the system where the Excel workbook is used determines the format for dates, times, and numbers. These settings (formats and the locale of the system) may differ from the settings used by the Fusion web application. ADF Desktop Integration does not attempt to synchronize these settings, but it ensures that the data retains its integrity. ADF Desktop Integration does not provide a mechanism for end users to change the language or display settings of the Oracle ADF components in an integrated Excel workbook at runtime.

When configuring or applying styles to ADF components in an integrated Excel workbook, configure or choose styles that are locale-sensitive. For more information, see [Section 9.2, "Working with Styles."](#)

For more information about internationalizing Fusion web applications, see the "Internationalizing and Localizing Pages" chapter in the *Developing Web User Interfaces with Oracle ADF Faces*.

10.1.1 Internationalizing Integrated Excel Workbook Use Cases and Examples

You can create integrated Excel workbooks for your internationalized Fusion web application. Designing your integrated Excel workbook as part of the internationalized Fusion web application helps in its easy adaptation to specific local languages and cultures. Using resource bundles and other components, you can configure your integrated Excel workbook for a specific local language or culture by translating text and adding locale-specific components at design time, and also at runtime. For more information, see [Section 10.3, "Localization in ADF Desktop Integration."](#)

[Figure 10-1](#) shows an example of an integrated Excel workbook configured for the Japanese language.

Figure 10-1 Integrated Excel Workbook in Japanese

	E	C	D	E	F	G	H	I	J	K	L	M	N
1													
2													
3		変更済	フラグ付	ステータス	職員番号	職員名前	職位	マネージャ	日付	給料	コメント	部門	キー
4					9201	werqqq	工場A	1001	2010年9月14日	9999		34	30
5					1002	张三	工場B		2006年8月22日	1245.67			20
6					1001	lisi	工場C	1003	2006年8月28日	33333			30
7					1003	经理	工場A		2005年10月17日	9998.9		333	10
8					3110	meyyyy	办公室A	1001	2009年10月15日	6797.66			20
9					100	dd	办公室A		2006年8月22日	1245.67			20
10					1004	aa	办公室A		2009年4月3日	9876.58			10
11					1005	馬車	办公室A		2009年1月1日	7792.34			20
12					9001	fol	办公室B		2009年4月15日	3456.7			20
13					1012	hh	办公室B		2009年2月2日	4352.2			20
14					1006	cc	办公室B		2009年1月2日	8897.3			10
15					1007	dd	办公室C		2009年2月2日	1283.6			10
16					1008	ee	办公室C		2008年4月6日	7786.49			30
17					1010	ff	工場B		2009年1月3日	8888.22			20
18					2266	楼疑	办公室B		2010年12月13日	10000		20	30

10.1.2 Additional Functionality for Internationalizing Integrated Excel Workbook

After you have internationalized your integrated Excel workbook, you may find that you need to add additional functionality to configure your workbook. Following are links to other functionalities that you can use:

- Security:** Whether you are using a secure Fusion web application or not, you must be aware of security implementations in your integrated Excel workbook. For more information, see [Chapter 11, "Securing Your Integrated Excel Workbook."](#)
- Validating integrated Excel workbook:** You can configure server-side and client-side validation for the Fusion web application and the integrated Excel workbook. For more information, see [Chapter 12, "Adding Validation to an Integrated Excel Workbook."](#)
- Publishing and deploying integrated Excel workbook:** The final step after you design and validate your integrated Excel workbook is to publish and deploy it. For more information, see [Chapter 14, "Deploying Your Integrated Excel Workbook."](#)

10.2 Using Resource Bundles in an Integrated Excel Workbook

ADF Desktop Integration uses resource bundles to manage user-visible strings that appear in the ADF components of an integrated Excel workbook at design time and runtime. JDeveloper stores resource bundles in the ADF Desktop Integration project.

You can register up to twenty resource bundles containing string values you define with an integrated Excel workbook. A resource bundle must not exceed 1 megabyte. If you attempt to register more than twenty resource bundles or a resource bundle that exceeds 1 megabyte, ADF Desktop Integration writes a warning to the client-side log file and stops registration of additional resource bundles or resource bundle data after the 1 megabyte limit is reached.

For example, if resource bundle *A* measures 2 megabyte and resource bundle *B* measures 1 megabyte, ADF Desktop Integration registers the first megabyte of data in resource bundle *A* and all of the data in resource bundle *B*. For information about client-side logging, see [Section C.3.2, "About Client-Side Logging."](#)

The `Resources` workbook property specifies what resource bundles an integrated Excel workbook can use. This property specifies an array of resource bundles (`Resources` list) in the integrated Excel workbook. Each element in the array has a property that uniquely identifies a resource bundle (`Alias`) and a property that identifies the path to the resource bundle in the JDeveloper desktop integration project (`Class`). For example, `EditCustomers-DT.xlsx` in the Summit sample application for ADF Desktop Integration references the `res` resource bundle that has the following value for the `Class` property:

```
oracle.summitdi.resources.UIResources
```

More information about the `Resources` workbook property can be found in [Section A.15, "Workbook Actions and Properties."](#)

By default, ADF Desktop Integration provides a *reserved resource bundle* that supplies string key values used by many component properties at runtime. When you enable an Excel workbook to integrate it with a Fusion web application, the reserved resource bundle is registered by default with the workbook. ADF Desktop Integration uses the value `_ADFDIRes` to uniquely identify this resource bundle. Many EL expressions reference string values in this resource bundle.

If you register another resource bundle, you can replace default string key values assigned from the `_ADFDIRes` resource bundle to many of the ADF component properties.

10.2.1 How to Register a Resource Bundle in an Integrated Excel Workbook

You register a resource bundle by adding an element to the `Resources` list using the Edit Resources dialog.

Before you begin:

It may be helpful to have an understanding of how to use resource bundles. For more information, see [Section 10.2, "Using Resource Bundles in an Integrated Excel Workbook."](#)

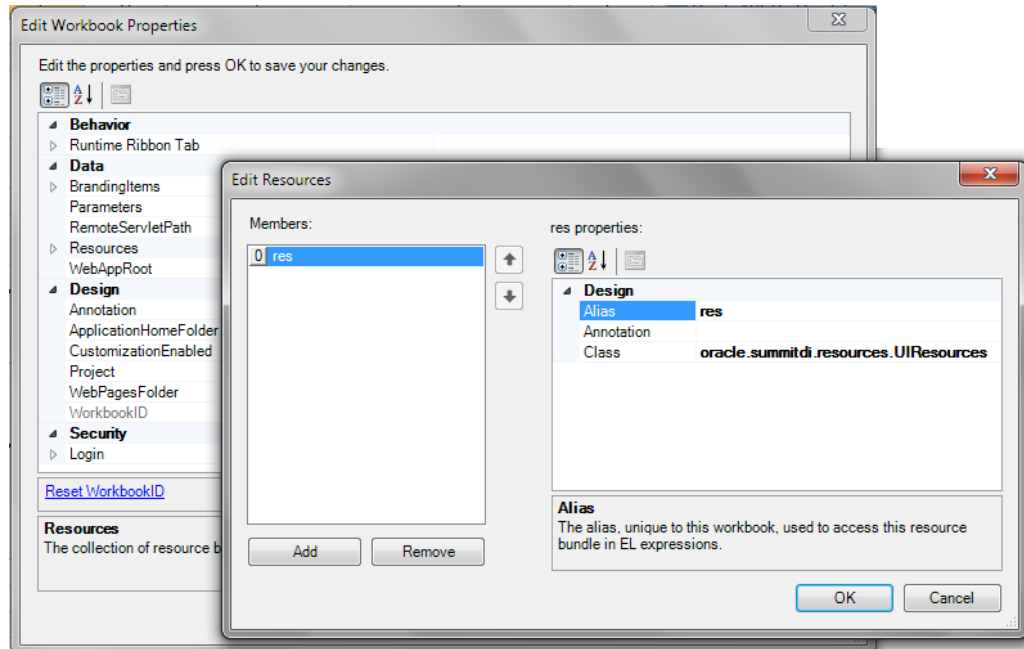
You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 10.1.2, "Additional Functionality for Internationalizing Integrated Excel Workbook."](#)

To register a resource bundle:

1. Open the integrated Excel workbook.

2. In the Workbook group of the Oracle ADF tab, click **Workbook Properties**.
3. In the Edit Workbook Properties dialog, click the browse (...) icon beside the input field for **Resources** to display the Edit Resources dialog shown in [Figure 10–2](#).

Figure 10–2 Edit Resources Dialog



4. Specify values for the resource bundle and then click **OK**.

For information about the values to specify for a resource bundle, see the entry for Resources in [Table A–21](#).

Tip: While registering a resource bundle class, do not include the file extension.

10.2.2 How to Override Resources That Are Not Configurable

The overridable resources contains several user-visible runtime strings that you cannot replace by configuring the properties of the ADF Desktop Integration components. Examples include the strings that appear in the default upload dialog illustrated in [Figure 7–11](#).

To replace these user-visible runtime strings, you create a resource bundle in your Fusion web application that contains the string keys from the overridable resource that ADF Desktop Integration supports. [Appendix E, "String Keys in the Overridable Resources"](#) lists these string keys. You define values for the string keys listed in [Appendix E](#) to override in the resource bundle you create.

Before you begin:

It may be helpful to have an understanding of how to use resource bundles. For more information, see [Section 10.2, "Using Resource Bundles in an Integrated Excel Workbook."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 10.1.2, "Additional Functionality for Internationalizing Integrated Excel Workbook."](#)

To override resources that are not configurable:

1. Create a resource bundle in your Fusion web application.

For information about creating a resource bundle, see the "Manually Defining Resource Bundles and Locales" section in the *Developing Web User Interfaces with Oracle ADF Faces*.

2. Define the string key values you want to appear at runtime in the resource bundle for the string keys listed in [Appendix E, "String Keys in the Overridable Resources."](#)
3. Set `_ADFDIRes` as the value of the **Alias** property when you register the resource bundle you created in Step 1.

For information about how to register a resource bundle, see [Section 10.2.1, "How to Register a Resource Bundle in an Integrated Excel Workbook."](#)

[Table E-1](#) describes the string keys in the overridable resources that ADF Desktop Integration supports. Supply an alternative value to the value listed in the English value column for each string key in the overridable resource. Note that override resources should not be used in component properties, they are only intended for the original usages.

10.2.3 What Happens at Runtime: Override Resources That Are Not Configurable

ADF Desktop Integration retrieves the values of string keys listed in [Table E-1](#) that you defined in the resource bundle you created. It retrieves the values of other string keys that you did not define in the resource bundle you created from the reserved resource bundle.

10.2.4 What You May Need to Know About Resource Bundles

See the following sections for additional information about resource bundles in an integrated Excel workbook.

10.2.4.1 Resource Bundle Types

ADF Desktop Integration supports use of the following types of resource bundle:

- Properties bundle (`.properties`)
- List resource bundle (`.rts`)
- Xliff resource bundle (`.xlf`)

For more information about resource bundles, see the "Manually Defining Resource Bundles and Locales" section in the *Developing Web User Interfaces with Oracle ADF Faces*.

10.2.4.2 Caching of Resource Bundles in an Integrated Excel Workbook

ADF Desktop Integration caches the values of string keys from the resource bundles that an integrated Excel workbook retrieves when it first connects to the Fusion web application. If you change a string key value in a resource bundle after an integrated Excel workbook has cached the previous value, the modified value does not appear in the workbook unless the `ClearAllData` workbook action is invoked and the end user closes and reopens the workbook so that it retrieves the modified value from the Fusion web application. For more information about the `ClearAllData` workbook action, see [Table A-20](#).

10.2.4.3 EL Expression Syntax for Resource Bundles

ADF Desktop Integration requires that you enclose the string key name in EL expressions using the `[]` characters, as in the following example:

```
#{res['StringKey']}
```

Note that ADF Desktop Integration does not support the following syntax:

```
#{res.StringKey}
```

10.3 Localization in ADF Desktop Integration

ADF Desktop Integration integrates several diverse sets of technologies. Each of these technologies provides various options for controlling the choice of natural human language when you localize your Fusion web application.

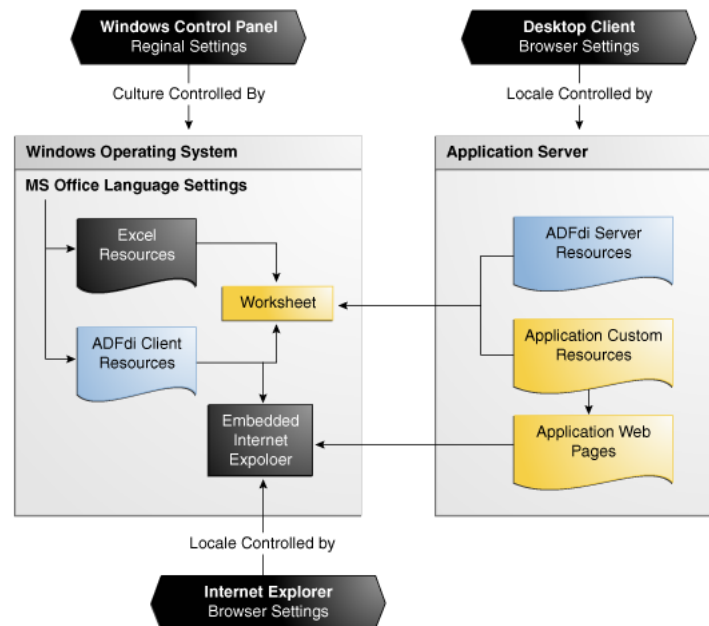
When the end user interacts with an integrated Excel workbook, various elements are involved. Each of these elements has its own set of supported languages and resource translations. In such a scenario, the translation of language is the responsibility of the respective publisher.

Table 10–1 presents a summary of elements involved and their role in translation:

Table 10–1 Summary of Localization

Area subject to localization	Determination of language to use
Microsoft operating system	Operating system language settings. You can choose the language through Regional Settings on Control Panel.
Microsoft Office	Microsoft Office language settings
Web pages displayed in ADF Desktop Integration Dialog actions	Usually controlled by Microsoft Internet Explorer's Language Preferences.
ADF Desktop Integration client resources	Microsoft Office language settings
ADF Desktop Integration server resources	Microsoft Internet Explorer language preferences
ADF Desktop Integration custom resource bundles	Microsoft Internet Explorer language preferences

Figure 10–3 illustrates how various elements involved in a Fusion web application play their role in translation.

Figure 10-3 Localization in ADF Desktop Integration

For more information about localization in ADF Desktop Integration, see the "Oracle ADF Desktop Integration Localization *whitepaper*" on OTN at:

<http://www.oracle.com/technetwork/developer-tools/adf/overview/index-085534.html>

10.3.1 Configuring Fusion Web Application to Override Server-Side Locale Settings

The server-side localization comprises of ADF Desktop Integration server resources and Application Custom Resources. By default, ADF Desktop Integration uses the client-side Internet Explorer's language preference to determine server-side localization, but you can configure the Fusion web application to determine the server-side locale. To do that, you would need to create a user preference handler and register it by adding a `UserPreferences.Handler` initialization parameter for ADF Desktop Integration servlet.

10.3.1.1 How to Create a User Preference Handler

To create a user preference handler, create a public java class with a public method of `java.util.Locale getLocale()` signature that determines the ADF Desktop Integration server-side resources locale and returns the locale as a `java.util.Locale` object.

[Example 10-1](#) shows a sample implementation of a user preference handler.

Example 10-1 Implementation of a User Preference Handler

```
public class CustomUserPrefsHandler
{
    public Locale getLocale ()
    {
        UserPref info = (UserPref)
            ADFContext.getCurrent().getSessionScope().map.get("User_Pref_Info");
        return info.getLocale();
    }
}
```


}

Note: The handler class must have a constructor with no arguments, or uses the default Java constructor.

10.3.1.2 How to Register the User Preference Handler

To register a user preference handler, add the `UserPreferences.Handler` initialization parameter for ADF Desktop Integration in `web.xml`.

Before you begin:

It may be helpful to have an understanding of how to use resource bundles. For more information, see [Section 10.3, "Localization in ADF Desktop Integration."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 10.1.2, "Additional Functionality for Internationalizing Integrated Excel Workbook."](#)

To register a User Preference Handler:

1. Open the `web.xml` file of your Fusion web application.
2. Add an initialization parameter to configure the user preference handler, as described in [Table 10–2](#).

Table 10–2 Configuring Locale User Preference

Property	Value
Name	Enter the name of the initialization parameter as follows <code>UserPreferences.Handler</code>
Value	Complete path of the handler class.

3. Save the `web.xml` file.
4. Rebuild and restart your Fusion web application.

[Example 10–2](#) shows the `web.xml` file with `UserPreferences.Handler`.

Example 10–2 web.xml File With UserPreferences.Handler

```
<servlet>
  <servlet-name>adfdiRemote</servlet-name>
  <servlet-class>
    oracle.adf.desktopintegration.servlet.DIRemoteServlet
  </servlet-class>
  <init-param>
    <param-name>UserPreferences.Handler</param-name>
    <param-value>myCompany.XYZ.CustomUserPrefsHandler</param-value>
  </init-param>
</servlet>
```

In [Example 10–2](#), `myCompany.XYZ.CustomUserPrefsHandler` is the complete path of the handler class.

Securing Your Integrated Excel Workbook

This chapter describes security related features in ADF Desktop Integration.

This chapter includes the following sections:

- [Section 11.1, "About Security In Your Integrated Excel Workbook"](#)
- [Section 11.2, "Authenticating the Excel Workbook User"](#)
- [Section 11.3, "Checking the Integrity of an Integrated Excel Workbook's Metadata"](#)
- [Section 11.4, "What You May Need to Know About Securing an Integrated Excel Workbook"](#)
- [Section 11.5, "Authorizing the Excel Workbook User"](#)

11.1 About Security In Your Integrated Excel Workbook

If you are using a Fusion web application that does not enforce authentication, the integrated Excel workbook verifies and creates a valid user session when it connects to the Fusion web application before downloading any data. The session that is established is used for each and every data transfer between the integrated Excel workbook and Fusion web application. The session is also used for web pages displayed from the integrated Excel workbook.

In a Fusion web application that is enforcing authentication, the integrated Excel workbook ensures that a valid, authenticated user session is established before transferring data to or from the web application.

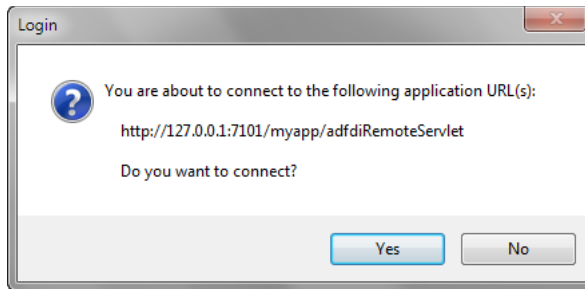
For both authenticated and non-authenticated Fusion web applications, ADF Desktop Integration relies on the establishment of cookie-based sessions. With no authentication mechanism in place, your Fusion web application is not completely safe. Hence, you should enable ADF Security in your Fusion web application before you deploy your web application with integrated Excel workbooks. For information about ADF Security, see the "Enabling ADF Security in a Fusion Web Application" chapter in the *Developing Fusion Web Applications with Oracle Application Development Framework*.

When you open the integrated Excel workbook, ADF Desktop Integration detects if the Fusion web application that the workbook runs against is a secure application and enforces authentication automatically. For authenticated web applications, the end user will always be prompted for credentials, even though the workbooks are downloaded from an authenticated web browser. Since the web browser and Excel are different operating system processes, they cannot share credentials (unless Windows native authentication is used).

11.1.1 Integrated Excel Workbook Security Use Cases and Examples

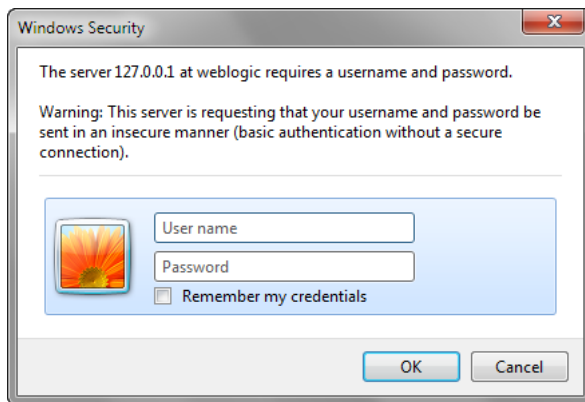
When you open the integrated Excel workbook of a secure Fusion web application opens, a Login dialog appears and prompts you to connect to the Fusion web application, as shown in [Figure 11-1](#). Note that the Login dialog also appears when the Fusion web application is not secure.

Figure 11-1 *Dialog to verify Connection*



If you click **Yes** to connect, another dialog prompts you to enter the credentials, as shown by the example in [Figure 11-2](#).

Figure 11-2 *Dialog to verify End User's Credentials*



The dialog that is shown in [Figure 11-2](#) might be different as it depends on how the Fusion web application is configured to enforce authentication. For example, if Form-based authentication is being used by the web application, then the end-user will see a browser dialog with a web page with fields for user name and password.

11.1.2 Additional Functionality for Integrated Excel Workbook in a Secure Fusion Web Application

After you have secured your integrated Excel workbook, you may find that you need to add additional functionality for your workbook. Following are links to other functionalities that you can use:

- **Validating integrated Excel workbook:** You can configure server-side and client-side validation for the Fusion web application and the integrated Excel workbook. For more information, see [Chapter 12, "Adding Validation to an Integrated Excel Workbook."](#)

- **Testing integrated Excel workbook:** Before publishing and deploying your integrated Excel workbook, you must test it. For more information, see [Chapter 13, "Testing Your Integrated Excel Workbook."](#)
- **Publishing and deploying integrated Excel workbook:** The final step after you design and validate your integrated Excel workbook is to publish and deploy it. For more information, see [Chapter 14, "Deploying Your Integrated Excel Workbook."](#)

11.2 Authenticating the Excel Workbook User

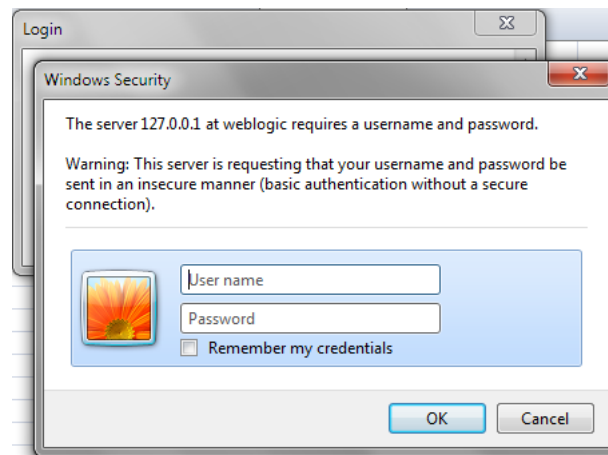
The integration of an Excel workbook with a secure Fusion web application requires an authenticated web session established between the integrated Excel workbook and the server that hosts the Fusion web application. ADF Security determines the mechanism used to authenticate the user.

If the end user opens an Excel workbook without a valid authenticated session, a login mechanism is invoked to authenticate the end user.

11.2.1 What Happens at Runtime: How the Login Method Is Invoked

After the login method is invoked, a new session between the integrated Excel workbook and the Fusion web application is created, and a modal dialog appears that contains a web browser control. This web browser control displays whatever login mechanism the Fusion web application uses. For example, if the Fusion web application uses HTTP Basic Authentication, the web browser control displays the simple dialog shown in [Figure 11-3](#).

Figure 11-3 *Dialog That Appears When a Fusion Web Application Uses Basic Authentication*



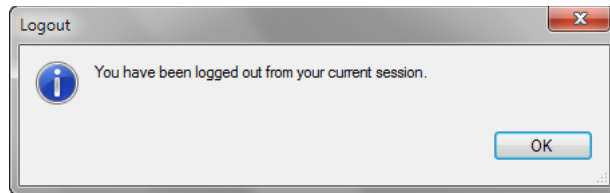
The end user enters user credentials and, assuming these are valid, an authenticated session is created and assigned to the client (the web browser control hosted by the Excel workbook).

Note: If the `Login` method is invoked when a session has already been established, it first invokes the `Logout` action internally to free that session.

11.2.2 What Happens at Runtime: How the Logout Method Is Invoked

After the logout method is invoked, a dialog appears informing users that they have logged out of the current session. The user is automatically logged out when the workbook is closed, or when the **Clear All Data** option is selected from the runtime custom tab in Excel ribbon.

Figure 11-4 *Dialog That Appears When a User Logs Out*



After logging out, the user must log in again to upload or download data.

If two or more workbooks are open (in test or runtime mode) and running against the same Fusion web application, closing one workbook does not initiate the logout mechanism. The user continues to stay logged in and may continue to work on remaining open workbooks, and can open the closed workbook without being asked for credentials again. The user is logged out when all workbooks running against the same Fusion web application are closed.

11.3 Checking the Integrity of an Integrated Excel Workbook's Metadata

ADF Desktop Integration provides a mechanism to verify that the metadata it uses to integrate an Excel workbook with a Fusion web application is not tampered with after you publish the Excel workbook for end users. It generates a hash code value and inserts the value into the ADF Desktop Integration client registry file (`adfdi-client-registry.xml`) that it also creates when you publish the integrated Excel workbook as described in [Section 14.3, "Publishing Your Integrated Excel Workbook."](#) ADF Desktop Integration stores the `adfdi-client-registry.xml` file in the `WEB-INF` directory of the Fusion web application.

If you republish the integrated Excel workbook, ADF Desktop Integration generates a new hash code value and replaces the value in the `adfdi-client-registry.xml` file. ADF Desktop Integration creates the `adfdi-client-registry.xml` file if it does not exist.

The `ApplicationHomeFolder` and `WebPagesFolder` workbook properties allow the integrated Excel workbook to identify the location of the Fusion web application's `WEB-INF` directory. You must set valid values for these properties before you can publish the integrated Excel workbook and ADF Desktop Integration can generate a hash code value.

ADF Desktop Integration generates the hash code value using most of the elements in the metadata for the workbook and the value of the `WorkbookID` workbook property. The `WorkbookID` workbook property is read-only and uniquely identifies the integrated Excel workbook. You must reset the `WorkbookID` workbook property if you create a new integrated Excel workbook by copying an existing integrated Excel workbook. ADF Desktop Integration excludes the `WebAppRoot` property from the hash code calculation since its value is expected to change at runtime.

For more information about the workbook properties discussed here, see [Table A-21](#).

Note: Tamper-check is not performed for customization-enabled workbooks.

11.3.1 How to Reset the Workbook ID

The value of the `WorkbookID` workbook property is unique to each workbook and cannot be modified by you. You can, however, reset the `WorkbookID` workbook property. You must do this when you create a new integrated Excel workbook by copying an existing integrated Excel workbook.

Before you begin:

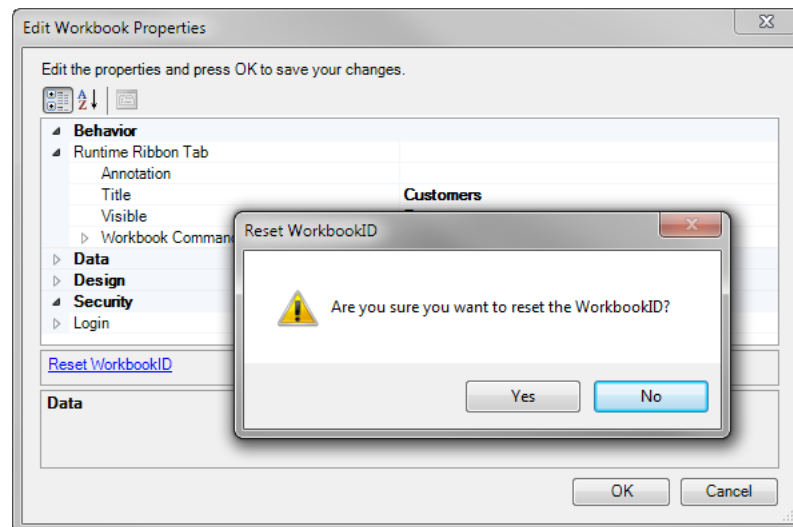
It may be helpful to have an understanding of how to verify the integrity of integrated Excel workbook's metadata. For more information, see [Section 11.3, "Checking the Integrity of an Integrated Excel Workbook's Metadata."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 11.1.2, "Additional Functionality for Integrated Excel Workbook in a Secure Fusion Web Application."](#)

To reset a workbook ID:

1. Open the integrated Excel workbook.
2. In the Workbook group of the Oracle ADF tab, click Workbook Properties.
3. In the Edit Workbook Properties dialog, click the **Reset WorkbookID** link.
4. Click **Yes** to confirm and reset the `WorkbookID` workbook property in the dialog that appears., as shown in [Figure 11–5](#).

Figure 11–5 Reset Workbook ID Dialog



5. Click OK.

11.3.2 What Happens When the Metadata Tamper-Check Is Performed

At runtime, the integrated Excel workbook regenerates the metadata hash code and provides it to the Fusion web application with the first server request. If the Fusion

web application cannot get a match on this hash code, it returns an error to the integrated Excel workbook. On receiving an error from the tamper check process, the integrated Excel workbook reports this failure to the end user and closes the integration framework.

11.4 What You May Need to Know About Securing an Integrated Excel Workbook

Note the following points about securing an integrated Excel workbook with a Fusion web application:

- **Data security**

If you save an Excel workbook containing data downloaded from a Fusion web application to a location, such as a network directory, where other users can access the Excel workbook, the data stored in the Excel workbook is accessible to other users.
- **Security in Microsoft Excel**

You can enhance the security of an integrated Excel workbook using Excel's functionality to set a password on a workbook. It prevents unauthorized users from opening or modifying the workbook. For more information about Excel security features, see Excel's documentation.
- **Integrated Excel workbooks can be configured to cache data, as described in [Section 15.2, "Restore Server Data Context Between Sessions."](#) Make sure that you do not cache sensitive data in the integrated Excel workbook.**
- **If the Fusion web application is running on the https protocol, you may receive a certificate error while connecting from an integrated Excel workbook. You can either install the required certificate using Microsoft Internet Explorer, or choose to continue to log in and connect to the web application.**
- **End users that download integrated Excel workbooks using Microsoft Internet Explorer may be prompted unexpectedly for credentials before the Excel application is visible, and then prompted again once the workbook opens. This may occur when the web application is configured to use certain authentication methods like Basic or Digest. The extra prompt is due to Excel making an OPTIONS request on the web directory containing the workbook.**

To avoid the extra login prompt, end users can choose to save the workbook locally instead of opening it directly from the browser.
- **For a non-authenticated Fusion web application, end-users will not be prompted to log in. However if the application uses the https protocol, then end users may briefly see a login dialog appear when the first connection is established to the web application. Workbook developers can control the size of the dialog with the `Workbook.Login.WindowSize` property.**

If you are an administrator, you should also see the "What You May Need to Know About Configuring Security in a Fusion Web Application" section in *Administering Oracle ADF Applications*.

11.5 Authorizing the Excel Workbook User

ADF Desktop Integration enforces view permission for integrated Excel worksheets through page definition authorization. At runtime, end users without proper permissions for a page definition (binding container) are prevented from interacting

with the associated integrated Excel worksheet. Any attempt to interact with an unauthorized binding container (for example, download or submit data) is aborted, the end user is informed of the authorization failure, and all ADF Desktop Integration activity on the worksheet is disabled. No further interaction with the ADF Desktop Integration-disabled worksheet is possible until a new user session is established. To allow end users to interact with the integrated Excel worksheet, assign them the roles that have been granted access to the page definition.

If you have upgraded ADF Desktop Integration from an old version, you may need to review the resource grants for all of the page definitions that are used with integrated Excel worksheets. For example, if your Fusion web application supports authorization, and you have a page definition `myWorksheetPageDef.xml` that has no resource grants and is used by one (or more) integrated Excel worksheets, then you need to assign end users the roles that have been granted access to the page definition. When transitioning to the new version of ADF Desktop Integration, you may find it helpful to temporarily create resource grants for the worksheet page definitions that are granted to authenticated-role, or some other generic role, allowing you to run those worksheets as before while you fine tune your roles and resource associations.

For more information about authorization, roles, and resource grants, see the "Enabling ADF Security in a Fusion Web Application" chapter in the *Developing Fusion Web Applications with Oracle Application Development Framework*.

Note: ADF Desktop Integration does not enforce authorization of other resource types, such as entity object-level, entity attribute-level, ADF Method, Insert while New, Panel Tab, and Task Flow authorization.

You can configure resources and grants from the Resource Grants page of the overview editor for the `jazn-data.xml` file. For more information, see the "Defining ADF Security Policies" section in the *Developing Fusion Web Applications with Oracle Application Development Framework*.

On an authorization failure, the end user receives an error message, such as the following, and ADF Desktop Integration in the worksheet is disabled:

```
ADFDI-05589 You are not authorized to use this worksheet for interacting
with the Fusion web application.
```

11.5.1 What You May Need to Know About ADF Desktop Integration-Disabled Worksheet

The following limitations apply to an ADF Desktop Integration-disabled worksheet:

- All ADF buttons, worksheet-level ribbon commands, and worksheet-level events are disabled.
- If the authorization failure occurs during worksheet initialization, no form labels, table column headers, or buttons are drawn on the worksheet.
- If the authorization failure occurs for an initialized worksheet, all ADF buttons are disabled, but other worksheet components (such as ADF Input Text and ADF Table) are not affected and are left visually unchanged.
- End user can perform standard Excel interactions on the disabled worksheet. The user may alter the data in an ADF Table component in the worksheet, but the Changed column will not be updated.

- There is no impact on workbook-level commands. End users can continue to use the following commands: Login, Logout, About, Edit Options, and Clear All Data.

An ADF Desktop Integration-disabled worksheet is automatically enabled when the end user reopens the integrated Excel workbook and establishes a new session, provided the new session is authorized. Logging out, and then logging in again, also re-enables ADF Desktop Integration in a disabled integrated Excel worksheet.

Adding Validation to an Integrated Excel Workbook

This chapter describes how to provide server-side and client-side validation for your integrated Excel workbook, how to report errors such as validation failures and data conflict, and how to configure error reports using custom error handler.

This chapter includes the following sections:

- [Section 12.1, "About Adding Validation to an Integrated Excel Workbook"](#)
- [Section 12.2, "Providing Server-Side Validation for an Integrated Excel Workbook"](#)
- [Section 12.3, "Providing Client-Side Validation for an Integrated Excel Workbook"](#)
- [Section 12.4, "Error Reporting in an Integrated Excel Workbook"](#)
- [Section 12.5, "Providing a Row-by-Row Status on an ADF Table Component"](#)
- [Section 12.6, "Adding Detail to Error Messages in an Integrated Excel Workbook"](#)
- [Section 12.7, "Handling Data Conflicts When Uploading Data from a Workbook"](#)

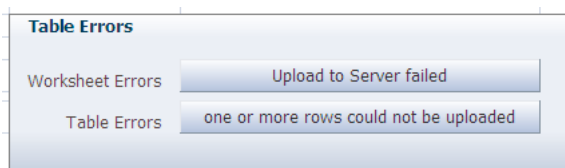
12.1 About Adding Validation to an Integrated Excel Workbook

You configure server-side and client-side validation for the Fusion web application and the integrated Excel workbook to make use of the validation options offered by the ADF Model layer and Microsoft Excel. In addition to these validation options, you can make use of components in ADF Desktop Integration to return error messages from the Fusion web application, to provide status on the results of component actions, and to manage errors that may occur when data is changed in an integrated Excel workbook conflict with data hosted by the Fusion web application.

Adding validation to your integrated Excel workbook gives you several benefits. You can create validation rules in your Fusion web application and in your integrated Excel workbook to validate data entry by the end user.

12.1.1 Integrated Excel Workbook Validation Use Cases and Examples

Validation rules protect the server by stopping invalid data from getting uploaded. For example, using worksheet and table validation rules, you can capture invalid data when you upload it to the server, as shown in [Figure 12-1](#).

Figure 12–1 Workbook and Table Validation Errors at Runtime

12.1.2 Additional Functionality for Adding Validation to an Integrated Excel Workbook

After you have applied validation rules in your integrated Excel workbook, you may find that you need to add additional functionality to configure your workbook. Following are links to other functionalities that you can use:

- **Testing integrated Excel workbook:** Before publishing and deploying your integrated Excel workbook, you must test it. For more information, see [Chapter 13, "Testing Your Integrated Excel Workbook."](#)
- **Publishing and deploying integrated Excel workbook:** The final step after you design and validate your integrated Excel workbook is to publish and deploy it. For more information, see [Chapter 14, "Deploying Your Integrated Excel Workbook."](#)

12.2 Providing Server-Side Validation for an Integrated Excel Workbook

ADF Desktop Integration uses the validation rules that the ADF Model layer sets for a binding's attributes. Data that the end user enters or edits in one of the ADF Desktop Integration components, such as the ADF Table component, can be validated against set rules and conditions in the Fusion web application. For general information about defining validation rules in Oracle ADF, see the "Defining Validation and Business Rules Declaratively" chapter in the *Developing Fusion Web Applications with Oracle Application Development Framework*.

For information about adding ADF Model layer validation, see the "Defining Validation Rules in the ADF Model Layer" section in the *Developing Fusion Web Applications with Oracle Application Development Framework*.

12.3 Providing Client-Side Validation for an Integrated Excel Workbook

ADF Desktop Integration does not provide client-side validation. However, you can use Excel's data validation features to control the type of data or the values that end users enter into a cell. These features allow you to restrict data entry to a certain range of dates, limit choices by using a list, or ensure that only positive whole numbers are entered in a cell. For example, you could configure the `ZipCode` field in the `EditWarehouses-DT.xlsx` workbook so that users can enter only whole numbers in the cells of this field.

If you apply custom validation to columns within an ADF Table component, the validation is propagated when the ADF Table component's columns are populated at runtime. Note, however, that ADF Desktop Integration overwrites at runtime any custom validation applied to columns that reference the `TreeNodeList` subcomponent at design time. This is because ADF Desktop Integration applies its own list-constraint validation, which is invoked at runtime.

Note:

- Excel displays error messages when a validation fails; these error messages cannot be localized.
- ADF Desktop Integration does not support server-side validation warnings. Validation warnings, set for rules defined in the Fusion web application, are not displayed by the integrated Excel workbook.

For more information about data validation in Excel, see Excel's documentation.

12.4 Error Reporting in an Integrated Excel Workbook

The server that hosts the Fusion web application you integrate your Excel workbook with can return error messages to end users that provide feedback on the results of operations. The error messages returned can be one many types: validation failures, conflict errors, deleted records, and so on.

To add automatically configured error components to the integrated Excel worksheet, see [Section 6.8, "Inserting Error Components."](#) To create error components manually, follow the steps described in subsequent sections.

12.4.1 Error Reporting Using EL Expressions

To return error message summaries to end users, you must set an EL expression for the `Value` property of an ADF Output Text component. At runtime, the ADF Output Text component displays the error message summary to the end user if an error occurs.

The type of EL expression that you set for the `Value` property of the ADF Output Text component depends on whether you want to return error message summaries generated by action sets invoked on a worksheet, or by actions invoked by other components such as the ADF Table and ADF Read-only Table components. The following EL expression displays error message summaries which are returned during the invocation of an action set on a worksheet:

```
{worksheet.errors}
```

At runtime, the previous error message summary is cleared (if one existed) when the action set starts the invocation. If no errors occur during invocation, error message remains blank. If an error occurs, the ADF Output Text component displays the error message summary.

An alternative approach to returning error message summaries generated by action sets invoked on a worksheet is to set `{worksheet.errors}` as the value for an action set's `Alert.FailureMessage` property. This approach displays the generated error message summary in a dialog.

Components such as the ADF Table and ADF Read-only Table components that have actions which interact with the Fusion web application can also return error message summaries. Set the following EL expression for the `Value` property of the ADF Output Text component or for an action set's `Alert.FailureMessage` property:

```
{components.componentID.errors}
```

where `componentID` refers to the ID of the component (ADF Table or ADF Read-only Table component) that invokes the action.

The `EditCustomers-DT.xlsx` file in the Summit sample application for ADF Desktop Integration demonstrates how to return error message summaries generated by action sets invoked on a worksheet and by the actions of an ADF Table component. [Figure 12-2](#) shows these EL expressions in design mode.

Figure 12-2 EL Expressions to Return Error Messages in an ADF Output Text Component

<code>#{res['excel.worksheet.errors']}</code>	<code>#{worksheet.errors}</code>
<code>#{res['excel.customers.table.errors']}</code>	<code>#{components.TAB1598787742.errors}</code>

12.4.2 Error Reporting Using Component Actions

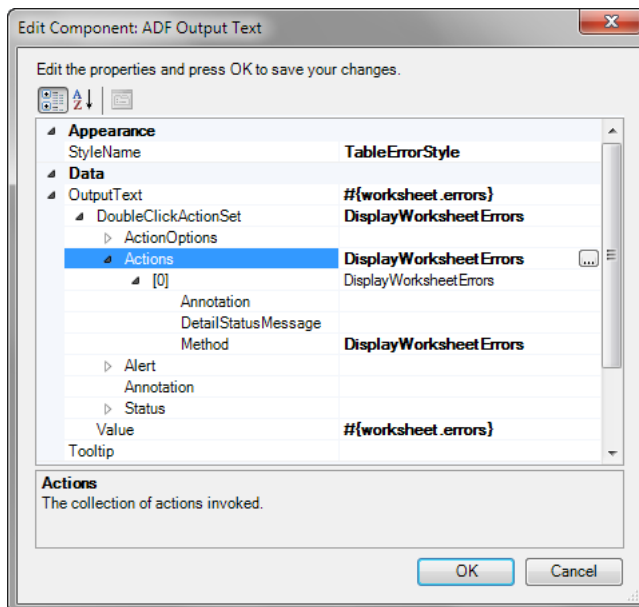
ADF Desktop Integration provides actions that display error details generated by an ADF Table component or an integrated Excel worksheet.

The action set in which you invoke one of these actions must include only one action. In general, action sets clear error labels and message lists when invoked. An action set that invokes one of the following actions returns error labels and message lists to the end user:

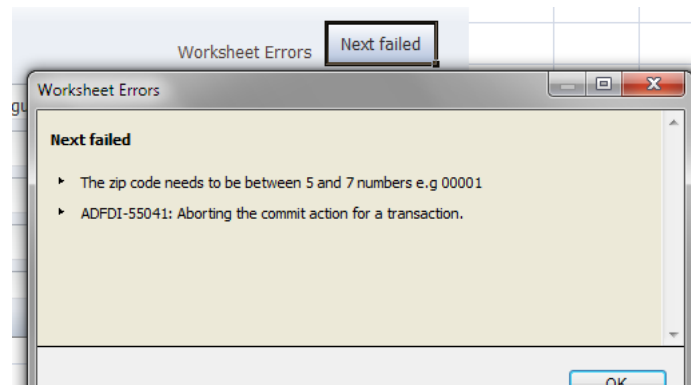
- Worksheet's `DisplayWorksheetErrors` action

To display a worksheet's error messages, configure the action set of a component on the worksheet or the worksheet ribbon button to invoke this action. For example, [Figure 12-3](#) shows the Edit Component: ADF Output Text dialog configuring the `DisplayWorksheetErrors` action as a `DoubleClickActionSet` item for an ADF Output Text component on the worksheet.

Figure 12-3 DisplayWorksheetErrors Action



At runtime, double-clicking the ADF Output Text component invokes the `DisplayWorksheetErrors` action as shown in [Figure 12-4](#).

Figure 12–4 Runtime View of DisplayWorksheetErrors action

For more information about the Worksheet's `DisplayWorksheetErrors` action, see [Section A.16, "Worksheet Actions and Properties."](#)

- ADF Table component's `DisplayRowErrors` action

To display row-level failures that occur in an ADF Table component, invoke this action. Row-level failures occur when end user invokes the following actions:

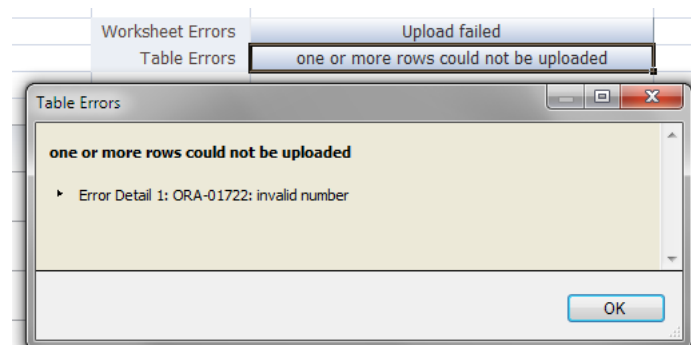
- Upload
- UploadAllOrNothing
- DeleteFlaggedRows
- `DoubleClickActionSet` invoked from an ADF Table component column

For more information about using this action, see [Section 12.5, "Providing a Row-by-Row Status on an ADF Table Component."](#)

- ADF Table component's `DisplayTableErrors` action

To display table-level failures that occur in an ADF Table component, invoke this action. It is not intended that an ADF Table component column's `DoubleClickActionSet` invoke this action. Instead add this action to an action set that returns error messages to end users when failures occur during invocation of the action binding specified by an ADF Table component's `BatchOptions.CommitBatchActionID` property.

At runtime, double-clicking the ADF Output Text component invokes the `DisplayTableErrors` action as shown in [Figure 12–5](#).

Figure 12–5 Runtime View of DisplayTableErrors action

For more information about ADF Table component actions, see [Section A.12, "ADF Table Component Properties and Actions."](#)

12.5 Providing a Row-by-Row Status on an ADF Table Component

The ADF Table component provides a mechanism to indicate to end users whether rows from the ADF Table component have been processed successfully or not after invocation of following ADF Table component actions:

- DeleteFlaggedRows
- Upload
- UploadAllOrNothing
- DoubleClickActionSet invoked from an ADF Table component's column

The ADF Table component populates the `_ADF_StatusColumn` column with the status for each row following the invocation of the ADF Table component action. For example, it populates the `_ADF_StatusColumn` column with the upload status for each row following the invocation of the ADF Table component's `Upload` action.

[Figure 12–6](#) shows rows in an ADF Table component where the values in those rows have been changed, as indicated by the upward pointing arrows in the **Changed** column. In the **ZipCode** column, a text value has been entered where a number value is expected.

Figure 12–6 ADF Table Component with Changed Rows Before Upload

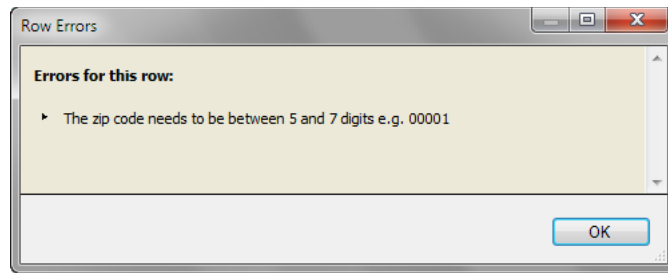
Changed	Flagged	Status	Name	Credit Rati	Phone	Address	City	State	Zip Code	Country
			Unisports	Excellent	55-2066101	72 Via Bahia	Sao Paulo			Brazil
			Simms Athletics	Poor	81-20101	Takashi	Osaka			Japan
▲			Delhi Sports	Good	91-10351	Chanakya	New Delhi		text	India
▲			Womansports	Fair	1-206-104-0103	281 King Street	Seattle	Washington	98101	USA

[Figure 12–7](#) shows the same rows in the ADF Table component after invocation of the ADF Table component's `Upload` action. The ADF Table component populates the `_ADF_StatusColumn` column (labeled **Status** in this example at runtime) with a message indicating whether the row updated successfully or not.

Figure 12–7 ADF Table Component with Changed Rows After Upload

Changed	Flagged	Status	Name	Credit Rati	Phone	Address	City	State	Zip Code	Country
			Unisports	Excellent	55-2066101	72 Via Bahia	Sao Paulo			Brazil
			Simms Athletics	Poor	81-20101	Takashi	Osaka			Japan
▲		Update failed	Delhi Sports	Good	91-10351	Chanakya	New Delhi		text	India
		Row updated successfully	Womansports	Fair	1-206-104-0103	281 King Street	Seattle	Washington	98101	USA
			Kam's Sporting Goods	Excellent	852-3692888	95 Queensway	Hong Kong			Hong Kong

By default, the `_ADF_StatusColumn` column's `DoubleClickActionSet` is configured to invoke the ADF Table component's `DisplayRowErrors` action. When end users double-click a row in this column at runtime, the ADF Table component invokes the `DisplayRowErrors` action. This action displays a dialog with a list of errors for that row if errors exist. If no errors exist, the dialog displays a message to indicate that no errors occurred. [Figure 12–8](#) shows the dialog that appears if the end user double-clicks the cell in [Figure 12–7](#) that displays `Update failed` in the **Status** column.

Figure 12–8 Dialog Displaying Row Error Message

For more information about the `_ADF_StatusColumn` column, see [Section 7.12, "Special Columns in the ADF Table Component."](#)

12.6 Adding Detail to Error Messages in an Integrated Excel Workbook

You can configure your Fusion web application to report errors using a custom error handler to provide more detail to the error messages displayed to end users in an integrated Excel workbook.

To implement this functionality, the custom error handler must override the `getDetailedDisplayMessage` method to return a `DCErrorMessage` object. At runtime, ADF Desktop Integration detects the custom error handler and invokes the `getHtmlText` method on the `DCErrorMessage` object. ADF Desktop Integration includes the HTML returned by the `getHtmlText` method in the error message list as detail.

For more information about creating a custom error handler, see the "Customizing Error Handling" section of the *Developing Fusion Web Applications with Oracle Application Development Framework*.

12.7 Handling Data Conflicts When Uploading Data from a Workbook

If one of your end users (User X) makes changes to a row of data downloaded from a Fusion web application to an Excel workbook, and another end user (User Y) in a different session modifies the same row in the Fusion web application after User X downloads the row, User X may encounter an error while uploading the modified row, as the changes conflict with those that User Y made. Depending on the configuration of your Fusion web application, User X may receive `RowInconsistentException` type error messages. For information about how to configure your Fusion web application to protect your data, see the "How to Protect Against Losing Simultaneously Updated Data" section in the *Developing Fusion Web Applications with Oracle Application Development Framework*.

To resolve this conflict in the integrated Excel workbook, User X needs to download the most recent version of data from the Fusion web application. However, invoking the ADF Table component's `Download` action causes the component to refresh all data that the component hosts in the Excel workbook. This may overwrite other changes that User X made that do not generate conflict error messages. To resolve this scenario, you can expose the ADF Table component's `DownloadFlaggedRows` action. When invoked, this action downloads data only for the rows that the end user flags for download. Using this action, User X can resolve the conflict issues and upload his modified data.

[Chapter 15, "Using an Integrated Excel Workbook Across Multiple Web Sessions and in Disconnected Mode"](#) provides information about using an integrated Excel workbook across multiple sessions. For information about flagging rows, see

[Section 7.11.2, "Row Flagging in an ADF Table Component."](#) For information about invoking component actions, see [Section 8.2.2, "How to Invoke Component Actions in an Action Set."](#) For more information about the components that the ADF Table component supports, see [Section A.12, "ADF Table Component Properties and Actions."](#)

12.7.1 How to Configure a Workbook to Handle Data Conflicts When Uploading Data

You specify a row-specific attribute of the tree binding for the `RowData.ChangeIndicatorAttribute` property to determine whether a row has been modified by another user since the row was last downloaded by the ADF Table component.

To configure a workbook to handle data conflicts:

1. Open the integrated Excel workbook.
2. Select any cell of the ADF Table component and click **Edit Properties** in the **Oracle ADF** tab.
3. In Edit Component: ADF Table dialog, for the `RowData.ChangeIndicatorAttribute` property, specify the row-specific attribute of the tree binding that you use to determine whether a row has been modified by another user since the row was last downloaded by the ADF Table component in your integrated Excel workbook.
4. Click **OK**.

12.7.2 What Happens at Runtime: How Data Conflicts Are Handled

The ADF Table component caches the original value of the row-specific attribute of the tree binding that you specified as a value for `RowData.ChangeIndicatorAttribute` when it invokes the `RowDownSync` action. When the ADF Table component invokes the `RowUpSync` action, it checks if the value of the binding hosted by the Fusion web application and the original value cached by the ADF Table component differ. If they differ, it indicates data conflict, as changes have been made to the value of the binding hosted by the Fusion web application since the ADF Table component downloaded the value of the binding.

Testing Your Integrated Excel Workbook

This chapter describes how to test and validate the integrated Excel workbooks as you configure it, and how to run a server ping test.

This chapter includes the following sections:

- [Section 13.1, "About Testing Your Integrated Excel Workbook"](#)
- [Section 13.2, "Testing Your Fusion Web Application"](#)
- [Section 13.3, "Validating the Integrated Excel Workbook Configuration"](#)
- [Section 13.4, "Testing Your Integrated Excel Workbook"](#)
- [Section 13.5, "Running a Server Ping Test"](#)

13.1 About Testing Your Integrated Excel Workbook

Testing an integrated Excel workbook before you publish and deploy it to your end users enables you to verify that the functionality you configure behaves as you intend. Before you test your integrated Excel workbook, test the Fusion web application with which you integrate the Excel workbook.

Before you deploy the integrated Excel workbook, you should validate it and test its integration with your Fusion web application. Testing an integrated Excel workbook includes the following processes:

- Validating the integrated Excel workbook
- Running the integrated Excel workbook in test mode

13.1.1 Integrated Excel Workbook Testing Use Cases and Examples

To test your integrated Excel Workbook, click the **Run** button on the **Oracle ADF** tab, and click the **Stop** button to return to the design mode. [Figure 13–1](#) shows the buttons of the **Oracle ADF** tab in design mode and in test mode.

Figure 13–1 Run and Stop buttons in Oracle ADF tab



Buttons in Design mode

Buttons in Test mode

13.1.2 Additional Functionality of Testing an Integrated Excel Workbook

After you have validated and tested your integrated Excel workbook, you may find that you need to add additional functionality for your workbook. Following are links to other functionalities that you can use:

- **Publishing your integrated Excel workbook:** After you test and validate your workbook, you must publish it. For more information see, [Section 14.3, "Publishing Your Integrated Excel Workbook."](#)
- **Deploying your integrated Excel workbook:** After you publish your workbook, you may wish to deploy it with your Fusion web application. For more information, see [Section 14.4, "Deploying a Published Workbook with Your Fusion Web Application."](#)
- **Working in disconnected mode:** After downloading data to the integrated Excel workbook, sometimes end users may need to use the workbook when a server connection may not be available, or after the server session has expired. For more information, see [Section 15.1, "About Disconnected Workbooks."](#)

13.2 Testing Your Fusion Web Application

Test the Fusion web application that you integrate your Excel workbook with before you start testing the integrated Excel workbook. For information about testing a Fusion web application, see the *Developing Fusion Web Applications with Oracle Application Development Framework*. Verify that the Fusion web application you want to integrate an Excel workbook with, supports ADF Desktop Integration by carrying out the procedure described in [Section C.1, "Verifying That Your Fusion Web Application Supports ADF Desktop Integration."](#) You may also want to test the view instances of the ADF application module before you test the Fusion web application. For more information about testing ADF application module, see the "Using the Oracle ADF Model Tester for Testing and Debugging" section of the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

If the integrated Excel workbooks are not saved in Application Sources directory of the Fusion web application, then before you run the Fusion web application in JDeveloper, ensure that all integrated Excel workbooks and the Excel application are closed. The application deployment may fail if it encounters locked files as Excel locks the files that it opens.

Tip: If you plan to test integrated Excel workbooks that you downloaded from web pages of the Fusion web application, you should republish them before redeploying the application. Republishing the workbooks ensures that you have their latest versions.

If you make changes to the Fusion web application to resolve problems identified by testing the application, you need to:

- Close Excel and all integrated Excel workbooks. The application deployment may fail if it encounters locked files, as Excel locks the files that it opens.
- Rebuild the JDeveloper project where you develop the Fusion web application.
- Run the Fusion web application.
- Reload the page definition files that are associated with the integrated Excel workbook. Click the **Refresh Bindings** button in **Oracle ADF** tab of the integrated Excel workbook to reload the page definition files.

These steps make sure that the changes in the Fusion web application are available to the integrated Excel workbook. For information about how to reload a page definition file, see [Section 4.3.3, "How to Reload a Page Definition File in an Excel Workbook."](#)

13.3 Validating the Integrated Excel Workbook Configuration

ADF Desktop Integration provides a set of validation rules for the integrated Excel workbook configuration. After creating your integrated Excel workbook, you may validate the workbook before you proceed for testing or deployment.

13.3.1 How to Validate the Integrated Excel Workbook Configuration

You should validate the integrated Excel workbook configuration before testing or deploying the workbook.

To validate the integrated Excel workbook configuration:

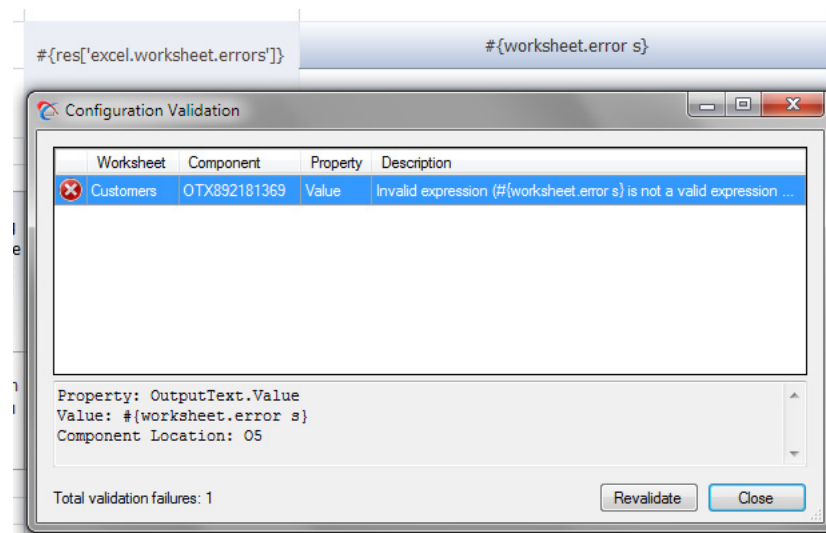
1. Open the integrated Excel workbook.
2. In your integrated Excel workbook, click the **Oracle ADF** tab.
3. In the Test group, click **Validate**.

The Configuration Validation dialog appears listing all your warnings and errors.

4. If any warning or error is displayed, click to select it. A description of the warning or error message is displayed in the dialog.

For example, [Figure 13–2](#) illustrates a validation failure message of an invalid EL expression.

Figure 13–2 Invalid EL Expression Resulting in a Validation Failure



If no warning or error appears, click **Close** to close the dialog.

Note: You may continue to keep the Configuration Validation dialog open while you resolve the validation failures. To verify whether you have resolved an error or a warning, click **Revalidate** to run the validation rules again.

13.3.2 What Happens When You Validate the Integrated Excel Workbook Configuration

When you validate the workbook at design time, ADF Desktop Integration validates all workbook configuration properties, including worksheet and worksheet component properties, against defined validation rules. Any and all validation failures (errors and warnings) are listed in the Configuration Validation dialog. Each validation failure, when selected, provides contextual information about the failure, and provides enough detail to locate and fix each validation failure.

The Configuration Validation dialog provides the following information for each validation failure:

- Severity type (error or warning)
- Name of the worksheet. The word **Workbook** is displayed if the validation failure does not correspond to a particular worksheet.
- Worksheet component ID ("Workbook" or "Worksheet" if the validation failure does not correspond to a particular worksheet component)
- Property containing the validation failure
- Description of the validation failure (error or warning)

When you select a specific failure entry in the dialog, the dialog displays additional details about the failure including:

- Full property context path
- Property value

Certain validation rules may result in multiple distinct failures. For example, when an expression is being validated, different validation failures occur based on expression type, expression syntax, or the location in which the property is exposed in the workbook configuration.

For example, consider the following expression value:

```
{bindings.EmpView1.hints.Empno.label}
```

The expression value is legal when used within a column header label inside of a table component, but the same expression value is illegal when specified as part of the `Worksheet.Title` expression.

Note: If `Enabled` is set to `False` for a group of workbook configuration properties, validation of other property values within the same group is skipped.

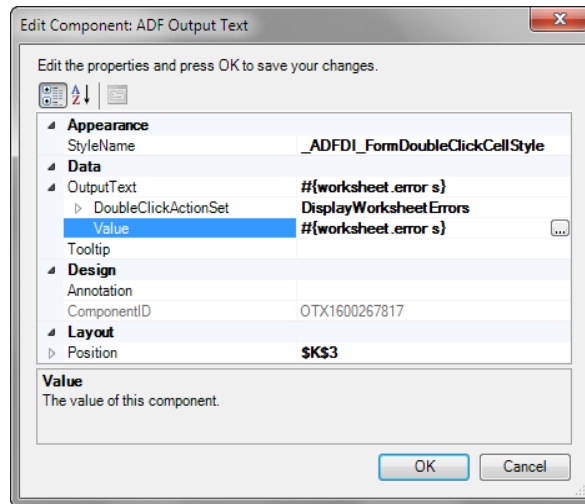
13.3.3 How to Fix Validation Failures

When you validate your workbook, you might get validation failures or warning messages, which you can fix easily by following these steps:

1. Identify the component that gave the error or warning message.
In [Figure 13-2](#), note that the ADF Output Text component generated the error. In the worksheet, search for Output Text component that uses `{worksheet.errors}` expression.
2. Open the property editor of the component.
3. Navigate to the invalid property value identified by the full property context path.
4. Edit the property value to resolve the validation failure.

Figure 13–3 illustrates the ADF Output Text component property editor and its invalid EL expression.

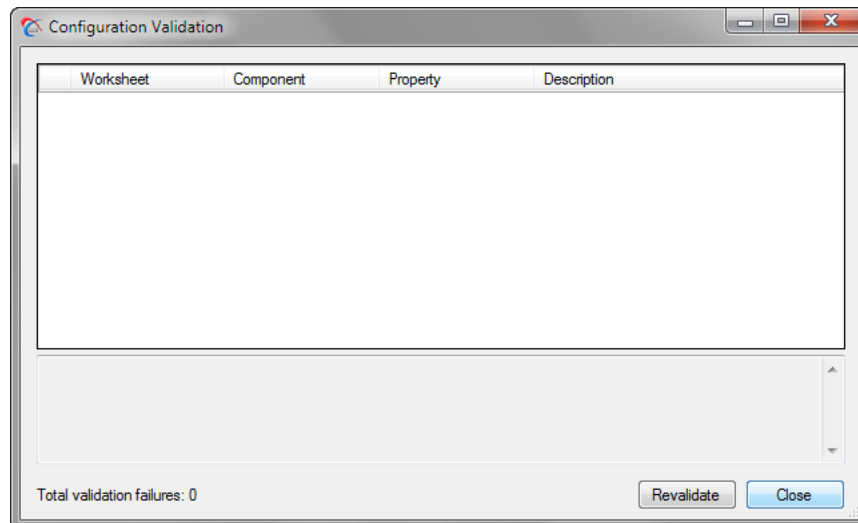
Figure 13–3 Resolving Validation Failure



5. Revalidate the workbook to verify whether the validation failure has been resolved. Click **Revalidate** to run the validation rules again.
6. After fixing all validation failures, click **Close** to close the Configuration Validation dialog.

Figure 13–4 illustrates the Configuration Validation dialog with no warnings or error messages.

Figure 13–4 Configuration Validation Dialog with No Validation Failure Messages



13.3.4 How to Log the Integrated Excel Workbook Configuration Validation Failures at Runtime

By default, there is no runtime validation of integrated Excel workbook configuration. However, you may log validation failures at runtime by setting the client log level to `Error` or `Warning`. For more information about enabling client-side logging, see

[Section C.3.2, "About Client-Side Logging."](#)

13.4 Testing Your Integrated Excel Workbook

As you configure your Excel workbook to integrate with a Fusion web application, you can switch to test mode from design mode to test the functionality that you add to the workbook. You use the **Oracle ADF** tab to switch to test mode from design mode and from design mode to test mode.

Test mode enables you to test the functionality of your integrated Excel workbook as you configure it incrementally. It also enables you to view the integrated Excel workbook from the end user's perspective, as test mode corresponds to what end users see when they view and run the published integrated Excel workbook. The difference between an integrated Excel workbook in test mode and a published integrated Excel workbook is that the ADF Desktop Integration task pane is not available to users of the published integrated Excel workbook.

For more information about test mode and design mode, see [Section 5.1, "About Development Tools."](#)

There are some differences between the test mode and the runtime mode when you run the integrated Excel workbook. [Table 13–1](#) lists these differences.

Table 13–1 Differences between Test mode and Runtime mode

Test mode	Runtime mode
Does not perform tamper check	Performs tamper check
Does not display the connection confirmation dialog	Displays the connection confirmation dialog
Displays the Oracle ADF ribbon tab	Does not display Oracle ADF tab
Allows you to switch back to design mode	Does not allow you to switch back to design mode

ADF Desktop Integration can generate log files that capture information based on events triggered by an integrated Excel workbook. For more information about these log files, see [Appendix C, "Troubleshooting an Integrated Excel Workbook."](#)

Note: Before you start testing the integrated Excel workbook, ensure that:

- The Fusion web application is running.
 - The ping to server is successful, and the server is configured for ADF Desktop Integration.
 - The ADF Desktop Integration version of the server and the client are same, or compatible. Note that an older client (version X) works fine with the newer server (version X+1) during the transition period, but an older server (version X) may not work with a newer client (version X+1).
-
-

To run an integrated Excel workbook in test mode:

- To test and run an integrated Excel workbook, click the **Run** button on the **Oracle ADF** tab.

The integrated Excel workbook switches to test mode from design mode. Before starting the test mode, ADF Desktop Integration clears all design time component placeholders.

To stop test mode and return the integrated Excel workbook to design mode:

- In the integrated Excel workbook that you are testing, click the **Stop** button on the Oracle ADF tab.

The integrated Excel workbook switches to design mode from test mode. Before switching back to design mode, ADF Desktop Integration removes all visible and cached data from all parts of the workbook, and then redraws the design time component placeholders.

Note: When the end user tries to close the integrated Excel workbook, Microsoft Excel prompts a dialog to save the workbook even if the end user has not modified it after opening it. This behavior is expected because ADF Desktop Integration modifies an integrated Excel workbook each time the end user opens it.

13.5 Running a Server Ping Test

The server ping test enables you to check the version of ADF Desktop Integration Remote Servlet in a running system. It also helps to confirm that the remote servlet is loaded and responding. For Fusion web applications that enforce authentication, you can use the ping test to confirm that the proper authentication configuration is in place for the ADF Desktop Integration servlet URL.

After running the Fusion web application and logging in as a valid user, open a URL in the following format to verify whether the remote servlet is running:

`http://<hostname>:<portnumber>/<context-root>/adfdiRemoteServlet`

For example, if you run the Summit sample application for ADF Desktop Integration, open the following URL:

`http://127.0.0.1:7101/summit/adfdiRemoteServlet`

Figure 13–5 shows the remote servlet response page.

Figure 13–5 Remote Servlet Response

Oracle ADF Desktop Integration 12c

Diagnostic Information	
Status	Active
Version	12.1.3.0.0.7731
Build	ADFDI_MAIN_NT_130522.1901

[About ADF Desktop Integration](#)

Copyright © 1997, 2013, Oracle and/or its affiliates. All rights reserved.

Note: A valid user session is required to run the server ping test. If authentication is enabled for the web application, you will be prompted for valid credentials to log in.

Deploying Your Integrated Excel Workbook

This chapter describes how to publish and deploy a workbook integrated with a Fusion web application to end users, and how to pass parameters from the Fusion web application to the integrated Excel workbook.

This chapter includes the following sections:

- [Section 14.1, "About Deploying Your Integrated Excel Workbook"](#)
- [Section 14.2, "Making ADF Desktop Integration Available to End Users"](#)
- [Section 14.3, "Publishing Your Integrated Excel Workbook"](#)
- [Section 14.4, "Deploying a Published Workbook with Your Fusion Web Application"](#)
- [Section 14.5, "Passing Parameter Values from a Fusion Web Application Page to a Workbook"](#)
- [Section 14.6, "Customizing Workbook Integration Metadata at Runtime"](#)

14.1 About Deploying Your Integrated Excel Workbook

After you finish development of your integrated Excel workbook, you make the final integrated Excel workbook available to end users by deploying the resulting Fusion web application to an application server. Before you deploy a finalized Excel workbook that integrates with the Fusion web application, you must publish it as described in [Section 14.3, "Publishing Your Integrated Excel Workbook."](#) After you have published the Excel workbook, you can deploy it using one of the methods outlined in the "Deploying Fusion Web Applications" chapter of the *Developing Fusion Web Applications with Oracle Application Development Framework*.

The end users that you deploy an integrated Excel workbook to must do the following:

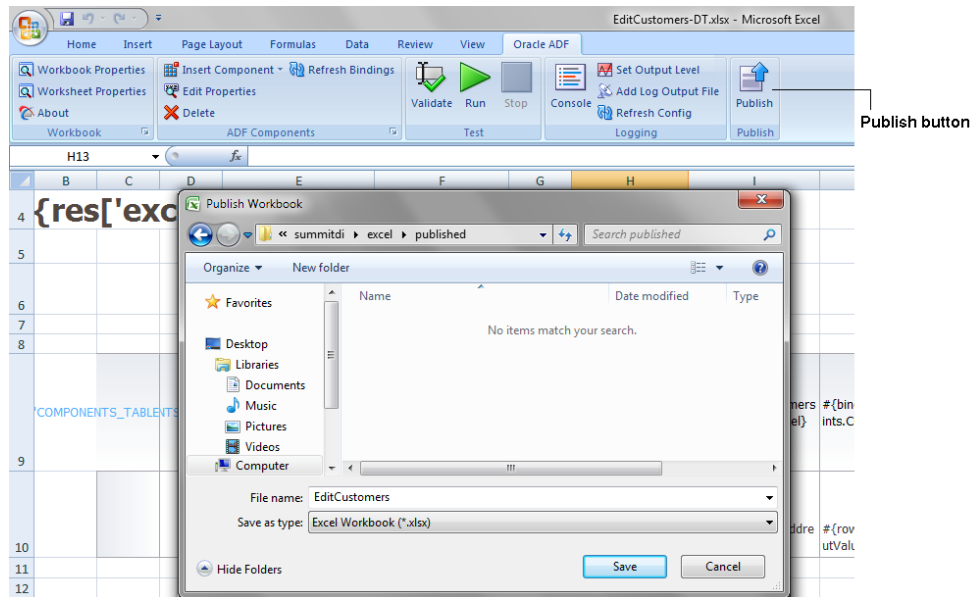
- Set up ADF Desktop Integration on their systems.
Make the ADF Desktop Integration `setup.exe` tool available to end users from, for example, a directory on your network. For more information, see [Section 14.2, "Making ADF Desktop Integration Available to End Users."](#)
- If required, configure the security settings for their Excel application.

When you deploy your integrated Excel workbook with your Fusion web application, you are not required to provide the download URL of workbooks explicitly. The end users can download the integrated Excel workbooks from the Fusion web application's user interface. For more information, see [Section 14.4, "Deploying a Published Workbook with Your Fusion Web Application."](#)

14.1.1 Integrated Excel Workbook Deployment Use Cases and Examples

You use the **Publish** button of the **Oracle ADF** tab to save a copy of the workbook, which is ready for publishing. [Figure 14–1](#) shows the Publish button and the Publish Workbook dialog that opens when you click the **Publish** button to save a copy of the integrated Excel workbook ready to be published and deployed with the Fusion web application.

Figure 14–1 Publish Workbook Dialog



14.1.2 Additional Functionality for Deploying Your Integrated Excel Workbook

After you have published and deployed your integrated Excel workbook, you may find that you need to add additional functionality for your workbook. Following are links to other functionalities that you can use:

- **Passing Parameters:** You can configure a page in your Fusion web application to pass parameter values to an integrated Excel workbook when the end user downloads the workbook from the page. For more information, see [Section 14.5, "Passing Parameter Values from a Fusion Web Application Page to a Workbook."](#)
- **Disconnected Workbooks:** End users can use your integrated Excel workbooks when there is no connectivity with the Fusion web application. For more information, see [Chapter 15, "Using an Integrated Excel Workbook Across Multiple Web Sessions and in Disconnected Mode."](#)

14.2 Making ADF Desktop Integration Available to End Users

End users who want to use the functionality that you configure in an integrated Excel workbook must install the Runtime edition of ADF Desktop Integration. The installation program (`setup.exe`) is located in the `adfdi-excel-runtime-client-installer.zip` file available in `MW_HOME\oracle_common\modules\oracle.adf.desktopintegration_12.1.3` directory, where `MW_HOME` is the Middleware Home directory.

For information about using the installation program, see the "Making the Runtime Edition of ADF Desktop Integration Available to Multiple End Users" section in *Administering Oracle ADF Applications*.

14.3 Publishing Your Integrated Excel Workbook

After you finish configuring the Excel workbook with Oracle ADF functionality, you must publish it. Publishing a workbook makes it available to the end users for whom you configured the integrated Excel workbook.

ADF Desktop Integration also provides you with two methods to publish your workbook. You can publish your integrated Excel workbook directly from Excel, or you can use the publish tool available in JDeveloper to publish the workbook from the command line. The command-line publish tool enables you to use ANT build scripts to publish an integrated Excel workbook from your Fusion web application.

Notes:

- After publishing one or more workbooks, you should restart the Fusion web application in order for those workbooks to be downloaded and opened successfully in Microsoft Excel. If the web application is not restarted, you might get errors, such as the following:

TampercheckErrorException: ADFDI-05537: The integrity of the workbook integration could not be determined.

- Customization-enabled workbooks can only be published to a target location that is under the `public_html` directory (or its sub-directories) of the associated project.
-
-

14.3.1 How to Publish an Integrated Excel Workbook from Excel

You publish a workbook by clicking a button on the **Oracle ADF** tab and specifying values in the dialogs that appear, or by using the command-line publish tool. You can use the command line publish tool to publish a workbook from your Fusion web application.

Before you begin:

It may be helpful to have an understanding about how to publish your integrated Excel workbook. For more information, see [Section 14.3, "Publishing Your Integrated Excel Workbook."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 14.1.2, "Additional Functionality for Deploying Your Integrated Excel Workbook."](#)

To publish a workbook from Excel:

1. Open the integrated Excel workbook.
2. Ensure that the `ApplicationHomeFolder` and `WebPagesFolder` properties in the Edit Workbook Properties dialog are correct. If these properties are not set, ADF Desktop Integration prompts to set them when you publish the integrated Excel workbook.

For more information, see [Section 4.2.2, "How to Configure a New Integrated Excel Workbook."](#)

3. In the **Oracle ADF** tab, click the **Publish** button.
4. Specify the directory and file name for the published workbook in the Publish Workbook dialog that appears, as shown in [Figure 14–1](#). The directory and file name that you specify for the published workbook must be different from the directory and file name for the design time workbook.
5. Click **Save** to save changes.

14.3.2 How to Publish an Integrated Excel Workbook Using the Command Line Publish Tool

The publish tool is run from the command line, and is available in the `MW_HOME\jdeveloper\adfdi\bin\excel\tools\publish` directory as `publish-workbook.exe`. Before you run the publish tool, open the source integrated Excel workbook and ensure that the `ApplicationHomeFolder` and `WebPagesFolder` properties in the Edit Workbook Properties dialog are correct.

Note: You cannot publish a workbook that is already published, or is in runtime mode.

Before you begin:

It may be helpful to have an understanding about how to publish your integrated Excel workbook. For more information, see [Section 14.3, "Publishing Your Integrated Excel Workbook."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 14.1.2, "Additional Functionality for Deploying Your Integrated Excel Workbook."](#)

Now, navigate to `MW_HOME\jdeveloper\adfdi\bin\excel\tools\publish` directory and run the publish tool using the following syntax:

```
publish-workbook -workbook (-w) <source-workbook-path> -out (-o)
<destination-workbook-path>
```

where `source-workbook-path` is the full path of the source workbook, and `destination-workbook-path` is the full path where the published workbook is saved.

For example:

```
publish-workbook -workbook
D:\Application1\Project1\ViewController\src\oracle\sampledemo\excel\workbo
ok-src.xlsx -out D:\Application1\Project1\ViewController\public_
html\excel\published\workbook.xlsx
```

Tip: For more information about the arguments required by the publish tool, run the following command:

```
publish-workbook -help (-h)
```

Notes:

- Always specify the absolute paths of the source and destination workbooks. The publish tool does not support relative paths of the workbooks.
 - The destination workbook cannot have the same name as the source, even if the workbook paths are different.
-
-

After publishing the integrated Excel workbook successfully, the publish tool displays a success message. If there is any error while publishing the workbook, the publish tool aborts the process and the error messages are displayed on the command line console.

If you are using the command line publish tool, note that by default the publish tool logs messages to the command line console at information level.

Using the Publish Tool with ANT

You can create ANT scripts to run the publish tool from JDeveloper when you build your Fusion web application. You can use either of the following methods to run the utility using ANT:

- Generate an ANT build script for the project and add a target to run the workbook command line publish tool
- Generate or create a separate ANT build script for running the workbook command line publish tool

A sample ANT build script (`publish-workbook.xml`) to run the publish tool is available in the `MW_HOME\jdeveloper\adfdi\bin\excel\samples` directory. The sample ANT script demonstrates the invocation of the command-line workbook publishing tool.

14.3.3 What Happens When You Publish an Integrated Excel Workbook

When you click the **Publish** button in design mode, ADF Desktop Integration performs the following actions:

1. Validates the mandatory workbook settings.
2. Updates the client registry. For more information, see [Section 11.3, "Checking the Integrity of an Integrated Excel Workbook's Metadata."](#)
3. Creates the published workbook with the specified file name in the specified directory.

Publish also exports the workbook definition. The published workbook definition XML file is saved at the same location as the design-time copy of the workbook. For more information about workbook definition, see [Section 5.15, "Exporting and Importing Excel Workbook Integration Metadata."](#)

4. Clears the `ApplicationHomeFolder`, `WebAppRoot`, and `WebPagesFolder` properties from the workbook settings of the published workbook.
5. Removes binding expressions that are visible in the worksheet while the workbook is in design mode.
6. Changes the mode of the workbook to runtime mode.

7. Inserts a Publishing Timestamp property into the workbook. This property is visible in the Properties tab of About dialog.

14.4 Deploying a Published Workbook with Your Fusion Web Application

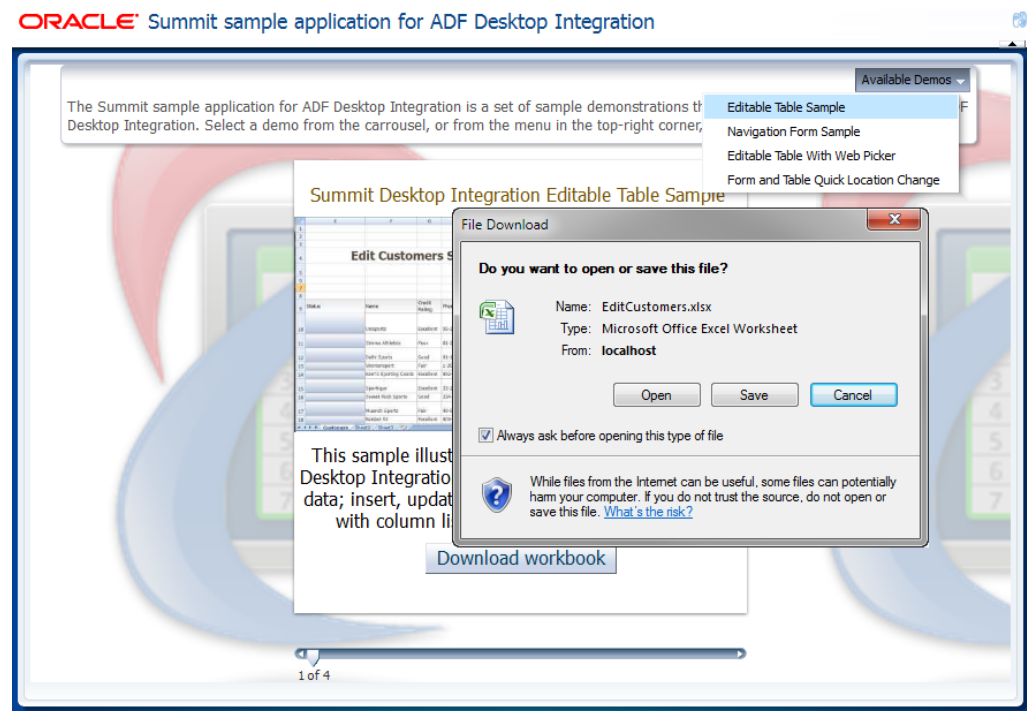
Add the integrated Excel workbook to the JDeveloper project for your Fusion web application if it is not packaged with the other files that constitute your JDeveloper project. This makes sure that the Excel workbooks you integrate with your Fusion web application get deployed when you deploy your finalized Fusion web application. For example, the Summit sample application for ADF Desktop Integration stores the deployed Excel workbooks that it integrates at the following location:

```
<Summit_HOME>\ViewController\public_html\excel
```

where `Summit_HOME` is the installation directory for the Summit sample application for ADF Desktop Integration.

After you decide on a location to store your integrated Excel workbooks, you can configure web pages in your Fusion web application allowing end users to access the integrated Excel workbooks. For example, [Figure 14–2](#) shows Internet Explorer's File Download dialog, which was invoked by clicking the **Available Demos > Editable Table Sample** menu option on the `index.jspx` page of the Summit sample application for ADF Desktop Integration.

Figure 14–2 Invoking an Integrated Excel Workbook from a Fusion Web Application



To enable the functionality illustrated in [Figure 14–2](#), the HTTP filter parameters for your Fusion web application must be configured to recognize Excel workbooks. JDeveloper automatically configures these parameters for you when ADF Desktop Integration is enabled in the Fusion web application. If you want to manually configure the HTTP filter parameters, see [Appendix D, "ADF Desktop Integration Settings in the Web Application Deployment Descriptor."](#)

After you have configured the HTTP filter for your Fusion web application, you configure the web pages that the Fusion web application displays to end users to allow them to invoke Excel workbooks. A basic method of invoking an Excel workbook that you have integrated with a Fusion web application is to provide a hyperlink that invokes the workbook. For example, you could write the following ADF code in a web page:

```
<af:link text="Editable Table Sample"
destination="/excel/EditCustomers.xlsx"/>
```

where `excel` is a subdirectory of the directory specified by the `WebPagesFolder` workbook property and `EditCustomers.xlsx` is the Excel workbook that the end user invokes.

You can provide functionality that allows end users to invoke Excel workbooks from buttons, lists, and ribbon commands. The following list provides some examples:

- **Button**
Display a button on the web page that, when clicked, invokes the integrated Excel workbook.
- **Selection list**
Use the ADF Faces `selectOneChoice` component with a button to invoke an integrated Excel workbook.
- **Menu**
Use the ADF Faces `goMenuItem` component.

The **Available Demos** menu, as illustrated in [Figure 14–2](#), uses the `goMenuItem` component. The following entry appears in the `index.jspx` page of the Summit sample application for ADF Desktop Integration and demonstrates the `goMenuItem` component:

```
<af:goMenuItem text="Editable Table Sample" id="gmi1"
destination="/excel/EditCustomers.xlsx"/>
```

For more information about creating web pages for a Fusion web application, see the *Developing Web User Interfaces with Oracle ADF Faces*.

14.4.1 What Happens at Runtime: Deploying a Published Workbook

When `web.xml` is configured for a Fusion web application that uses ADF Desktop Integration, the following happens:

- The `DIExcelDownloadFilter` filter is defined.
- Filter mappings are defined for `*.xlsx` and `*.xlsm` files.

When the end user makes an http request for a workbook (for example, user clicks a link in a web page from the application), the `DIExcelDownloadFilter` filter embeds the `WebAppRoot` property into the workbook as it gets streamed back as the http response. The `WebAppRoot` property is later used by the ADF Desktop Integration client to connect to the Fusion web application, establish a user session, and send data back and forth.

The `DIExcelDownloadFilter` filter constructs the `WebAppRoot` value from the current `HttpServletRequest` object that is passed in to the `doFilter()` entry point. The filter code calls `HttpServletRequest.getRequestURL()` and gets the "root" portion of the full URL by removing everything after the context path portion (uses `HttpServletRequest.getContextPath()`).

14.5 Passing Parameter Values from a Fusion Web Application Page to a Workbook

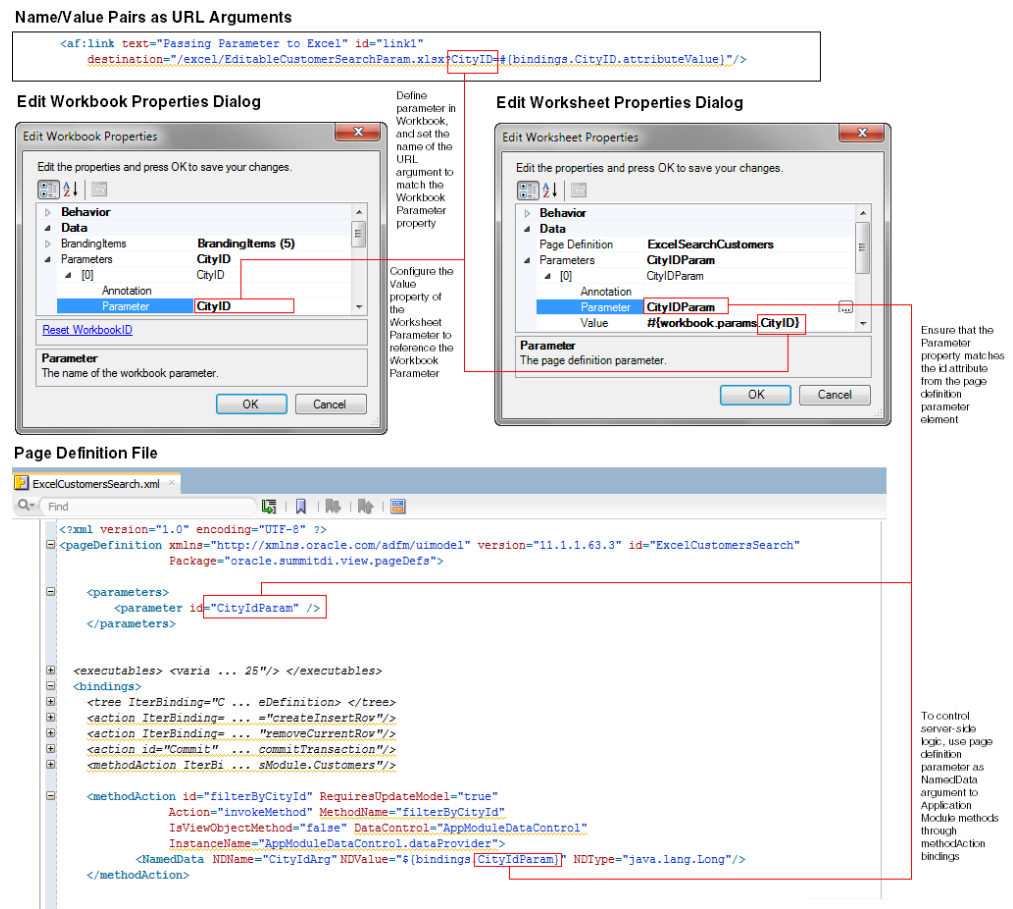
A Fusion web application page can be configured to pass parameter values to an integrated Excel workbook when the end user downloads the workbook from the page. The workbook parameter values can be used as arguments to ADFm methods (among other uses). For example, consider a page that displays a table of customers, but filtered for a particular city specified by the end user. The City ID provided by the end user is used as an argument to a custom method that alters the View Object's query to return customers with a matching City ID. Such a page could also allow the end user to download an integrated workbook that will display the same customer table with data automatically filtered for the city currently selected on the page. As the integrated workbook is downloaded, the City ID is embedded in it as a workbook parameter value.

To pass parameters from the Fusion web application page to the integrated Excel workbook, follow these steps:

1. Verify that the HTTP filter is configured to allow end users to download integrated Excel workbooks from the Fusion web application. By default, JDeveloper configures the HTTP filter with appropriate values when you enable ADF Desktop Integration in a project. To verify the parameter values of the HTTP filter, see [Section D.2, "Configuring the ADF Desktop Integration Excel Download Filter."](#)
2. Use Name/Value pairs as URL arguments in the web page of Fusion web application that allows end user to download the workbook. For more information, see [Section 14.5.1, "How to Configure the Fusion Web Application's Page to Pass Parameters."](#)
3. Define the parameter name in the Edit Workbook Properties dialog and in the Edit Worksheet Properties dialog. For more information, see [Section 14.5.2, "How to Configure Parameters Properties in the Integrated Excel Workbook."](#)
4. Configure the page definition file associated with the worksheet in the integrated Excel workbook by adding <parameter> elements. For more information, see [Section 14.5.3, "How to Configure the Page Definition File for the Worksheet to Receive Parameters."](#)

[Figure 14–3](#) illustrates the steps to pass parameter from the Fusion web application to the integrated Excel workbook.

Figure 14-3 Configuring Workbook and Fusion Web Application to Pass Parameters



14.5.1 How to Configure the Fusion Web Application's Page to Pass Parameters

A component, such as `<af:link>`, can be used to end users to download a published copy of an integrated workbook. The component's destination URL would reference the integrated workbook, and in its query portion, the URL parameter names and values correspond to the workbook's parameter names and values. You also specify the commands on the page that, when invoked, require the Fusion web application to refresh the values referenced by the component and its property values.

For more information about downloading files using action components, see the "How to Use an Action Component to Download Files" section in *Developing Web User Interfaces with Oracle ADF Faces*.

Before you begin:

It may be helpful to have an understanding of how to pass parameter values from the Fusion web application to the integrated Excel workbook. For more information, see [Section 14.5, "Passing Parameter Values from a Fusion Web Application Page to a Workbook."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 14.1.2, "Additional Functionality for Deploying Your Integrated Excel Workbook."](#)

To configure the page in the Fusion web application:

1. In JDeveloper, insert the component or tag (such as `af:link`) into the page from which the end user downloads the integrated Excel workbook.
2. In the Structure window, right-click the component and choose **Go to Properties**.
3. Expand the **Common** section and set values for the properties.

Table 14–1 describes the properties of `af:link` component.

Table 14–1 Properties for `af:link` Tag

Property	Value
Text	<p>Write the text that appears to end users at runtime.</p> <p>For example, write text such as the following to appear at runtime:</p> <p>Download to Excel</p>
Destination	<p>Invoke the expression builder to write an EL expression that specifies the integrated Excel workbook and the values to download as a URL argument:</p> <p>For example, write an EL expression such as the following:</p> <pre>"/excel/EdittableCustomerSearchParam.xlsx?CityID=#{bindings.CityID.attributeValue}"</pre> <p>Note that the runtime URL-encoded value of the entire query string to the right of <code>?</code> must be less than 2048 bytes. If the runtime value exceeds 2048 bytes, the integrated Excel workbook will contain only the URL arguments that fit in 2048 bytes. Subsequent URL arguments do not get included with the integrated Excel workbook. Instead, the Fusion web application writes log entries for these URL arguments identifying them as having not been included.</p> <p>For example, the total size of the result when the following EL expression is evaluated and then URL-encoded must be less than 2048 bytes.</p> <pre>CityID=#{bindings.CityID.attributeValue}&CityCtgType=#{bindings.CityCtgType.attributeValue}.</pre> <p>Also note that if the URL contains more than 256 characters, an exception is raised when the end user downloads and opens the integrated Excel workbook without saving it. To resolve this problem, you must limit your URL length to 256 characters, or instruct the end user to save the workbook before opening it.</p>

4. Optionally, expand the **Behavior** section and specify component IDs for the `partialTriggers` property that, when invoked, update the values of the `af:link` tag and its `Destination` property.

For example, if you have navigation buttons with the IDs `NextButton`, `PreviousButton`, `FirstButton`, and `LastButton`, specify them as follows:

```
:NextButton :PreviousButton :FirstButton :LastButton
```

5. Save the page.

The following example shows the entries that JDeveloper generates in a JSF page using the examples in this procedure:

```
<af:link text="Download to Excel"
destination="/excel/EdittableCustomerSearchParam.xlsx?CityID=#{bindings.CityID.a
ttributeValue}"
partialTriggers=":NextButton :PreviousButton :FirstButton :LastButton"/>
```

14.5.2 How to Configure Parameters Properties in the Integrated Excel Workbook

You configure the workbook `Parameters` property and the worksheet `Parameters` property so that the integrated Excel workbook that the end user downloads from the Fusion web application receives parameter values included in the query string of the workbook download URL.

Before you begin:

It may be helpful to have an understanding of how to pass parameter values from the Fusion web application to the integrated Excel workbook. For more information, see [Section 14.5, "Passing Parameter Values from a Fusion Web Application Page to a Workbook."](#)

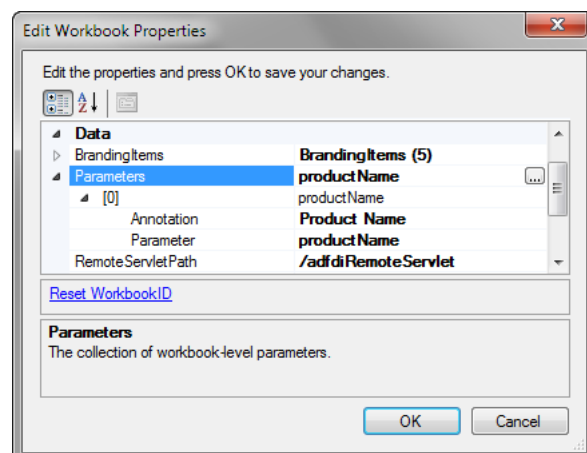
You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 14.1.2, "Additional Functionality for Deploying Your Integrated Excel Workbook."](#)

To configure the workbook `Parameters` property:

1. Open the integrated Excel workbook.
2. In the Workbook group of the Oracle ADF tab, click **Workbook Properties**.
3. Click the browse (...) icon beside the input field for **Parameters** to invoke the Edit Parameters dialog.
4. Click **Add** to add a new workbook initialization parameter and configure its properties as follows:
 - (Optional) In the **Annotation** field, enter a description of the workbook initialization parameter.
 - In the **Parameter** field, specify the name of the URL argument that you configured for the `af:link` tag's `Destination` property as described in [Section 14.5.1, "How to Configure the Fusion Web Application's Page to Pass Parameters."](#)

For example, if you added the `af:link` URL argument as described in [Table 14-1](#), the **Parameter** property value would be `CityID`, as illustrated in [Figure 14-4](#).

Figure 14-4 Workbook Parameters



5. Repeat Step 4 as necessary to add other workbook initialization parameters.
6. Click **OK**.

For more information about the workbook `Parameters` property, see [Table A-21](#).

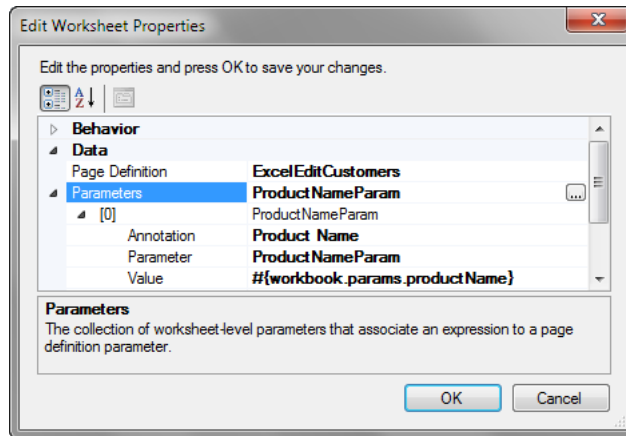
To configure the worksheet `Parameters` property:

1. Open the integrated Excel workbook.
2. In the Workbook group of the Oracle ADF tab, click **Worksheet Properties**.
3. Click the browse (...) icon beside the input field for **Parameters** to invoke the Edit Parameters dialog.
4. Click **Add** to add a new worksheet parameter and configure it as in [Figure 14-5](#):
 - (Optional) In the **Annotation** field, enter a description of the worksheet parameter.
 - In the **Parameter** field, specify a parameter element that you added to the page definition file associated with the worksheet, as described in [Section 14.5.3](#), "How to Configure the Page Definition File for the Worksheet to Receive Parameters."
 - In the **Value** field, write an EL expression that references the value of the Parameter property you specified for the workbook initialization parameter (workbook `Parameters` array). Use the following syntax when writing the EL expression:

```
#{workbook.params.parameter}
```

where *parameter* references the value of the `Parameter` property you specified for the workbook initialization parameter.

Figure 14-5 Worksheet Parameters



5. Repeat Step 4 as necessary to add other workbook initialization parameters.
6. Click **OK**.

For more information about the worksheet `Parameters` property, see [Table A-22](#).

By default, the workbook parameters are not sent every time the workbook connects to the server to request metadata, the end user logs out, or the session expires. If required, you can configure the workbook to send the initialization parameters by configuring the `SendParameters` property.

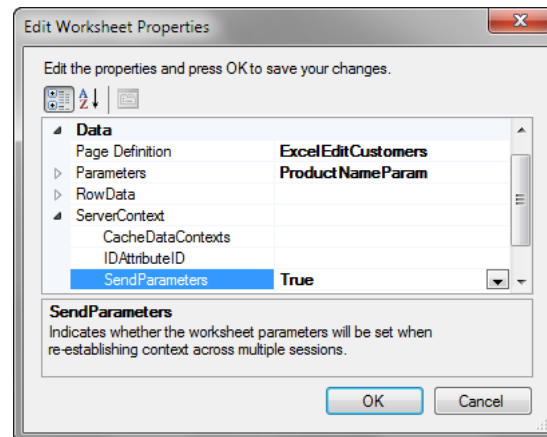
To configure the worksheet `SendParameters` property:

1. Open the integrated Excel workbook.
2. In the Workbook group of the Oracle ADF tab, click **Worksheet Properties**.
3. In the Edit Worksheet Properties dialog, set the value of `SendParameters` as shown in the [Table 14-2](#) and [Figure 14-6](#):

Table 14-2 *SendParameters Property*

Set this property to...	This value...
<code>SendParameters</code>	True to send workbook parameters when the workbook connects to the server to request metadata or data. When set to True, parameters are sent every time when the metadata is requested and the first time when data is requested, during each user session. False is the default value. For more information, see Section 15.2, "Restore Server Data Context Between Sessions."

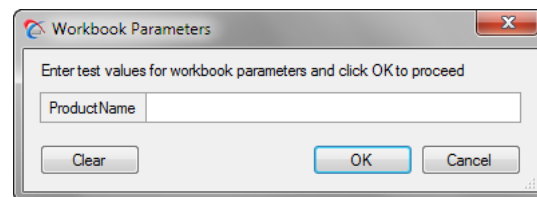
Figure 14-6 *SendParameters Property*



4. Click OK.

When entering the Test mode, the Workbook Parameter dialog prompts you to enter test parameter values. [Figure 14-7](#) shows the Workbook Parameters dialog that accepts test values for the workbook.

Figure 14-7 *Workbook Parameters dialog*



You are not required to enter values for any, or all, parameters. If you enter test parameter values, they are not cleared when you exit the test mode and return to design mode. When you run the integrated Excel workbook again, the workbook parameter values are displayed in the Workbook Parameters dialog from the cache.

The provided test values are stored in the workbook in the same way as the ADF Desktop Integration Excel download filter stores the parameter values. When you

publish the workbook, the test parameter values are cleared before the workbook is published.

Note: In the above example, the `filterByCityId` method can be used as the worksheet's setup action (`SetupActionID`). This will cause the method to be called automatically when the worksheet is initialized at runtime (or whenever a new instance of the worksheet's binding container is created). For more information about `SetupActionID`, see [Section 7.24, "Using Explicit Worksheet Setup Action."](#)

The `filterByCityId` method could also be configured as a part of the action set, such as one for a ribbon command or `Startup` event, depending on the use case. In the case of a ribbon command, its execution will be triggered by the end user. For more information about ribbon commands and `Startup` event, see [Section 8.3.2, "How to Configure a Worksheet Command for the Runtime Ribbon Tab"](#) and [Section 8.2.4, "How to Invoke an Action Set from a Worksheet Event."](#)

Workbook parameter values can be used as arguments for any method exposed by the page definition.

14.5.3 How to Configure the Page Definition File for the Worksheet to Receive Parameters

The page definition file associated with the worksheet in the integrated Excel workbook can be configured as follows:

- Add one or more parameter elements that initialize the worksheet's binding container. The values for these parameters will be supplied from URL arguments, as specified in [Section 14.5.1, "How to Configure the Fusion Web Application's Page to Pass Parameters."](#)

The following example shows a parameter element in a page definition file that is associated with a worksheet in an integrated Excel workbook:

```
<parameters>
  <parameter id="CityIDParam" />
</parameters>
```

- Add a method action binding that invokes an application module method (`filterByCityId`). The application module method takes a single `Long` argument (`CityIdArg`) that references the value of `CityIDParam`. This method is responsible for modifying and executing the view object's query to filter by City ID.

```
<bindings>
<methodAction id="filterByCityId" RequiresUpdateModel="true"
  Action="invokeMethod" MethodName="filterByCityId"
  IsViewObjectMethod="false" DataControl="AppModuleDataControl"
  InstanceName="AppModuleDataControl.dataProvider">
  <NamedData NDName="CityIdArg" NDValue="{bindings.CityIDParam}"
    NDType="java.lang.Long"/>
</methodAction>
. . .
</bindings>
```

For more information about configuring a page definition file, see [Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook."](#)

14.5.4 What Happens at Runtime: How Parameters Are Passed from a Fusion Web Application to the Integrated Excel Workbook

When the end user downloads the integrated Excel workbook from the Fusion web application, the component tag that triggered the download (such as `af:link` tag) is evaluated, the current CityID is captured and included on the URL. The `adfdiExcelDownload` filter embeds the names and values of all the parameters from the URL into the downloaded integrated Excel workbook.

The parameters are set into `BindingContainer` `DCParameters` before the binding container is refreshed. For more information about how worksheet parameters are mapped to binding containers, see [Section 14.5.3, "How to Configure the Page Definition File for the Worksheet to Receive Parameters."](#)

The action set in the worksheet `Startup` event is also executed during initialization. After initialization, the initialization status for each worksheet is recorded when the integrated Excel workbook is saved to disk.

After the integrated Excel workbook has been saved, closed, and reopened, the first-time initialization is skipped for any worksheets that were previously initialized. If workbook parameters were captured when the integrated Excel workbook was first downloaded, and those parameters are required to set up server context, then the `Worksheet.ServerContext.SendParameters` property should be set to `True`. When the `SendParameters` property is `True`, workbook parameters are sent on every request for metadata, and also on the first request for data in each user session.

To reset the initialization state for all worksheets in the workbook, invoke the `ClearAllData` action. For more information about the `ClearAllData` action, see [Table A-20](#).

14.6 Customizing Workbook Integration Metadata at Runtime

ADF Desktop Integration also supports Oracle Metadata Services (MDS) based runtime customization. For more information about MDS, see the "Customizing Applications with MDS" chapter in *Developing Fusion Web Applications with Oracle Application Development Framework*.

Workbook integration metadata defines how ADF Desktop Integration components appear and behave in the workbook, and how the workbook is integrated with its Fusion web application. When the workbook is published, its workbook integration metadata XML file is saved at the same location as the design-time copy of the workbook. For more information about publishing a customization-enabled workbook, see [Section 14.3, "Publishing Your Integrated Excel Workbook."](#)

The workbook integration metadata files for customization-enabled workbooks need to be deployed to MDS metadata repositories so that they can be managed by MDS. For more information about Metadata Repository, see the "Managing the Metadata Repository" chapter in *Administering Oracle Fusion Middleware*.

14.6.1 How to Enable Workbook Customization at Runtime

To enable customization of workbook integration metadata, open the Workbook Properties dialog, and set `CustomizationEnabled` to `True`.

Before you begin:

It may be helpful to have an understanding of customizing workbook integration metadata. For more information, see [Section 14.6, "Customizing Workbook Integration Metadata at Runtime."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 14.1.2, "Additional Functionality for Deploying Your Integrated Excel Workbook."](#)

To enable runtime customization for a workbook:

1. Open the integrated Excel workbook.
2. In the Workbook group of the Oracle ADF tab, click **Workbook Properties**.
3. Set **CustomizationEnabled** to `True`.
4. Click **OK**.
5. Publish the customization-enabled workbook.

14.6.2 What Happens at Runtime: Workbook Integration Metadata is Customized

A customization-enabled workbook obtains its metadata from the server when the workbook is initialized. The integration metadata is managed by MDS on the server end and can be accessed by the application through MDS APIs.

At runtime, the application can provide means for users to customize the workbook integration metadata. When a customization-enabled workbook is being initialized, it requests the server for workbook integration metadata. MDS applies all the customizations based on current customization context and returns the customized metadata to the workbook for its initialization.

For example, an application can provide a web page where users can customize the columns of a table in a customization-enabled workbook. The user can remove certain columns from the table on the web page and then download the customization-enabled workbook and see the change take effect in the workbook.

14.6.3 What You May Need to Know About Customizing Workbook Integration Metadata

Customization-enabled workbooks can only be published to a directory under the `public_html` directory of the associated project. When you deploy your application, make sure that the corresponding workbook integration metadata file can be found by MDS using the metadata path generated when the workbook is published.

Each customization-enabled workbook has its own workbook integration metadata file. When the workbook is published, its workbook integration metadata XML file is saved at the same location as the design-time copy of the workbook. This workbook integration metadata file should be deployed to MDS metadata repositories so that it can be managed by MDS at runtime. In MDS terms, a workbook integration metadata file is a base document and is referenced by MDS using a metadata path. The metadata path is determined when the customization-enabled workbook is published.

For example, if a design-time customization-enabled workbook is published to `<PROJECT_HOME>/public_html/myCompany/myPackage/myWorkbook.xlsx` and its workbook integration metadata file name is `myWorkbook-DT.xlsx-workbook-definition.xml`, then the metadata path for this workbook is `/myCompany/myPackage/myWorkbook-RT.xlsx-workbook-definition.xml`. At runtime, MDS looks for the workbook integration metadata using this metadata path in the repositories configured with the application. The metadata path must be unique across the application.

By default, if no MDS repository is configured for the workbook integration metadata files, MDS will look up the metadata files on the classpath using the metadata path mentioned. To avoid configuring MDS, you may host the workbook integration

metadata files on the classpath of the Fusion web application. The customizations created at runtime can be stored in MDS `default-cust-store`.

Using an Integrated Excel Workbook Across Multiple Web Sessions and in Disconnected Mode

This chapter describes how to configure the integrated Excel workbook so that your use cases work properly across multiple web application sessions.

This chapter includes the following sections:

- [Section 15.1, "About Disconnected Workbooks"](#)
- [Section 15.2, "Restore Server Data Context Between Sessions"](#)
- [Section 15.3, "Caching of Static Information in an Integrated Excel Workbook"](#)
- [Section 15.4, "Caching Lists of Values for Use in Disconnected Mode"](#)

15.1 About Disconnected Workbooks

End users can open an integrated Excel workbook and log on to a Fusion web application from the workbook ribbon command that you configure. The Fusion web application assigns a session to the user. After a connection to the Fusion web application is established and a valid session assigned, end users can download data from the Fusion web application to the workbook. They can then log off from the Fusion web application using the workbook ribbon command or otherwise disconnect from the Fusion web application by, for example, disconnecting from the network that hosts the Fusion web application.

How the Fusion web application terminates the session assigned to the user depends on how the user disconnects from the Fusion web application. If the user logs off from the Fusion web application using a workbook command, the Fusion web application terminates the session immediately. If the user disconnects from the Fusion web application by some other means (for example, closing the workbook), the Fusion web application terminates the session assigned to the user after session timeout expires.

Using integrated Excel workbooks disconnected from the Fusion web application, end users can perform the following actions:

- Modify data downloaded from the Fusion web application
- Insert new data into the appropriate ADF Table component contained in the workbook
- Save changes to data and close and reopen the workbook without having to upload data to the Fusion web application
- Track and update changes in the ADF Table component

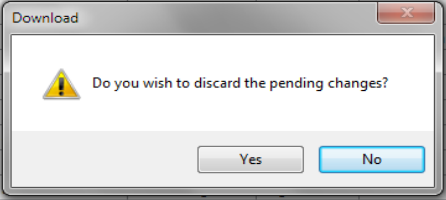
15.1.1 Disconnected Workbooks Use Cases and Examples

When end users open a published integrated Excel workbook, the workbook downloads required data. Then, if they disconnect from the server, they can continue to edit and update the data in the integrated Excel workbook, and save and close it.

The integrated Excel workbook, when opened after saving, detects the updates and prompts the end user to discard the changes, as shown in [Figure 15–1](#). If the end user clicks **No**, the changes would appear in the workbook. If the end user clicks **Yes**, the integrated Excel workbook prompts to connect to the server to download data.

Figure 15–1 Discard Pending Changes Dialog of a Disconnected Workbook

Changed	Flagged	Status	Name	Credit Rati	Phone	Address	City	State	Zip Code
			Unisports	Excellent	55-2066101	72 Via Bahia	Sao Paulo		
			Simms Athletics	Poor	81-20101	Takashi	Osaka		
▲			Delhi Sports	Good					
▲			Womansport	Fair					98101
			Kam's Sporting G	Excellent					
			Sportique	Excellent					
			Sweet Rock Spor	Good					
			Muench Sports	Fair					
			Beisbol Sit	Excellent					
			Futbol Sonora	Excellent					



15.1.2 Additional Functionality for Disconnected Workbooks

After you have validated and tested your integrated Excel workbook in disconnected mode, you may find that you need to add additional functionality for your workbook. Following are links to other functionalities that you can use:

- **Troubleshooting integrated Excel workbook:** You might encounter some problems while developing or deploying an integrated Excel workbook. For more information, see [Appendix C, "Troubleshooting an Integrated Excel Workbook."](#)
- **Installing runtime edition of ADF Desktop Integration:** You must install the runtime edition of ADF Desktop Integration to enable end users to use ADF Desktop Integration and integrated Excel workbooks. For more information, see [Appendix H, "End User Actions."](#)

15.2 Restore Server Data Context Between Sessions

You must configure the page definition file so that the correct view object state is restored if the Fusion web application assigns the end user a new session after one of the following events occurs:

- The end user makes changes to data in a workbook, saves and closes the workbook, reopens the workbook at a later time, and attempts to upload the changes he or she made before saving and closing the workbook.
- The time between invocation of an ADF Table component's `Download` and `Upload` actions (or some other ADF Table component action that contacts the Fusion web application) exceeds the session timeout value specified for a Fusion web application session.

Both the scenarios described in the previous list involve two sessions. The first session is assigned when the end user opens an integrated Excel workbook and logs on to the Fusion web application. The Fusion web application terminates this session when the end user logs off from the Fusion web application or when the session expires. The Fusion web application assigns a second session when the end user reopens the

integrated Excel workbook or invokes an action that interacts with the Fusion web application.

In addition to configuring the page definition file, configure the functionality in an integrated Excel workbook so that pending changes are not lost if the end user logs off from the Fusion web application or a session expires before changes are committed to the Fusion web application. For example, you configure the worksheet `Startup` event to invoke a `CreateInsert` action binding and a worksheet `DownSync` action. You also configure an ADF Button component labeled **Save** to invoke the worksheet `UpSync` action and the `Commit` action binding. If the end user's session ends, no record is saved even if the end user clicks the **Save** button after the Fusion web application assigns a new session. To prevent this scenario occurring, it is better to invoke the `CreateInsert` action binding from the ADF Button component labeled **Save**.

Another example is the behavior of the ADF Table component's `DownloadForInsert` action. If you create a custom method in the Fusion web application that creates temporary records to support the invocation by the ADF Table component of the `DownloadForInsert` action, make sure to remove these temporary records after successful invocation of the `DownloadForInsert` action. For more information about the use of the `DownloadForInsert` action, see [Section 7.5, "Configuring a Worksheet to Download Data as Pending Insert Rows in an ADF Table component."](#)

15.2.1 How to Configure an Integrated Excel Workbook to Restore Server Data Context

You specify the attribute bindings that you want to cache in an integrated Excel workbook between sessions as values for the worksheet's `ServerContext` group of properties. This group of properties also enables you to specify the action binding that uses the cached attribute binding data to restore server-side context when a Fusion web application assigns a new session to the integrated Excel workbook.

Before you can specify values for the `ServerContext` group of properties, the page definition file that is associated with the worksheet must expose the attribute bindings and action bindings for which you want to restore server context. For information about adding attribute bindings and action bindings to a page definition file, see [Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook."](#) For information about the `ServerContext` group of properties, see the entry for `ServerContext` in [Table A-22](#).

Before you begin:

It may be helpful to have an understanding of how to restore server data context. For more information, see [Section 15.2, "Restore Server Data Context Between Sessions."](#)

You may also find it helpful to understand functionality that can be added using other ADF Desktop Integration features. For more information, see [Section 15.1.2, "Additional Functionality for Disconnected Workbooks."](#)

To configure an integrated Excel workbook to restore server data context:

1. Open the integrated Excel workbook.
2. In the Workbook group of the Oracle ADF tab, click **Worksheet Properties**.
3. In the Edit Worksheet Properties dialog, configure values for the `ServerContext` group of properties as described by [Table 15-1](#).

Table 15–1 ServerContext Properties to Restore Server Data Context

For this property...	Enter or select this value...
CacheDataContexts	<p>Add an element to the collection of CacheDataContexts. Configure the element you add as follows:</p> <ul style="list-style-type: none"> ■ RestoreDataContextActionID Specify the action binding (for example, the Execute action binding) that connects to the Fusion web application to restore the data specified by CachedServerContexts. ■ CachedServerContexts An array that identifies the attribute binding values to cache and set before the action binding specified by RestoreDataContextActionID is invoked. Each element in the array (CachedServerContext) supports the CachedAttributeID and RestoredAttributeID properties. <p>For more information about the CacheDataContexts property and its subproperties, see Section A–22, "Worksheet Properties."</p>
IDAttributeID	<p>Specify the attribute binding that uniquely identifies the row displayed in the current worksheet. At runtime, the value that this property references is used to determine if the server data context has been correctly restored.</p> <p>For more information about this property and its subproperties, see Section A–22, "Worksheet Properties."</p>

If your integrated Excel workbook uses parameters and you have deployed it by downloading it from your Fusion web application, see [Section 14.5.2, "How to Configure Parameters Properties in the Integrated Excel Workbook."](#)

4. Click OK.

Note: For integrated Excel workbooks that use Parameters and <invokeAction> executable, you may not need to configure RestoreDataContextActionID and CachedServerContexts, if Parameters and <invokeAction> can restore server data context when a new session is created.

15.2.2 What Happens at Runtime: How the Integrated Excel Workbook Restores Server Data Context

During the session that is assigned the initial session (for example, session ID 1), the worksheet caches data using the ServerContext group of properties. In a later session with a different session ID (for example, session ID 2), where the ADF Table component's Upload action is invoked, the data cached in the ServerContext group of properties is sent to the Fusion web application.

15.3 Caching of Static Information in an Integrated Excel Workbook

Certain types of relatively static data are cached in the integrated Excel workbook to allow end users to use the workbook while disconnected from the Fusion web application. [Table 15–2](#) describes the types of data that an integrated Excel workbook caches. It also describes when the integrated Excel workbook refreshes the data.

Table 15–2 Types of Data an Integrated Excel Workbook Caches

This type of data...	Is cached when...	And refreshed when...
Page definition configuration that is not runtime specific such as control binding types, IDs, and labels.	An integrated Excel worksheet bound to a page definition file is activated and no cache of the page definition file's configuration exists.	The page definition configuration is not refreshed unless you download a new copy of the integrated Excel workbook or invoke the workbook actions <code>ClearAllData</code> and <code>EditOptions</code> described in Table A–20 .
ADF List of Values component list items	The ADF List of Values component first downloads the list items from the Fusion web application.	The values of the list items hosted by the Fusion web application differ from those cached by the integrated Excel workbook. The cached list items are refreshed only once per workbook session and only if a workbook session exists. Invoking the workbook actions <code>ClearAllData</code> and <code>EditOptions</code> described in Table A–20 also clears cached list items.
Resource bundle strings	The integrated Excel workbook is first initialized. A workbook is initialized when it is opened for the first time after conversion, or after <code>ClearAllData</code> is invoked.	The cache of resource bundle strings is not refreshed unless you download a new copy of the integrated Excel workbook or invoke the workbook actions <code>ClearAllData</code> and <code>EditOptions</code> described in Table A–20 .

15.4 Caching Lists of Values for Use in Disconnected Mode

ADF Desktop Integration caches the values referenced by the ADF List of Values and the `TreeNodeList` subcomponents that you use to create lists of values and dependent lists of values so that these components do not send a request to the Fusion web application when the end user selects a value at runtime. For more information about using these components to create lists of values, see the following sections:

- [Section 6.6, "Inserting an ADF List of Values Component"](#)
- [Section 7.14, "Creating a List of Values in an ADF Table Component Column"](#)
- [Section 8.8, "Creating Dependent Lists of Values in an Integrated Excel Workbook"](#)

ADF Desktop Integration caches up to two hundred and fifty values for each component. If a component references a list of values with more than two hundred and fifty values, ADF Desktop Integration caches the first two hundred and fifty values and writes a warning message to the client-side log file for subsequent values. Consider configuring your integrated Excel workbook to invoke a pick dialog from a page in your Fusion web application where a list of values references more than two hundred and fifty values. For more information about client-side log files, see [Section C.3, "Generating Log Files for an Integrated Excel Workbook."](#) For more information about invoking a pick dialog from a Fusion web application page, see

[Section 8.4, "Displaying Web Pages from a Fusion Web Application"](#) and [Section 8.5, "Adding a Custom Popup Picker Dialog to an ADF Table Column."](#)

Cached list of values in an integrated Excel workbook get refreshed once per workbook session. This refresh occurs after the user reestablishes a web session with the Fusion web application and if the values referenced by the Fusion web application have changed since the integrated Excel workbook last cached the list of values.

The upload of a selected value from a list of values causes the upload to fail if the selected value no longer exists in the Fusion web application. This may occur if, for example, one end user deletes the value in the Fusion web application while another end user modifies the selected value in the cached list of values of an integrated Excel workbook and attempts to upload the modified value to the Fusion web application. For more information about handling data conflict, see [Section 12.7, "Handling Data Conflicts When Uploading Data from a Workbook."](#)

Note that if you change the Fusion web application configuration after you have deployed the Fusion web application and the end users have started using the published integrated Excel workbooks, you must inform the end users to download a fresh copy of the integrated Excel workbook, or run the `ClearAllData` command. For more information about the `ClearAllData` action, see [Table A-20](#)

The changes in your Fusion web application might include changing the definitions of the list bindings associated with the ADF List of Values and `TreeNodeList` subcomponents exposed in the worksheet. Changing list binding configuration can cause unexpected exceptions in workbooks that have been downloaded and run prior to the change.

A

ADF Desktop Integration Component Properties and Actions

This appendix lists and describes the properties of ADF Desktop Integration components. It also describes the actions that certain components (such as ADF Input Text, ADF Output Text, ADF List of Values, ADF Button, ADF Table, Workbook, and Worksheet) expose.

This appendix includes the following sections:

- Section A.1, "Frequently Used Properties in the ADF Desktop Integration"
- Section A.2, "ADF Input Text Component Properties"
- Section A.3, "ADF Output Text Component Properties"
- Section A.4, "ADF Output Text (Worksheet Errors)"
- Section A.5, "ADF Output Text (Table Errors)"
- Section A.6, "ADF Label Component Properties"
- Section A.7, "ADF List of Values Component Properties"
- Section A.8, "ADF Image Component Properties"
- Section A.9, "TreeNodeList Subcomponent Properties"
- Section A.10, "ModelDrivenColumnComponent Subcomponent Properties"
- Section A.11, "ADF Button Component Properties"
- Section A.12, "ADF Table Component Properties and Actions"
- Section A.13, "ADF Read-only Table Component Properties and Actions"
- Section A.14, "Action Set Properties"
- Section A.15, "Workbook Actions and Properties"
- Section A.16, "Worksheet Actions and Properties"

A.1 Frequently Used Properties in the ADF Desktop Integration

Table A-1 lists alphabetically properties in ADF Desktop Integration that many components reference.

Table A–1 *Frequently Used Properties in ADF Desktop Integration*

Name	Type	EL	Description
ActionSet		N	For information about action sets, see Section A.14, "Action Set Properties."
Annotation	String	N	Use this field to enter a comment about the component's use in the worksheet. Comments you enter have no effect on the behavior of the workbook. They are the equivalent of code comments.
ComponentID	String	N	ADF Desktop Integration generates this string to uniquely identify each instance of an ADF component in an integrated Excel workbook.
Label	String	Y	Specify an EL expression that is evaluated at runtime. For information about EL expressions in ADF Desktop Integration, see Appendix B, "ADF Desktop Integration EL Expressions." For information about using labels, see Section 9.4, "Using Labels in an Integrated Excel Workbook."
Position		N	This property defines the upper-left corner of the Oracle ADF component in the integrated Excel workbook.
ReadOnly	Boolean	Y	<p>Set this property to <code>TRUE</code> so that ADF Desktop Integration ignores changes a user makes to a cell that references a component which uses this property. This property is independent of Excel's workbook and worksheet protection functionality. Setting <code>ReadOnly</code> to <code>TRUE</code> does not prevent a user from modifying a cell. When <code>TRUE</code>, the behavior for cells that reference Oracle ADF components is as follows:</p> <ul style="list-style-type: none"> ■ ADF Desktop Integration overwrites changes without warning when a worksheet is refreshed. ■ No changes are sent to the Fusion web application when the integrated Excel workbook is synchronized with the Fusion web application. <p>To avoid end user confusion, apply styles to the cells where you set <code>ReadOnly</code> to <code>TRUE</code> that provide a visual clue to users that they cannot modify the cell's contents. For information about applying styles, see Section 9.2, "Working with Styles."</p> <p>You can also use the Worksheet Protection feature of ADF Desktop Integration to prevent editing of locked cells at runtime. For more information, see Section 9.7, "Using Worksheet Protection."</p> <p>Setting the <code>ReadOnly</code> property to <code>True</code> for an ADF List of Values component, a <code>TreeNodeList</code> subcomponent, or a <code>ModelDrivenColumn</code> component which renders a list binding may confuse end users. To avoid any confusion, consider using an ADF Output Text component or subcomponent.</p>

Table A-1 (Cont.) Frequently Used Properties in ADF Desktop Integration

Name	Type	EL	Description
StyleName	String	Y	Specifies the style in the current Excel workbook to apply when the Oracle ADF component is rendered. For more information, see Section 9.2, "Working with Styles."
Tooltip	String	Y	Specify the hint message about the content or function of the ADF form component, or table column, to appear when the mouse hovers the component, or the column. For more information, see Section 6.9, "Displaying Tooltips in ADF Desktop Integration Form-Type Components" and Section 7.18, "Displaying Tooltips in ADF Desktop Integration Table-Type Components."
Value	Varies	Y	This property references an EL expression that is evaluated after the invocation of the ADF Table component's RowDownSync action or a worksheet's DownSync action. The resulting value is typically the primary value seen in the selected component.

Many label-type properties are optional and default to empty. At runtime, if the value of the property is empty, ADF Desktop Integration provides a default, localized value. If you want the value of the property to appear as empty, set its value to a single space character, or provide an EL expression that evaluates to an empty string.

A.2 ADF Input Text Component Properties

[Table A-2](#) lists alphabetically the properties of the ADF Input Text component.

Table A-2 ADF Input Text Component Properties

Name	Description
Annotation	For information about this property, see Table A-1 .
ComponentID	For information about this property, see Table A-1 .
InputText.DoubleClickActionSet	Specifies the action set invoked when a user double-clicks the cell. For information about action sets, see Section A.14, "Action Set Properties."
InputText.ReadOnly	For information about this property, see Table A-1 .
InputText.Value	For information about this property, see Table A-1 .
Position	For information about this property, see Table A-1 .
StyleName	For information about this property, see Table A-1 .
Tooltip	For information about this property, see Table A-1 .

A.3 ADF Output Text Component Properties

[Table A-3](#) lists alphabetically the properties of the ADF Output Text component.

Table A-3 ADF Output Text Component Properties

Name	Description
Annotation	For information about this property, see Table A-1 .
ComponentID	For information about this property, see Table A-1 .
OutputText.DoubleClickActionSet	Specifies the action set invoked when a user double-clicks the cell. For information about action sets, see Section A.14, "Action Set Properties."
OutputText.Value	For information about this property, see Table A-1 .
Position	For information about this property, see Table A-1 .
StyleName	For information about this property, see Table A-1 .
Tooltip	For information about this property, see Table A-1 .

A.4 ADF Output Text (Worksheet Errors)

[Table A-4](#) lists alphabetically the properties of the ADF Output Text (Worksheet Errors) component.

Table A-4 ADF Output Text (Worksheet Errors) Component Properties

Name	Description
Annotation	For information about this property, see Table A-1 .
ComponentID	For information about this property, see Table A-1 .
OutputText.DoubleClickActionSet	Specifies the action set invoked when a user double-clicks the cell. For information about action sets, see Section A.14, "Action Set Properties."
OutputText.Value	For information about this property, see Table A-1 .
Position	For information about this property, see Table A-1 .
StyleName	For information about this property, see Table A-1 .
Tooltip	For information about this property, see Table A-1 .

A.5 ADF Output Text (Table Errors)

[Table A-5](#) lists alphabetically the properties of the ADF Output Text (Table Errors) component.

Table A-5 ADF Output Text (Table Errors) Component Properties

Name	Description
Annotation	For information about this property, see Table A-1 .
ComponentID	For information about this property, see Table A-1 .
OutputText.DoubleClickActionSet	Specifies the action set invoked when a user double-clicks the cell. For information about action sets, see Section A.14, "Action Set Properties."
OutputText.Value	For information about this property, see Table A-1 .
Position	For information about this property, see Table A-1 .
StyleName	For information about this property, see Table A-1 .
Tooltip	For information about this property, see Table A-1 .

A.6 ADF Label Component Properties

The ADF Label component displays a static string value at runtime. ADF Desktop Integration generates the value when the EL expression that the `Label` property references is evaluated. For information about using labels, see [Section 9.4, "Using Labels in an Integrated Excel Workbook."](#)

[Table A-6](#) lists alphabetically the properties that the ADF Label component references.

Table A-6 ADF Label Component Properties

Name	Description
Annotation	For information about this property, see Table A-1 .
ComponentID	For information about this property, see Table A-1 .
Label	For information about this property, see Table A-1 .
Position	For information about this property, see Table A-1 .
StyleName	For information about this property, see Table A-1 .
Tooltip	For information about this property, see Table A-1 .

A.7 ADF List of Values Component Properties

[Table A-7](#) lists the properties of the ADF List of Values component. For information about creating an ADF List of Values component, see [Section 6.6, "Inserting an ADF List of Values Component."](#)

Table A-7 ADF List of Values Component Properties

Name	Type	EL	Description
Annotation			For information about this property, see Table A-1 .
ComponentID			For information about this property, see Table A-1 .
ListOfValues.DependsOnListID	List binding	N	Select the list binding whose value at runtime determines the choices available in the dependent list of values at runtime. The list binding that you select can be a model-driven list. For more information about dependent list of values, see Section 8.8, "Creating Dependent Lists of Values in an Integrated Excel Workbook."
ListOfValues.ListID	List binding	N	Select the list binding that defines the values available in the list of values. The list binding that you select can be a model-driven list.
ListOfValues.ReadOnly	Boolean	N	For information about this property, see Table A-1 .
Position			For information about this property, see Table A-1 .
StyleName			For information about this property, see Table A-1 .
Tooltip	String	Y	For information about this property, see Table A-1 .

A.8 ADF Image Component Properties

The ADF Image component displays an image at runtime. For more information about adding an ADF Image component, see [Section 6.7, "Inserting an ADF Image Component."](#)

Table A-8 ADF Label Component Properties

Name	Description
Source	The absolute, or relative, URL of the image file.
ShortDesc	Enter the EL expression that resolves to the alternate text of the image component.
Annotation	For information about this property, see Table A-1 .
ComponentID	For information about this property, see Table A-1 .
Position	For information about this property, see Table A-1 .

A.9 TreeNodeList Subcomponent Properties

The `TreeNodeList` is an ADF Table subcomponent that renders dropdown menus in columns of the ADF Table component at runtime. It provides the same functionality to end users as the ADF List of Values component.

The `TreeNodeList` subcomponent does not appear in the components palette of the ADF Desktop Integration task pane. Instead, you configure properties for this subcomponent when you specify `TreeNodeList` as the subcomponent to invoke for the ADF Table component's `UpdateComponent` or `InsertComponent` table column properties described in [Section A.12.2, "ADF Table Component Column Properties."](#)

[Table A-9](#) describes the properties that you configure for the `TreeNodeList` subcomponent.

Table A-9 TreeNodeList Subcomponent Properties

Name	Type	EL	Description
<code>DependsOnList</code>	Tree binding attribute or List binding	Y	Specify the tree binding attribute or list binding that serves as the parent list of values in a dependent list of values. Note that the tree binding attribute you specify must be associated with a model-driven list. For more information about dependent list of values, see Section 8.8, "Creating Dependent Lists of Values in an Integrated Excel Workbook."
<code>List</code>	Tree binding attribute	Y	Specify the tree binding attribute associated with a model-driven list that defines the values available in the runtime dropdown menu to appear in the ADF Table component's column.
<code>ReadOnly</code>	Boolean	Y	For information about this property, see Table A-1 .

A.10 ModelDrivenColumnComponent Subcomponent Properties

The `ModelDrivenColumnComponent` subcomponent, like the `TreeNodeList` subcomponent, does not appear in the components palette of the ADF Desktop Integration task pane. Instead, you configure properties for this subcomponent when you specify `ModelDrivenColumnComponent` as the subcomponent to invoke for the ADF Table component's `UpdateComponent` or `InsertComponent` table column properties described in [Section A.12.2, "ADF Table Component Column Properties."](#)

[Table A-10](#) describes the properties that you configure for the `ModelDrivenColumnComponent` subcomponent.

Table A–10 ModelDrivenColumnComponent Subcomponent Properties

Name	Type	EL	Description
DoubleClickActionSet			Specifies the action set invoked when a user double-clicks the cell. For information about action sets, see Section A.14, "Action Set Properties."
ReadOnly	Boolean	Y	For information about this property, see Table A–1 .
Value	Varies	Y	For information about this property, see Table A–1 .

A.11 ADF Button Component Properties

[Table A–11](#) lists alphabetically the properties of the ADF Button component.

Table A–11 ADF Button Component Properties

Name	Description
Annotation	For information about this property, see Table A–1 .
ClickActionSet	Specify the action set to invoke when a user clicks the button. For information about action sets, see Section A.14, "Action Set Properties."
ComponentID	For information about this property, see Table A–1 .
Label	For information about this property, see Table A–1 .
LowerRightCorner	This property is an Excel cell reference. Used with <code>Position</code> , it specifies the area that the button occupies on the Excel worksheet.
Position	For information about this property, see Table A–1 .

A.12 ADF Table Component Properties and Actions

The ADF Table component uses the properties and component actions listed here.

A.12.1 ADF Table Component Properties

[Table A–12](#) lists alphabetically the properties the ADF Table component uses.

Table A–12 ADF Table Component Properties

Name	Type	EL	Description
Annotation			For information about this property, see Table A–1 .
BatchOptions			This group of properties enables you to configure batch options for the ADF Table component. For more information about how you use these properties, see Section 7.11, "Batch Processing in an ADF Table Component."
BatchOptions.BatchSize	Integer	N	Specifies how many rows to process before an ADF Table component action (<code>Upload</code> or <code>DeleteFlaggedRows</code>) invokes <code>CommitBatchActionID</code> . Any value other than a positive integer results in all rows being processed in a single batch. The default value is 100 rows. A value for this property is required.
BatchOptions.CommitBatchActionID	Action binding	N	Specify an action binding to invoke when the number of rows specified by <code>BatchSize</code> have been processed. The action binding is expected to be a commit-type action.

Table A–12 (Cont.) ADF Table Component Properties

Name	Type	EL	Description
BatchOptions.LimitBatchSize	Boolean	N	Set this property to <code>TRUE</code> to process rows in batches where each batch contains the number of rows specified by <code>BatchSize</code> . If set to <code>FALSE</code> , all rows are processed in a single batch.
BatchOptions.StartBatchActionID	Action binding	N	Specify an action binding to invoke at the beginning of each batch. For example, this property might be used for an operation like "start transaction", if required by a particular database. A value for this property is optional.
Columns			An array of columns. For information about the properties that each column in the array supports, see Section A.12.2, "ADF Table Component Column Properties."
ComponentID			For information about this property, see Table A–1 .
Position			For information about this property, see Table A–1 .
ResizeColumnsMode			Controls whether and how the columns in the entire table are resized. For more information, see Section 7.17, "Configuring Column Widths in an ADF Table Component."
RowActions			This group of properties allows you specify which actions are enabled and can be invoked.
RowActions.DeleteRowActionID	Action binding	N	Specify an action binding to invoke for each row flagged for deletion. A value for this property is optional.
RowActions.DeleteRowEnabled	Boolean	N	Set to <code>TRUE</code> to allow a user to delete existing rows. <code>FALSE</code> is the default value. A value for this property is required.
RowActions.FailureActionID	Action binding	N	Specify an action binding to invoke if failures occur during the processing of rows. A value for this property is optional.
RowActions.InsertAfterRowActionID	Action binding	N	Specify an action binding to invoke for each row inserted using the ADF Table component <code>Upload</code> action. The action binding is invoked after the attributes are set. Use of this property is suitable with a custom action where a variable iterator is employed along with the main iterator. A value for this property is optional.
RowActions.InsertBeforeRowActionID	Action binding	N	Specify an action binding to invoke for each row inserted using the <code>Upload</code> component action. The action binding specified is invoked before the attributes are set. A value for this property is optional.

Table A-12 (Cont.) ADF Table Component Properties

Name	Type	EL	Description
<code>RowActions.InsertRowEnabled</code>	Boolean	N	<p>Set to <code>TRUE</code> to allow the end user insert new rows in the ADF Table component. <code>FALSE</code> is the default value.</p> <p>If you set this property to <code>TRUE</code>, you must specify values for one or both of the following properties:</p> <ul style="list-style-type: none"> ■ <code>RowActions.InsertAfterRowActionID</code> ■ <code>RowActions.InsertBeforeRowActionID</code> <p>Which property (<code>InsertAfterRowActionID</code> or <code>InsertBeforeRowActionID</code>) you specify a value for depends on how your Fusion web application creates new rows. Typically, a Fusion web application uses the <code>CreateInsert</code> action binding to create and insert a new row. In this scenario, you specify the <code>CreateInsert</code> action binding as the value for <code>InsertBeforeRowActionID</code>.</p> <p>For more information about inserting rows in an ADF Table component, see Section 7.7, "Configuring an ADF Table Component to Insert Data."</p>
<code>RowActions.InsertRowsAfterUploadEnabled</code>	Boolean	N	<p>Set to <code>TRUE</code> to allow the end user to reinsert changed rows regardless of whether they have been previously uploaded. <code>FALSE</code> is the default value.</p> <p>The property is ignored if <code>InsertRowEnabled</code> is set to <code>FALSE</code>.</p>
<code>RowActions.UpdateRowActionID</code>	Action binding	N	<p>Specify an action binding to invoke for each row updated.</p> <p>A value for this property is optional.</p>
<code>RowActions.UpdateRowEnabled</code>	Boolean	N	<p>Set to <code>TRUE</code> to allow a user update an existing row. <code>TRUE</code> is the default value.</p> <p>A value for this property is required.</p>
<code>RowData</code>			<p>Set values for the <code>CachedAttributes</code> property when you want to cache data in an integrated Excel workbook across multiple sessions with the Fusion web application.</p> <p>Set a value for the <code>ChangeIndicatorAttributeID</code> property to determine whether a row has been modified by another user since you downloaded it from the Fusion web application.</p>

Table A-12 (Cont.) ADF Table Component Properties

Name	Type	EL	Description
<code>RowData.CachedAttributes</code>	Array	N	<p>Specify values for the properties in this array to determine the attributes for which data is cached. Each <code>CachedTreeAttribute</code> element in this array supports the following properties:</p> <ul style="list-style-type: none"> ▪ Value Select the tree binding attribute for which data is to be cached. ▪ Annotation For more information about this property, see Table A-1. <p>Do not configure a component (for example, an ADF Table component's column or an ADF Input Text component) so the end user can view or edit an attribute binding that you have also specified for an element in the <code>RowData.CachedAttributes</code> array. The <code>RowData.CachedAttributes</code> array caches the values retrieved by the worksheet <code>DownSync</code> action. The worksheet <code>UpSync</code> action sends the values cached by the <code>RowData.CachedAttributes</code> array to the Fusion web application. This may override edits the end user makes to an attribute binding exposed through a component in the worksheet.</p> <p>For information about using the <code>RowData.CachedAttributes</code> array to cache data in an ADF Table component, see Section 8.5, "Adding a Custom Popup Picker Dialog to an ADF Table Column."</p>
<code>RowData.ChangeIndicatorAttributeID</code>	Attribute Binding	Y	<p>Specify an EL expression that evaluates to a row-specific tree attribute binding value. The attribute value is used to determine if a row has been modified by another user since the row was last downloaded to your integrated Excel workbook.</p> <p>For more information, see Section 12.7, "Handling Data Conflicts When Uploading Data from a Workbook."</p>
<code>RowLimit</code>			<p>This group of properties allows you configure the number of rows that the ADF Table component or ADF Read-only Table component download and display.</p> <p>For more information, see Section 7.20, "Limiting the Number of Rows Your Table-Type Component Downloads."</p>
<code>RowLimit.Enabled</code>	Boolean	N	<p>Set to <code>TRUE</code> to limit the number of rows downloaded to the value specified by <code>RowLimit.MaxRows</code>. <code>TRUE</code> is the default value.</p> <p>A value for this property is required.</p>

Table A-12 (Cont.) ADF Table Component Properties

Name	Type	EL	Description
RowLimit.MaxRows	Integer	Y	<p>Specify an EL expression that evaluates to the maximum number of rows to download. The component evaluates the EL expression when it invokes its <code>Download</code> action. The default value is 500. If <code>MaxRows</code> is not a positive integer, the component attempts to download as many rows as possible. An invalid expression such as "ABC" is interpreted as -1 (negative integer). As a result, the component attempts to download as many rows as possible.</p> <p>Note that setting the value of <code>MaxRows</code> to 0 results in a message where the user is asked if they want to download the first 0 rows. To avoid this, set <code>MaxRows</code> to a positive integer other than 0.</p>

Table A–12 (Cont.) ADF Table Component Properties

Name	Type	EL	Description
RowLimit.WarningMessage	String	Y	<p>(Optional) Write an EL expression to generate a message to display to the end user if the number of rows available to download exceeds the number specified by <code>RowLimit.MaxRows</code>. This expression is evaluated each time the Table's <code>Download</code> action is invoked. The maximum number of rows that a Excel 2007, or a higher version, worksheet can contain is approximately 1 million. If this property is left blank, ADF Desktop Integration displays a message similar to "Too many rows available. Do you want to download the first {0} rows?" that is translated for the current culture settings.</p> <p>You can specify a string key from a custom resource bundle to use, instead of the default value. If desired, you may supply a custom message to replace the default one. Any custom message must contain {0}. {0} will be replaced by the <code>MaxRows</code> value.</p> <p>For more information about resource bundles, see Section 10.2, "Using Resource Bundles in an Integrated Excel Workbook."</p>
TreeID	Binding	N	<p>Specify a tree binding from the current worksheet's page definition file. You must specify a value for this property so that row downloads and uploads function properly. For more information about the page definition requirements for an integrated Excel workbook, see Table 4–1.</p>
UniqueAttribute	Attribute binding	Y	<p>Specify an EL expression that evaluates to a unique row-specific tree attribute binding value. The value of this attribute is cached in the integrated Excel workbook during the ADF Table component's <code>Download</code> action. ADF Desktop Integration uses this value to ensure that the tree binding's iterator is positioned correctly before setting or getting data for a given ADF Table component row.</p> <p>Note that this value is required only when the underlying tree binding iterator does not expose a <code>rowKey</code>.</p> <p>This value is optional when:</p> <ul style="list-style-type: none"> ■ The tree binding iterator exposes a <code>rowKey</code>, in which case the <code>rowKey</code> value is used for positioning OR ■ The ADF Table component is configured to be insert-only (<code>RowActions.InsertRowEnabled</code> is set to <code>True</code> and <code>RowActions.UpdateRowEnabled</code> is set <code>False</code>)

A.12.2 ADF Table Component Column Properties

[Table A–13](#) describes the properties that a column in the `TableColumn` array can use.

Table A-13 ADF Table Component Column Properties

Name	Type	EL	Description
Annotation			For information about this property, see Table A-1 .
CellStyleName	String	Y	Write an EL expression that resolves to an Excel style name that is applied to each cell in the column.
Tooltip	String	Y	For information about this property, see Table A-1 .
DynamicColumn	Boolean	N	Set to <code>True</code> to make a column dynamic. <code>False</code> is the default value. For more information about dynamic columns, see Section 7.16, "Adding a Dynamic Column to Your ADF Table Component."
HeaderLabel	String	Y	Write an EL expression that, when evaluated at runtime, displays a label in the column header.
HeaderStyleName	String	Y	Write an EL expression that resolves to an Excel style name that is applied to each cell in the column header.
ID	String	N	<p>Assign a name to the column to identify it and its purpose. The value that you assign for this property has no functional impact. However, you must specify a value and the value that you specify must be unique within the list of columns. It serves to help you keep track of columns in the ADF Table component. The following IDs are reserved to the three default columns in the ADF Table component:</p> <ul style="list-style-type: none"> ■ <code>_ADF_ChangedColumn</code> ■ <code>_ADF_FlagColumn</code> ■ <code>_ADF_StatusColumn</code> <p>For more information about these columns, see Section 7.12, "Special Columns in the ADF Table Component."</p>
InsertComponent	ADF component	N	<p>Specifies the properties of the component that represents the binding for insert operations. This component can be one of the following:</p> <ul style="list-style-type: none"> ■ Input Text component For information about the properties that this component supports, see Section A.2, "ADF Input Text Component Properties." ■ Output Text component For information about the properties that this component supports, see Section A.3, "ADF Output Text Component Properties." ■ TreeNodeList component For information about the properties that this component supports, see Section A.9, "TreeNodeList Subcomponent Properties." ■ ModelDrivenColumnComponent For information about the properties that this component supports, see Section A.10, "ModelDrivenColumnComponent Subcomponent Properties." <p>When <code>InsertUsesUpdate</code> is set to <code>True</code>, the ADF Table component ignores the value of the <code>InsertComponent</code> property.</p>

Table A–13 (Cont.) ADF Table Component Column Properties

Name	Type	EL	Description
InsertUsesUpdate	Boolean	N	Set to <code>True</code> if insert and update operations use the same component type. When <code>True</code> , the ADF Table component ignores the values of the <code>InsertComponent</code> property and reads the value of the <code>UpdateComponent</code> property. The default value is <code>True</code> .
ResizeMode			Specifies how ADF Desktop Integration changes the column width at runtime. For more information, see Section 7.17, "Configuring Column Widths in an ADF Table Component."
UpdateComponent	ADF component	N	Specifies the properties of the component that represents the binding for update and download operations. This component can be one of the following: <ul style="list-style-type: none"> ■ Input Text component For information about the properties that this component supports, see Section A.2, "ADF Input Text Component Properties." ■ Output Text component For information about the properties that this component supports, see Section A.3, "ADF Output Text Component Properties." ■ TreeNodeList component For information about the properties that this component supports, see Section A.9, "TreeNodeList Subcomponent Properties." ■ ModelDrivenColumnComponent For information about the properties that this component supports, see Section A.10, "ModelDrivenColumnComponent Subcomponent Properties."
Visible	Boolean	Y	Write an EL expression that resolves to <code>True</code> or <code>False</code> . If <code>True</code> , the column appears in the ADF Table component. If <code>False</code> , the column does not appear. <code>True</code> is the default value. If you make a column dynamic, the ADF Table component ignores the value of the <code>Visible</code> property. For more information about dynamic columns, see Section 7.16, "Adding a Dynamic Column to Your ADF Table Component."
Width			Specify the width of the column in number of characters. The value is used when <code>ResizeMode</code> is <code>SpecifiedWidth</code> . For more information, see Section 7.17, "Configuring Column Widths in an ADF Table Component."

A.12.3 ADF Table Component Actions

[Table A–14](#) describes the component actions available for use with the ADF Table component.

Table A-14 ADF Table Component Actions

Component Action	Description
ClearCachedRowAttributes	Clears the values of cached attributes for the current row of the ADF Table component. Only a <code>DoubleClickActionSet</code> in an ADF Table component's column should invoke this action.
DeleteFlaggedRows	Invokes a specified action on each of a set of flagged rows in the ADF Table component and then removes these rows from the ADF Table component. For more information, see Section 7.10, "Configuring an ADF Table Component to Delete Rows in the Fusion Web Application."
DisplayRowErrors	Displays error details for the current row in the ADF Table component if error details are available. This action should only be invoked from a column's action set in an ADF Table component. By default, the <code>_ADF_StatusColumn</code> described in Table 7.12 is configured with an action set that invokes this action.
DisplayTableErrors	Displays a detailed list of errors in a message dialog for the ADF Table component if any errors are available. Do not invoke this action from a column's action set in an ADF Table component. Instead configure an action set for an ADF Button, ADF Output Text component, or worksheet ribbon button to invoke this action.
Download	Download the rows corresponding to the current state of <code>TreeID</code> . For information about <code>TreeID</code> , see Section A.12.1, "ADF Table Component Properties."
DownloadFlaggedRows	Downloads the flagged rows from the tree binding specified by <code>TreeID</code> . For information about <code>TreeID</code> , see Table A-12 . This action applies to the downloaded rows only, and inserted rows are ignored.
DownloadForInsert	This action is obsolete. For more information, see Section 7.4, "Configuring Oracle ADF Component to Download Data to an ADF Table Component."
FlagAllRows	Sets the flag for all rows. Invoke this action to set a flag character in all rows of the <code>_ADF_FlagColumn</code> column. The flag character has the following properties: Character Code 25CF, Unicode(hex) It appears as a solid circle. For more information about the <code>_ADF_FlagColumn</code> column, see Section 7.11.2, "Row Flagging in an ADF Table Component" and Section 7.12, "Special Columns in the ADF Table Component."
Initialize	This action performs the following actions: <ul style="list-style-type: none"> ■ Removes all rows of data from the ADF Table component ■ Clears the values of cached attributes from rows in the ADF Table component ■ Creates the placeholder row ■ Recalculates how many dynamic columns to render in the ADF Table component ■ Redraws column headers If the ADF Table component contains pending changes that have not been saved in the integrated Excel workbook, a dialog appears to the end user that allows cancellation of invocation of this action.
MarkAllRowsChanged	Specify this component action to mark all rows in the table as changed in <code>_ADF_ChangeColumn</code> .
MarkAllRowsUnchanged	Specify this component action to clear all flags from the <code>_ADF_ChangedColumn</code> column.

Table A–14 (Cont.) ADF Table Component Actions

Component Action	Description
ResizeColumns	Resizes the table columns depending on the values of the <code>Table.ResizeColumnsMode</code> and <code>Column.ResizeMode</code> properties. For more information, see Section 7.17, "Configuring Column Widths in an ADF Table Component."
RowDownSync	Synchronizes data from the row in the ADF Table component's iterator in the Fusion web application that corresponds to the current worksheet row to the worksheet. As this action acts upon the current worksheet row, only a <code>DoubleClickActionSet</code> associated with a column in the ADF Table component should invoke this action. The ADF Table component does not evaluate or apply the value of a column's <code>Visible</code> property when invoking <code>RowDownSync</code> . The ADF Table component evaluates and applies the value of a column's <code>CellStyleName</code> property when invoking <code>RowDownSync</code> . For more information about column properties, see Section A.12.2, "ADF Table Component Column Properties."
RowUpSync	Synchronizes any pending changes in the current worksheet row that the ADF Table component references to the Fusion web application. <code>RowUpSync</code> acts upon the current worksheet row so only a <code>DoubleClickActionSet</code> associated with a column in the ADF Table component should invoke this action. The <code>DoubleClickActionSet</code> that invokes <code>RowUpSync</code> also changes the position of the ADF Table component's iterator on the Fusion web application to the current worksheet row (assuming it exists in the Fusion web application).
UnflagAllRows	Removes flags from cells in the <code>_ADF_FlagColumn</code> column. For more information about the <code>_ADF_FlagColumn</code> , see Section 7.11.2, "Row Flagging in an ADF Table Component" and Section 7.12, "Special Columns in the ADF Table Component."
Upload	Uploads pending changes to the Fusion web application. Commits successful rows even if row failures occur. For more information, see Section 7.8, "Configuring an ADF Component to Upload Changes from an ADF Table Component." For more information about resolving data conflict between the Excel workbook and the Fusion web application, see Section 12.7, "Handling Data Conflicts When Uploading Data from a Workbook" .
UploadAllOrNothing	Uploads pending changes to the Fusion web application. Commits successful rows only if none of the rows fail. For more information about <code>Upload</code> action, see Section 7.8, "Configuring an ADF Component to Upload Changes from an ADF Table Component." For more information about <code>UploadAllOrNothing</code> action, see Section 7.9, "Configuring an ADF Table Component to Upload Changes Using UploadAllOrNothing Action."

A.13 ADF Read-only Table Component Properties and Actions

The ADF Read-only Table component exposes one action, `Download`. This action downloads the current rows in the table identified by the ADF Read-only Table property, `TreeID`. [Table A–15](#) describes `TreeID` and the other properties that the ADF Read-only Table component supports.

Table A–15 ADF Read-only Table Component Properties

Name	Type	EL	Description
Annotation			For information about this property, see Table A–1 .
Columns	Array	N	References an array of read-only columns. For information about the properties that a column in this array can support, see Table A–16 .
ComponentID			For information about this property, see Table A–1 .
Position			For information about this property, see Table A–1 .
RowLimit			For information about this group of properties, see Table A–12 .
TreeID	Tree binding	N	References a tree binding ID from the page definition file associated with the current worksheet if the ADF Read-only Table component was created by inserting a tree binding into the worksheet.

[Table A–16](#) lists alphabetically the properties that a column in the `ReadOnlyColumn` array can use.

Table A–16 ADF Read-only Table Component Column Properties

Name	Type	EL	Description
Annotation			For information about this property, see Table A–1 .
CellStyleName	String	Y	Write an EL expression that resolves to an Excel style name that is applied to each cell in the column.
HeaderLabel	String	Y	Write an EL expression that resolves to a label for the column header.
HeaderStyleName	String	Y	Write an EL expression that resolves to an Excel style name that is applied to each cell in the column header.
ID	String	N	Assign a name to the column to identify it and its purpose. The value that you assign for this property has no functional impact. However, you must specify a value and the value that you specify must be unique within the list of columns. It serves to help you keep track of columns in the ADF Read-only Table component.
OutputText	ADF Component		For information about the properties that this component supports, see Section A.3, "ADF Output Text Component Properties."

A.14 Action Set Properties

[Table A–17](#) lists alphabetically the properties that you can configure for an action set.

Table A–17 Action Set Properties

Name	Type	EL	Description
ActionOptions			This group of properties specifies options for invoking local and remote actions.
ActionOptions.AbortOnFailure	Boolean	N	When set to <code>TRUE</code> , the remaining actions in the array are not invoked if an action fails. If <code>FALSE</code> , all actions are invoked regardless of the success or failure of previous actions. The default value is <code>TRUE</code> .
ActionOptions.FailureActionID	Action binding	N	Specify the action binding to invoke if an action set does not complete successfully. For example, you could specify an action binding that rolls back changes made during the unsuccessful invocation of the action set.

Table A-17 (Cont.) Action Set Properties

Name	Type	EL	Description
ActionOptions.SuccessActionID	Action binding	N	<p>Specify an action binding to invoke if an action set completes successfully. For example, you could specify an action binding that runs a commit action.</p> <p>A value for this property is optional.</p>
Actions	Array	N	<p>Specifies an ordered array of actions. An action can be one of the following:</p> <ul style="list-style-type: none"> <li data-bbox="691 470 1349 663"> <p>■ <code>ADFmAction</code></p> <p>Invokes an action binding or method action binding in the underlying page definition file. The <code>ADFmAction.ActionID</code> property identifies the action binding or method action binding to invoke. For information about page definition files, see Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook."</p> <li data-bbox="691 680 1365 1045"> <p>■ <code>ComponentAction</code></p> <p>Invokes an action that a component on the worksheet exposes. <code>ComponentAction.ComponentID</code> identifies the component that exposes the action while <code>ComponentAction.Method</code> identifies the action to invoke.</p> <p>The ADF Table and ADF Read-only Table components are the only components in ADF Desktop Integration that expose actions. For information about these actions, see Section A.12, "ADF Table Component Properties and Actions" and Section A.13, "ADF Read-only Table Component Properties and Actions." For information about invoking component actions, see Section 8.2.2, "How to Invoke Component Actions in an Action Set."</p> <li data-bbox="691 1062 1333 1178"> <p>■ <code>WorksheetMethod</code></p> <p>Invokes a worksheet action. For information about worksheet actions, see Section A.16, "Worksheet Actions and Properties."</p> <li data-bbox="691 1194 1365 1310"> <p>■ <code>Confirmation</code></p> <p>Invokes a confirmation dialog. For more information about the properties that this action uses, see Section A.14.1, "Confirmation Action Properties."</p> <li data-bbox="691 1327 1349 1444"> <p>■ <code>Dialog</code></p> <p>Invokes a web page in a popup dialog or Excel's task pane. For more information, see Section 8.4, "Displaying Web Pages from a Fusion Web Application."</p>
Alert			<p>This group of properties determines if and how an alert-style dialog appears to the user to indicate that the requested action is complete. The dialog that appears contains one button that allows the user to acknowledge the message and dismiss the dialog. For information about how to display an alert message, see Section 8.2.9, "How to Provide an Alert After the Invocation of an Action Set."</p> <p>Many properties in this group make use of EL expressions to retrieve string values from resource bundles. For more information about using EL expressions, see Section 10.2, "Using Resource Bundles in an Integrated Excel Workbook."</p>

Table A–17 (Cont.) Action Set Properties

Name	Type	EL	Description
Alert.Enabled	Boolean	N	Set to <code>TRUE</code> to display an alert message to end users that notifies them when an action set operation completes successfully or includes one or more failures. For more information, see Section 8.2.9, "How to Provide an Alert After the Invocation of an Action Set."
Alert.FailureMessage	String	Y	Specify an EL expression that evaluates to a message to appear in the dialog if errors occur during execution of the action set.
Alert.OKButtonLabel	String	Y	Specify an EL expression that evaluates to a message to appear in the OK button of the dialog.
Alert.SuccessMessage	String	Y	Specify an EL expression that evaluates to a message to appear in the dialog if no errors occur during the execution of the action set.
Alert.Title	String	Y	Specify an EL expression that evaluates to a message to appear in the title area of the dialog.
Annotation			For information about <code>Annotation</code> , see Table A–1 .
Status			This group of properties determines if and how a status message appears during the execution of an action set. For information about how to display a status message, see Section 8.2.5, "How to Display a Status Message While an Action Set Runs." Many properties in this group make use of EL expressions that reference string keys defined in resource bundles. For more information, see Section 10.2, "Using Resource Bundles in an Integrated Excel Workbook."
Status.AllowCancel	Boolean	N	If <code>TRUE</code> , a Cancel button is displayed in the status dialog box. For more information, see Section 8.2.5, "How to Display a Status Message While an Action Set Runs."
Status.Enabled	Boolean	N	If <code>TRUE</code> (default), a status window appears during the execution of the action set. If <code>FALSE</code> , no status window appears.
Status.Message	String	Y	Specify an EL expression to evaluate and display in the status window while the action set runs.
Status.Title	String	Y	Specify an EL expression to evaluate and display in the title area of the status window while the action set runs.
Status.Mode	String	N	Choose the visual appearance of progress bars. The valid values are <code>Automatic</code> , <code>BothBarsAlways</code> , <code>MainBarOnly</code> , <code>DetailBarOnly</code> , and <code>MainMessageOnly</code> .
Status.DetailStatusMessage	String	Y	Specify an optional EL expression or literal value that evaluates to a status message to appear as the associated action progresses.

A.14.1 Confirmation Action Properties

[Table A–18](#) lists alphabetically the properties that the `Confirmation` action in the array of `Actions` of an action set supports. For information about the other properties the array of `Actions` and action sets use, see [Table A–17](#).

Table A–18 Confirmation Action Properties

Name	Type	EL	Description
Annotation			For information about Annotation, see Table A–1 .
CancelButtonLabel	String	Y	An EL expression that is evaluated and displayed in the Cancel button at runtime.
OKButtonLabel	String	Y	An EL expression that is evaluated and displayed in the OK button at runtime.
Prompt	String	Y	An EL expression that is evaluated and displayed in the main area of the confirmation dialog at runtime.
Title	String	Y	An EL expression that is evaluated and displayed in the title area of the confirmation dialog at runtime.

A.14.2 Dialog Action Properties

[Table A–19](#) describes the properties that the Dialog action in the array of Actions of an action set supports. For information about the other properties the array of Actions and action sets use, see [Table A–17](#).

For information about how to use the properties in [Table A–19](#) to invoke a web page from a Fusion web application, see [Section 8.4, "Displaying Web Pages from a Fusion Web Application."](#)

Table A–19 Dialog Action Properties

Name	Type	EL	Description
Annotation	String	N	For information about this property, see Table A–1 .
Page	String	N	Specify the web page that the action invokes. Relative and absolute URLs are valid values.
ShareFrame	Boolean	N	Set to TRUE (default) to run the web page specified by the Dialog.Page property in the same data control frame as the Excel worksheet. If you specify an absolute URL, ADF Desktop Integration ignores the value of the Dialog.ShareFrame property.
Target	List	N	Specifies how the web page the action invokes is rendered. Select: <ul style="list-style-type: none"> ■ Popup to render the web page in a modal dialog within an embedded web browser. ■ TaskPane to render the web page in runtime task pane.
Title	String	Y	Write an EL expression that resolves to the title of the Dialog at runtime or write a literal string.
WindowSize	Integer	N	Specify the initial size in pixels of the dialog that appears to the user. Valid values range from 0 to 2147483647. Values will be revised upwards or downwards as appropriate at runtime if the specified values are too large or too small. The default value for Height is 625 and 600 for Width.

A.15 Workbook Actions and Properties

[Table A–20](#) describes the actions that a workbook can invoke. For information about configuring ribbon buttons to invoke these actions, see [Section 8.3.1, "How to Define a Workbook Command Button for the Runtime Ribbon Tab."](#)

Table A–20 Workbook Actions

Action	Description
Login	<p>When invoked, this action creates a new session between the integrated Excel workbook and the Fusion web application.</p> <p>If invoked when a session has already been established, it first invokes the <code>Logout</code> action internally to free that session. For a workbook running against a web application that is enforcing authentication, the <code>Login</code> action prompts the end user to provide valid user credentials.</p>
Logout	<p>When invoked, ADF Desktop Integration sends a request to the Fusion web application to invalidate the session between the integrated Excel workbook and the Fusion web application. After invoking this action, the end user must be authenticated the next time the Excel workbook accesses the Fusion web application.</p>
ClearAllData	<p>When invoked, this action clears all data entered by the user from cells that reference Oracle ADF bindings. Tables, such as those created by the ADF Table and ADF Read-only Table components, will be truncated so that they only display header rows with labels cleared. Values in cells that reference the Input Text or Output Text components are cleared. Column headers and labels are cleared as well. References to all resource bundles that the integrated Excel workbook uses are cleared. Worksheets that do not contain bindings or reference a page definition file remain unchanged. A dialog prompts the end user to confirm invocation of this action. Once the end user confirms invocation, ADF Desktop Integration runs the following events after invocation of the action:</p> <ul style="list-style-type: none"> ■ Invokes the integrated Excel workbook's <code>Logout</code> action ■ Terminates the runtime session and clears all data from the integrated Excel workbook and all caches ■ Reinitializes the integrated Excel workbook and invokes the workbook's <code>Login</code> action <p>Invocation of the <code>ClearAllData</code> action does not change data hosted by the Fusion web application. One or more of the following actions must be invoked to change data hosted by the Fusion web application:</p> <ul style="list-style-type: none"> ■ A worksheet's <code>UpSync</code> action <ul style="list-style-type: none"> This action synchronizes all data referenced by non-table type components. For more information, see Section A.16, "Worksheet Actions and Properties." ■ An ADF Table component's <code>RowUpSync</code> action can be used to synchronize any pending changes in a row to the Fusion web application. The ADF Table component's <code>DeleteFlaggedRows</code> action can be invoked to delete flagged rows. For more information about ADF Table component actions, see Section A.12.3, "ADF Table Component Actions."
EditOptions	<p>When invoked, this action launches a dialog that shows the current value of the <code>WebAppRoot</code> property and allows the end user to enter a new value.</p> <p>If the end user chooses to change the value of <code>WebAppRoot</code>, a confirmation dialog appears after the end user clicks OK. Once the change is confirmed, the following events occur:</p> <ul style="list-style-type: none"> ■ Workbook <code>ClearAllData</code> action is invoked ■ Workbook <code>Logout</code> action is invoked ■ All data referenced by bindings in the workbook is removed ■ References to <code>WebAppRoot</code> are updated in the Excel workbook's configuration ■ Workbook <code>Login</code> action is invoked to authenticate the user with the Fusion web application that is specified as the value for <code>WebAppRoot</code> <p>The <code>ClearAllData</code> workbook action clears all resource bundles referenced by the integrated Excel workbook. After <code>WebAppRoot</code> is changed, the integrated Excel workbook attempts to retrieve resource bundles from the Fusion web application as part of the reinitialization process. This request to the Fusion web application triggers the authentication process.</p>
ViewAboutDialog	<p>When invoked, this action launches a dialog called About that displays information defined in the <code>BrandingItems</code> workbook property and other information such as the versions of supporting software.</p>

[Table A-21](#) lists alphabetically the ADF Desktop Integration properties that an Excel workbook can use.

Table A-21 Workbook Properties

Name	Type	EL	Description
ApplicationHomeFolder	String	N	<p>Specify the absolute path to the directory that is the root for the JDeveloper application workspace (.jws) where you developed the desktop integration project.</p> <p>For example, the value of this property in a workbook integrated with the Summit sample application for ADF Desktop Integration could be something similar to the following:</p> <pre>D:\Oracle\Applications\Summit_ADFdi</pre> <p>ADF Desktop Integration prompts you to specify a value for this property the first time that you open an integrated Excel workbook.</p> <p>For more information, see Section 4.2.2, "How to Configure a New Integrated Excel Workbook."</p>
BrandingItems	Array	N	<p>An array of name-value pairs that resolve to resource bundle references (for example, #{res['myAppName']}) or a literal string. Each pair in the array consists of a name and a value. Each name and value can reference a literal string or an EL expression.</p> <p>For information about branding your integrated Excel workbook, see Section 9.6, "Branding Your Integrated Excel Workbook."</p>
CustomizationEnabled	Boolean	N	<p>Specify whether the workbook is customizable. If <code>True</code>, the published workbook will obtain its metadata from the server, which can be customized at runtime.</p> <p>For more information about enabling customization, see Section 14.6, "Customizing Workbook Integration Metadata at Runtime."</p>
Login.WindowSize	Integer	N	<p>Specify the initial size in pixels of the login dialog that appears to the user. Valid values range from 0 to screen width or height. Values will be revised upwards or downwards as appropriate at runtime if the specified values are too large or too small. The default value for <code>Height</code> is 625 and <code>Width</code> is 600.</p>

Table A–21 (Cont.) Workbook Properties

Name	Type	EL	Description
Parameters	Array	N	<p>An array of workbook initialization parameters that you configure to pass the parameters from a page in a Fusion web application to an integrated Excel workbook. You can define multiple workbook initialization parameters in the Fusion web application's page. Each workbook initialization parameter (parameter that references a URL argument) that you define in a page must be specified in a <code>Parameter</code> property of this array, otherwise it is ignored.</p> <p>Each element in the array supports the following properties:</p> <ul style="list-style-type: none"> ■ Annotation For more information about this property, see Table A–1. ■ Parameter You specify the name of the workbook initialization parameter you defined in the page of the Fusion web application from which the end user downloads the integrated Excel workbook. <p>For information about using this property, see Section 14.5, "Passing Parameter Values from a Fusion Web Application Page to a Workbook."</p>
Project	String	N	<p>Specify the name of a JDeveloper project in the current JDeveloper workspace. ADF Desktop Integration attempts to load the <code>.jpr</code> file that corresponds to the project that you specify. An error appears if the <code>.jpr</code> file is not available or is not in the expected format.</p> <p>When you open an integrated Excel workbook for the first time in design mode, ADF Desktop Integration searches for a <code>.jpr</code> file in the parent folder hierarchy. If it finds a <code>.jpr</code> file, it sets the value of <code>Project</code> to the name of the project that corresponds to the <code>.jpr</code> file.</p> <p>ADF Desktop Integration loads the names of the available projects from the <code>application_name.jws</code> file specified by <code>ApplicationHomeFolder</code>.</p>
RemoteServletPath	String	N	<p>Specify the path to the ADF Desktop Integration remote servlet. This path must be relative to the value specified for <code>WebAppRoot</code>. Note that the value you specify for <code>RemoteServletPath</code> must match the value that is specified in the web application's deployment descriptor file (<code>web.xml</code>). The default value for this property is:</p> <pre data-bbox="784 1425 1016 1446">/adfdiRemoteServlet</pre>

Table A-21 (Cont.) Workbook Properties

Name	Type	EL	Description
Resources	Array	N	<p>Specifies an array of resource bundles to register with the workbook. Each element in the array supports the following properties:</p> <ul style="list-style-type: none"> ■ Alias Specify a string value that is unique within <code>Workbook.Resources</code>. EL expressions use this string to reference the resource bundle. ■ Annotation For more information about this property, see Table A-1. ■ Class Specify a fully qualified class name, but do not include the file extension. The class name that you specify is expected to be a Java resource bundle class that the Fusion web application you integrate your workbook with uses. For example, the <code>EditCustomers-DT.xlsx</code> workbook in the Summit sample application for ADF Desktop Integration references the following resource bundle: <code>oracle.summitdi.resources.UIStrings</code> <p>For more information, see Section 10.2, "Using Resource Bundles in an Integrated Excel Workbook."</p>
Runtime Ribbon Tab	-	-	<p>This group of properties defines whether and how a Ribbon tab appears in Excel at runtime. The following entries in this table describe the properties in the <code>Runtime Ribbon Tab</code> group. For more information about Ribbon tab and its commands, see Section 8.3, "Configuring the Runtime Ribbon Tab."</p>
Runtime Ribbon Tab.Annotation	String	N	<p>For information about this property, see Section A.1, "Frequently Used Properties in the ADF Desktop Integration."</p>
Runtime Ribbon Tab.Visible	Boolean	N	<p>If <code>TRUE</code>, the Ribbon tab appears at runtime. The Ribbon tab does not appear if you set <code>Enabled</code> to <code>FALSE</code>. <code>TRUE</code> is the default value.</p>
Runtime Ribbon Tab.Title	String	Y	<p>Specify an EL expression that evaluates to the title that appears for the Ribbon tab in the title area. Excel imposes a maximum limit of 1024 characters for Ribbon tab titles. Ensure that the runtime value of the EL expression you specify does not exceed 1024 characters as ADF Desktop Integration truncates the value so that Excel does not generate an error message.</p> <p>If you choose to assign a key tip character using the <code>&</code> character, consider avoiding the letter <code>K</code> for the <code>Runtime Ribbon Tab.Title</code>. Excel does not allow the letter <code>K</code> to be used here when the workbook is running in the <code>ar_SA</code> culture.</p>

Table A–21 (Cont.) Workbook Properties

Name	Type	EL	Description
Runtime Ribbon Tab.Workbook Commands	Array	N	<p>Each element in this array corresponds to a workbook command at runtime. Each element in the array uses the following properties:</p> <ul style="list-style-type: none"> ■ Annotation For more information about this property, see Table A–1. ■ Label For more information about this property, see Table A–1. If you want the & character to appear in the command label, you must specify &&. Excel interprets a single & character as a special character, and assigns the next character after & as the keyboard accelerator for the workbook command at runtime. ■ Method Specify the workbook action that the workbook ribbon button invokes. For more information about workbook actions, see Table A–20.
WebAppRoot	String	N	A fully qualified URL to the Fusion web application's root.
WebPagesFolder	String	N	Specify the path to the directory that contains the web pages that you intend to use with your integrated Excel workbooks. The value that you specify for the path must be relative to the value of <code>ApplicationHomeFolder</code> .
WorkbookID	String	N	<p>A unique identifier for the integrated Excel workbook. ADF Desktop Integration generates the unique identifier when you open the workbook for the first time in design mode.</p> <p>The value cannot be modified. However, ADF Desktop Integration can generate a new value if you use the Reset WorkbookID link in the Edit Workbook Properties dialog.</p> <p>The value of this property is used during tamper check, as described in Section 11.3, "Checking the Integrity of an Integrated Excel Workbook's Metadata."</p>

A.16 Worksheet Actions and Properties

An Excel worksheet with ADF Desktop Integration can invoke the following actions:

- `UpSync`
Synchronizes any pending changes from the ADF Input Text and ADF List of Values components in the worksheet to the Fusion web application.
- `DownSync`
Downloads any changes from the Fusion web application to the ADF Input Text, ADF Output Text, and ADF List of Values components in the worksheet.
- `DisplayWorksheetErrors`
Displays a detailed list of errors in a message dialog for the integrated Excel worksheet if any errors are available. Invoke this action in an action set that is invoked by an ADF component (other than the ADF Table-type components) or a worksheet ribbon button.

When you configure an ADF Button component to invoke an action binding or method action binding, the action set to invoke when a user clicks the ADF Button component at runtime is populated as follows by default:

1. UpSync
2. Action or method action binding that you specify for the ADF Button component
3. DownSync

If the first action that you invoke on a worksheet with an empty form is the UpSync worksheet action, you may encounter errors. For this reason, ensure that the first action invoked is the DownSync worksheet action. You can configure the ADF Button component's action set or one of the worksheet events (Startup or Activate) described in [Table A-22](#) to invoke the DownSync worksheet action first.

[Table A-22](#) describes the ADF Desktop Integration properties that an Excel worksheet can use.

Table A-22 *Worksheet Properties*

Name	Type	EL	Description
Annotation	String	N	For information about this property, see Table A-1 .
Events	Array	N	Each element in this array specifies an action set to invoke if the associated worksheet event occurs. For information about action sets, see Section A.14, "Action Set Properties." For information about worksheet events, see the entry in this table for <code>Events.n.Event</code> . The following entries in this table prefaced by <i>Events.n</i> describe the properties that an element in this array supports where <i>n</i> refers to a specific element in the array.
<code>Events.n.ActionSet</code>	ActionSet	N	For more information about the properties of action sets, see Section A.14, "Action Set Properties."
<code>Events.n.InvokeOnceOnly</code>	Boolean	N	The default value of this property is <code>FALSE</code> . When set to <code>TRUE</code> , the workbook stores information about whether the worksheet invoked the action set for this event and, if so, prevents the worksheet from invoking the action set a second time. Note that if the workbook is not saved, this information is lost. This means that the worksheet can invoke the event again the next time that the workbook opens.
<code>Events.n.Annotation</code>	String	N	For information about the annotation property, see Table A-1 .

Table A–22 (Cont.) Worksheet Properties

Name	Type	EL	Description
Events.n.Event	List	N	<p>The worksheet supports the following events that you can configure to invoke an action set:</p> <ul style="list-style-type: none"> ■ Startup Excel workbook opens and the worksheet is activated for the first time. ■ Shutdown Excel workbook closes or Excel application exits. ■ Activate User navigates to the current worksheet. ■ Deactivate User navigates away from the current worksheet or Shutdown event triggered. <p>Note that the worksheet events complete execution even if the action sets that they invoke fail.</p> <p>For more information about worksheet events and action sets, see Section 8.2.4, "How to Invoke an Action Set from a Worksheet Event."</p>
Protection.Mode	List	N	<p>The worksheet provides two options:</p> <ul style="list-style-type: none"> ■ Off Worksheet protection is not used at runtime. ■ Automatic Worksheet protection is enabled automatically at runtime. <p>The default value for this property is Off.</p>
Protection.Password	String	N	<p>Specify a password to prevent end-users from turning off sheet protection at runtime. The maximum password length allowed by Excel is 255 characters.</p>
Ribbon Commands	Array	N	<p>Specify one or more workbook actions that appear as commands at runtime. Each command is an element in the <code>WorksheetMenuItem</code> array. Entries in this array support the following properties:</p> <ul style="list-style-type: none"> ■ Annotation ■ Label ■ SelectActionSet <p>For more information about the <code>Annotation</code> and <code>Label</code> properties, see Table A–1. For more information about the <code>SelectActionSet</code> property, see Section A.14, "Action Set Properties."</p> <p>If you want the <code>&</code> character to appear in the command label, you must specify <code>&&</code>. Excel interprets a single <code>&</code> character as a special character, and assigns the next character after <code>&</code> as the keyboard accelerator for the worksheet command at runtime.</p>
Page Definition	String	N	<p>Specify the page definition file to associate with the worksheet. For information about page definition files, see Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook."</p>

Table A–22 (Cont.) Worksheet Properties

Name	Type	EL	Description
Parameters	Array	N	<p>An array of worksheet parameters that you configure to pass the parameters from a workbook <code>Parameters</code> property to a worksheet in an integrated Excel workbook. Each element in the array supports the following properties:</p> <ul style="list-style-type: none"> ■ Annotation For more information about this property, see Table A–1. ■ Parameter Specify the ID of a parameter element that you added to the page definition file associated with the worksheet. ■ Value Write an EL expression that references the value of the <code>Parameter</code> property you specified for the workbook initialization parameter (workbook <code>Parameters.Parameter</code> property). The workbook <code>Parameters.Parameter</code> property supplies this value the first time that the page definition file associated with this worksheet is initialized. <p>For information about using this property, see Section 14.5, "Passing Parameter Values from a Fusion Web Application Page to a Workbook."</p>
RowData			<p>Set values for the <code>CachedAttributes</code> property when you want to cache data in an integrated Excel workbook across a multiple sessions with the Fusion web application.</p> <p>Set a value for the <code>ChangeIndicatorAttributeID</code> property to determine if a row has been modified by another user since you downloaded it from the Fusion web application.</p>
RowData.CachedAttributes	Array	N	<p>Specify values for the properties in this array to determine the attributes for which data is cached. Each <code>CachedAttribute</code> element in this array supports the following properties:</p> <ul style="list-style-type: none"> ■ <code>AttributeID</code> This property references the attribute binding for which data is to be cached. Do not specify an attribute binding for <code>AttributeID</code> and as an editable field in a form (for example, in an ADF Input Text component) in the same worksheet. ■ Annotation For more information about this property, see Table A–1. <p>For more information about clearing the values of cached attributes, see Section 7.21, "Clearing the Values of Cached Attributes in an ADF Table Component."</p>
RowData.ChangeIndicatorAttributeID	Binding	N	<p>Specify the row-specific attribute of the tree binding used to determine if a row has been modified by another user since the row was last downloaded by to your integrated Excel workbook.</p> <p>For more information, see Section 12.7, "Handling Data Conflicts When Uploading Data from a Workbook."</p>

Table A–22 (Cont.) Worksheet Properties

Name	Type	EL	Description
ServerContext			<p>This group of properties references the attribute bindings that uniquely identify the row displayed in the current worksheet so that you can reestablish server data context across multiple sessions.</p> <p>For more information, see Section 15.2, "Restore Server Data Context Between Sessions."</p>
ServerContext.CacheDataContexts	Array	N	<p>Add elements to the CacheDataContexts array for cases where there is more than one iterator defined in the binding container whose server-side context must be reestablished. The CacheDataContexts array supports the following properties to store the worksheet's cached data context:</p> <ul style="list-style-type: none"> <li data-bbox="857 625 1321 688">■ RestoreDataContextActionID References an action binding to invoke. <li data-bbox="857 705 1446 1167">■ CachedServerContexts An array that identifies the attribute binding values to cache and set before the action binding specified by RestoreDataContextActionID is invoked. Each element in the CachedServerContext array supports the CachedAttributeID and RestoredAttributeID properties. CachedAttributeID identifies the attribute binding value to cache in the worksheet. RestoredAttributeID is an optional property for which you specify a value when the destination attribute binding value is different from the source attribute binding value. If you do not specify a value for RestoredAttributeID, the value of CachedAttributeID is used as the destination attribute binding value and its value is set before invoking the action set. <li data-bbox="857 1184 1403 1297">■ Annotation For more information about this property, see Section A.1, "Frequently Used Properties in the ADF Desktop Integration."
ServerContext.IDAttributeID	Binding	N	<p>Specifies an attribute binding that uniquely identifies the row displayed in the current worksheet. This property is used at runtime to determine whether the server context has been reestablished properly for non-table type components in the worksheet.</p>

Table A-22 (Cont.) Worksheet Properties

Name	Type	EL	Description
ServerContext.SendParameters	Boolean	N	<p>The default value of this property is <code>FALSE</code>.</p> <p>When set to <code>TRUE</code>, the workbook sends initialization parameters for this worksheet when reestablishing context across multiple sessions.</p>
SetupActionID	Binding	N	<p>Specify the ADFm action binding to be invoked before the binding container metadata is retrieved.</p> <p>A value for this property is optional.</p> <p>If two, or more, worksheets are using the same page definition, the action binding specified for the last worksheet will be invoked. Hence, create a page definition for each worksheet and do not specify a page definition to multiple worksheets.</p>
Title	String	Y	<p>Specifies an EL expression that resolves to a string and sets the name of the worksheet. At design time, the EL expression can be of any length and can include the following special characters:</p> <p>[] \ / * ?</p> <p>At runtime, the evaluated string can display a maximum of 31 characters and ignores the above special characters. If the length of the evaluated string exceeds 31 characters, the extra characters are truncated and are not displayed.</p> <p>Note that the <code>Title</code> property does not support binding parts in the EL expression. The expected usage is a resource-type expression.</p> <p>Ensure that the EL expressions you write for the <code>Title</code> property generate unique values for each worksheet at runtime and contain fewer than 31 characters.</p>

ADF Desktop Integration EL Expressions

This appendix describes the syntax for EL expressions in ADF Desktop Integration, provides guidelines for writing EL expressions, and how to use attribute control hints in EL expressions.

This appendix includes the following sections:

- [Section B.1, "Guidelines for Creating EL Expressions"](#)
- [Section B.2, "EL Syntax for ADF Desktop Integration Components"](#)
- [Section B.3, "Attribute Control Hints in ADF Desktop Integration"](#)

B.1 Guidelines for Creating EL Expressions

The following list describes the characteristics that EL expressions for your integrated Excel workbook can have and provides suggestions for writing EL expressions:

- Literal values that evaluate correctly to the type expected for the Oracle ADF component property. The following list describes some examples:
 - Boolean values `true` and `false`
 - Integer values such as `-1`, `0`, and `100`
 - String values such as `hello world`
- Strings that contain one or more valid EL expression parts. The following list shows examples of valid syntax:
 - `{row.bindings.ProductId.inputValue}`
 - `{components.TAB416222534.errors}`
 - `{res['excel.saveButton.label']}`
- A valid Excel formula. An Excel formula string must start with the `=` character. If the literal string includes an `{...}` expression, ADF Desktop Integration evaluates this expression first and inserts the resulting value into the Excel formula string. Excel then evaluates the Excel formula.

Note the following points if you write an EL expression:

- Excel formula elements must not be used inside an `{...}` expression.
- EL expressions should not contain references to Excel cells because EL expressions are managed within ADF configuration. Excel cannot update the ADF configuration if the referenced cell moves. A workaround is to define a named cell reference or range using the **Name** box in the Excel Formula Bar. You can reference the named cell reference or named cell range reference from

an EL expression. For information about defining named cell references or ranges, see Excel's documentation.

- EL expressions in a page definition file

For information about the syntax that you use to write EL expressions in a page definition file, see [Section 4.3, "Working with Page Definition Files for an Integrated Excel Workbook."](#)
- Excel formulas that include EL expressions

Ensure that any Excel formula that includes EL expression has no more than 255 characters. This also applies to formulas used to set conditional values to component properties.

B.2 EL Syntax for ADF Desktop Integration Components

[Table B-1](#) lists supported expression properties for the ADF Desktop Integration components that support EL expressions.

The EL expressions use the following syntax to reference these properties:

```
{components.componentID.property}
```

where *componentID* references the ID of the component and *property* references the property (for example, *rowCount*).

Table B-1 Expression Properties for ADF Desktop Integration Components

Property	Component Type	Property Type	Expected Runtime Values	Value at Design Time
rowCount	Table Read-only Table	Int	>=0	0
currentRowIndex	Table Read-only Table	Int	>= 0 AND < RowCount (zero based index)	-1
currentRowMode	Table	String	"insert" "update"	"unknown"
errors	Table	String	N/A	N/A
readOnly	Table.Column	Boolean	TRUE FALSE	FALSE

Write EL expressions with the following syntax to retrieve:

- Worksheet errors at runtime


```
{worksheet.errors}
```

For more information about worksheet errors, see [Section 12.4, "Error Reporting in an Integrated Excel Workbook."](#)
- Workbook initialization parameters


```
{workbook.params.parameterName}
```

where *parameterName* is the name of the workbook initialization parameter. For information about using these parameters, see [Section 14.5, "Passing Parameter Values from a Fusion Web Application Page to a Workbook."](#)
- Resource bundle string key values


```
#{resourceBundleAlias['resourceBundleKey']}
```

where *resourceBundleAlias* is the alias of the resource bundle and *resourceBundleKey* is the string key value. For more information about resource bundles, see [Section 10.2, "Using Resource Bundles in an Integrated Excel Workbook."](#)

[Table B-2](#) describes the supported syntax and properties for Oracle ADF control bindings. For information about the attribute control hints (*controlHint*) that ADF Desktop Integration supports, see [Table B-3](#).

You can use the expression builder described in [Section 5.8, "Using the Expression Builder"](#) to generate some of the EL expressions described in [Table B-2](#). You have to write some other EL expressions as indicated in [Table B-2](#).

Table B-2 Expression Properties and Syntax for Oracle ADF Control Bindings

Syntax	Component Type	Object Property	Value at Design Time
<p>Use the expression builder to generate EL expressions with the following syntax:</p> <pre>#{bindings.attributeID} #{bindings.attributeID.label} #{bindings.attributeID.hints.controlHint}</pre> <p>You can also write the previous EL expressions in addition to the following EL expression:</p> <pre>#{bindings.attributeID.inputValue}</pre>	Attribute	Attribute control hint	" "
<p>Use the expression builder to generate EL expressions with the following syntax:</p> <pre>#{bindings.ListID} #{bindings.ListID.label} #{bindings.ListID.hints.controlHint}</pre>	List	Attribute control hint	" "
<p>Write EL expressions with the following syntax for a column in a table-type component</p> <pre>#{row.bindings.attributeID.inputValue}</pre>	Table.Column n	inputValue	" "
<p>Write an EL expression with the following syntax when adding a dynamic column to an ADF Table component as described in Section 7.16, "Adding a Dynamic Column to Your ADF Table Component":</p> <pre>#{bindings.TreeID.[TreeNodeID].AttributeNamePrefix*.inputValue} #{bindings.TreeID.AttributeNamePrefix*.inputValue} #{bindings.TreeID.[TreeNodeID].hints.AttributeNamePrefix*.controlHint} #{bindings.TreeID.[TreeNodeID].hints.AttributeNamePrefix*.label}</pre> <p>A value for <i>AttributeNamePrefix</i> and <i>[TreeNodeID]</i> is optional while <i>*</i> is required.</p>			

B.3 Attribute Control Hints in ADF Desktop Integration

ADF Desktop Integration can read the values of the attribute control hint names described in [Table B-3](#). You write EL expressions that ADF Desktop Integration uses to retrieve the value of an attribute control hint from your Fusion web application. [Table B-2](#) describes the EL expression syntax that retrieves the values of attribute control hints at runtime.

You configure attribute control hints in your Fusion web application. Information about how to add an attribute control hint to an entity object can be found in the "Defining Attribute Control Hints for Entity Objects" section of the *Developing Fusion Web Applications with Oracle Application Development Framework*. Information about how to add an attribute control hint to a view object can be found in the "Defining UI Hints for View Objects" section of the *Developing Fusion Web Applications with Oracle Application Development Framework*.

Table B-3 Attribute Control Hints Used by ADF Desktop Integration

Attribute Control Hint	Type	Value to configure in the Fusion web application
label	String	Returns the value of the label attribute control hint configured for an entity or view object.
updateable	Boolean	Returns true if the associated attribute binding is updatable.
readOnly	Boolean	<p>This attribute control hint is unique to ADF Desktop Integration. Returns true if the associated attribute binding is not updatable.</p> <p>To optimize the performance of an integrated Excel workbook when it evaluates Excel formulas in EL expressions, you should write an EL expression with the following syntax for a component's <code>readOnly</code> property:</p> <pre>#{bindings.attributeID.hints.readOnly}</pre> <p>rather than:</p> <pre>=NOT(#{bindings.attributeID.hints.updateable})</pre> <p>Note that the attribute control hint <code>readOnly</code> property differs to the <code>readOnly</code> property of ADF Desktop Integration components described in Section A.1, "Frequently Used Properties in the ADF Desktop Integration."</p>
mandatory	Boolean	Returns true if a value for the associated attribute binding is required.
dataType	String	<p>Returns the data type of the attribute control hint. A Fusion web application can support many data types with complex names. The <code>dataType</code> attribute control hint was introduced in ADF Desktop Integration to simplify the writing of EL expressions. It maps the data types that a Fusion web application supports to the values supported by ADF Desktop Integration listed here:</p> <ul style="list-style-type: none"> ■ string ■ number ■ date ■ boolean ■ other
tooltip	String	Returns the message value of the Tooltip attribute control hint configured for an entity or view object.
displayWidth	String	<p>Returns the value of the <code>width</code> attribute control hint configured for an entity or the view object. The value represents the width in number of characters.</p> <p>The <code>displayWidth</code> hint can be used in a table column's <code>width</code> property when <code>ResizeMode</code> for that column is set to <code>SpecifiedWidth</code>.</p>

The ADF Desktop Integration attribute control hints are based on information available in the web application's model configuration. ADF Desktop Integration supports view object or entity object hint values, but does not support programmatic overrides of hint values if they are calculated at a row-by-row level at runtime.

Troubleshooting an Integrated Excel Workbook

This appendix describes how to troubleshoot an integrated Excel workbook and generate log files when you encounter problems during development. It also describes possible solutions for a number of errors and problems (such as version mismatch, 404 error, and Oracle ADF tab not visible in integrated Excel workbook) that you may encounter.

This appendix includes the following sections:

- [Section C.1, "Verifying That Your Fusion Web Application Supports ADF Desktop Integration"](#)
- [Section C.2, "Verifying End-User Authentication for Integrated Excel Workbooks"](#)
- [Section C.3, "Generating Log Files for an Integrated Excel Workbook"](#)
- [Section C.4, "Common ADF Desktop Integration Error Messages and Problems"](#)

Note: The property inspector does not validate that values you enter for a property or combinations of properties are valid. Invalid values may cause runtime errors. To avoid runtime errors, make sure you specify valid values for properties in the property inspector. For more information about the property inspector, see [Section 5.6, "Using the Property Inspector."](#)

C.1 Verifying That Your Fusion Web Application Supports ADF Desktop Integration

Using a specific URL, you can verify that the Fusion web application is running the ADF Desktop Integration remote servlet (`adfDiRemote`), and the version of ADF Desktop Integration. This information can be useful if you encounter errors with an integrated Excel workbook. For example, you can determine whether the ADF Desktop Integration remote servlet is running when you are troubleshooting an integrated Excel workbook.

To verify that the ADF Desktop Integration remote servlet is running:

1. Log on to the Fusion web application.
2. Type the concatenated values of the workbook properties `WebAppRoot` and `RemoteServletPath` into the address bar of your web browser. This corresponds to a URL similar to the following:

`http://hostname:7101/FusionApp/adfdiRemoteServlet`

If the ADF Desktop Integration remote servlet is running, a web page returns displaying a message similar to [Figure C-1](#).

Figure C-1 ADF Desktop Integration Remote Servlet

Oracle ADF Desktop Integration 12c

Diagnostic Information	
Status	Active
Version	12.1.3.0.0.7731
Build	ADFDI_MAIN_NT_130522.1901

[About ADF Desktop Integration](#)

Copyright © 1997, 2013, Oracle and/or its affiliates. All rights reserved.

C.2 Verifying End-User Authentication for Integrated Excel Workbooks

If end users of an integrated Excel workbook do not get prompted for user credentials when they invoke an action that interacts with the Fusion web application configured with ADF security, it may mean that security is not configured correctly for either the integrated Excel workbook or the Fusion web application. You can verify that your secure Fusion web application authenticates end users and that it is security-enabled by carrying out the following procedure.

To verify that a secure Fusion web application authenticates end users, in the web browser's address bar, enter the URL that you used to verify whether ADF Desktop Integration remote servlet is running. For more information, see [Section C.1, "Verifying That Your Fusion Web Application Supports ADF Desktop Integration."](#) If the Fusion web application is security-enabled, it will request that you enter user credentials.

For more information about securing your integrated Excel workbook, see [Chapter 11, "Securing Your Integrated Excel Workbook."](#)

C.3 Generating Log Files for an Integrated Excel Workbook

ADF Desktop Integration can generate log files that capture information based on events triggered by the following pieces of software within ADF Desktop Integration:

- HTTP filter and the ADF Desktop Integration remote servlet on the web server (server-side logging)
For more information about server-side logging, see [Section C.3.1, "About Server-Side Logging."](#)
- Excel workbook which you integrate with your Fusion web application (client-side logging)
For more information about client-side logging, see [Section C.3.2, "About Client-Side Logging."](#)

C.3.1 About Server-Side Logging

To know more about server-side logging, see the "Server-Side Logging" section in *Administering Oracle ADF Applications*.

C.3.2 About Client-Side Logging

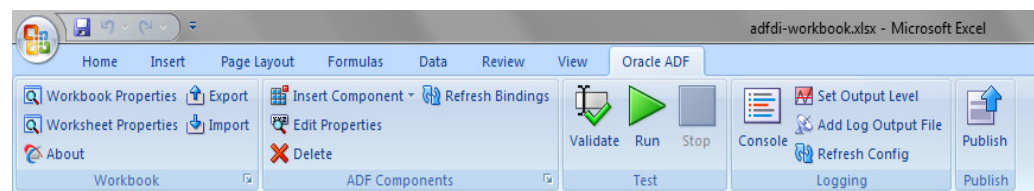
You can configure ADF Desktop Integration to save logs of triggered events on the client. By default, no log files are generated. For more information about how to configure ADF Desktop Integration module to save logs, see [Section C.3.2.1, "How to Configure ADF Desktop Integration to Save Logs."](#)

C.3.2.1 How to Configure ADF Desktop Integration to Save Logs

ADF Desktop Integration provides logging tools to generate event logs and make them easily accessible. The logging tools are located in the Logging group of the **Oracle ADF** tab, and are available in both the design mode and the test mode.

[Figure C-2](#) shows the logging tools in the **Oracle ADF** tab.

Figure C-2 Logging Tools in Oracle ADF Tab



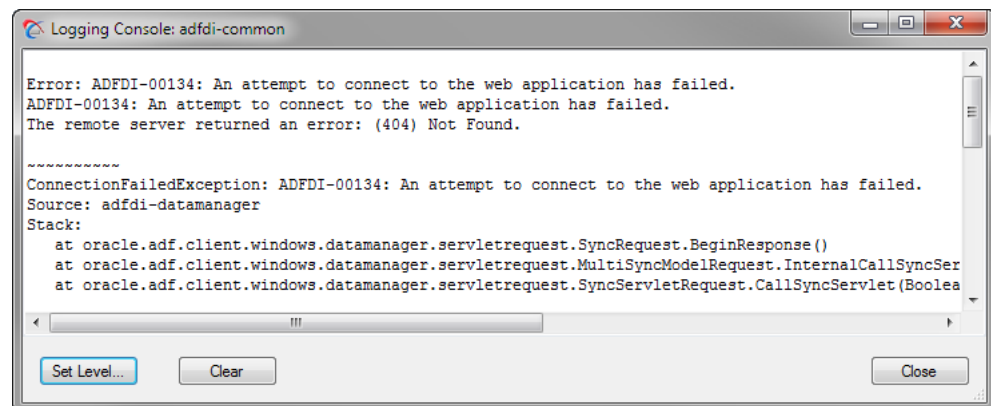
The Logging group provides the following buttons:

- **Console**

Displays the Logging Console window, which enables you to review the recent log entries while you are developing and testing the integrated Excel workbook. The console displays entries that are logged while the console is open. [Figure C-3](#) illustrates the Logging Console window with error log entries.

The console is a resizable, non-modal window with a buffer size of 64,000 characters. When the buffer is full, the old entries are removed. To save log entries, select and copy them to a text file.

Figure C-3 Logging Console Window



The dialog has the following buttons:

- **Set Level:** Click to set the log output level. The button opens the Logging Output Level dialog, where you can choose the desired log output level.
- **Clear:** Click to clear the log buffer.

- **Close:** Click to close the dialog.

Note: A common Logging Console window logs entries for all open integrated Excel workbooks.

- **Set Output Level**

Prompts you to choose the log output level. [Table C-1](#) describes the log levels that client-side logging supports.

Figure C-4 Logging Output Level Dialog

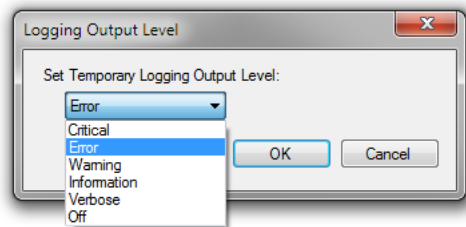


Table C-1 Client-Side Logging Levels

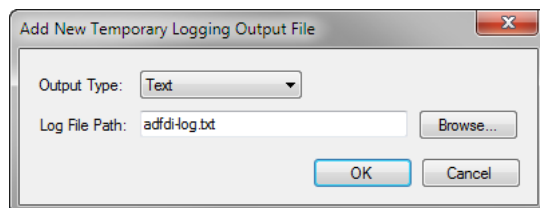
Level	Description
Critical	Captures critical information.
Error	Captures information about severe errors and exceptions.
Warning	Captures irrecoverable conditions.
Information	Captures lifecycle and control flow events.
Verbose	Captures detailed information about the execution flow of the application.
Off	No logs are captured. This is the default value.

Note: The log output level applies to all listeners for a given logger.

- **Add Log Output File**

Creates a new temporary logging listener to direct logging output to the specified file or format. In the Add New Temporary Logging Output File dialog, choose the desired file output type (text or XML), and specify the path and file name of the log output file.

Figure C-5 Add New Temporary Logging Output File Dialog



The temporary listener directs the logging output for the current Excel session only, and is not registered in the ADF Desktop Integration configuration file. After you close the integrated Excel workbook, the temporary listener is removed.

Note: When you click the **Add Log Output File** button, a new listener is created. The new listener does not replace any existing listener defined in the ADF Desktop Integration configuration file, or any other temporary listener.

- **Refresh Config**

Reloads the ADF Desktop Integration configuration file. The ADF Desktop Integration configuration file determines the type of information logged by ADF Desktop Integration. It also determines the location and the output format of the log file.

For more information about the creation and configuration of the ADF Desktop Integration configuration file, see [Section C.3.2.2, "About the ADF Desktop Integration Configuration File."](#)

C.3.2.2 About the ADF Desktop Integration Configuration File

The ADF Desktop Integration configuration file is saved as `adfdi-excel-addin.dll.config` in the Designer edition, and as `adfdi-excel-addin-runtime.dll.config` in the Runtime edition. To determine the correct file name and location, click the **About** button in the Workbook group of the **Oracle ADF** tab. In the dialog that opens, click the **Properties** tab, and consult the **Configuration** entry for file name and location of configuration file.

For more information about elements of the configuration file, see the "Configuration File Schema for the .NET Framework" section in Microsoft Developer Network documentation. For more information about trace and debug settings, see the "Trace and Debug Settings Schema" section in Microsoft Developer Network documentation.

[Example C-1](#) shows a sample configuration file, one of many valid ways to configure client-side logging, that generates two different log files with different formats (`.txt` and `.xml`). The file captures different types of information such as `ThreadId`, `ProcessId`, and `DateTime` at a `Verbose` logging level.

Example C-1 Sample Configuration File

```
<?xml version="1.0"?>
<configuration>
  <system.diagnostics>
    <sources>
      <source name="adfdi-common" switchValue="Verbose">
        <listeners>
          <add type="System.Diagnostics.DelimitedListTraceListener"
              name="adfdi-common-excel.txt"
              initializeData="c:\logs\adfdi-common-excel.txt"
              delimiter="|"
              traceOutputOptions="ThreadId, ProcessId, DateTime"/>
          <add type="System.Diagnostics.XmlWriterTraceListener"
              name="adfdi-common-excel.xml"
              initializeData="c:\logs\adfdi-common-excel.xml"
              traceOutputOptions="None"/>
        </listeners>
      </source>
    </sources>
  </system.diagnostics>
</configuration>
```

```

    </sources>
  </system.diagnostics>
</configuration>

```

C.3.2.3 How to Configure Logging Using User Environment Variables

Users who do not have access to the directory that stores the ADF Desktop Integration configuration file can change the location where log files are saved, and the logging level by setting values for user environment variables. You can add two user environment variables to configure the logging level and location for XML log files.

To add or configure user environment variables on Windows:

1. Click the Windows **Start** button and then click **Control Panel**.
2. In the Control Panel, click **System**, and then **Advanced System Settings**.
3. In the **Advanced** tab of System Properties dialog, click **Environment Variables**.
4. In the Environment Variables dialog, click **New** under the *User variables for* **username** input field, and add variables as described in the [Table C-2](#).

Table C-2 *User Environment Variables to Configure Logging*

Enter a variable named...	With a value...
adfdi-common-file	That defines the directory path and file name for the XML file that captures logging information. The directory that you specify here must exist before you add the <code>adfdi-common-file</code> variable. The generated log file will be in XML format.
adfdi-common-level	That specifies the level of logging. Table C-1 lists valid values.

5. Click **OK**.

C.3.2.4 What You May Need to Know About the `adfdi-common` Object

The `adfdi-common` object is an instance of the `TraceSource` class from the `System.Diagnostics` namespace in the Microsoft .NET Framework. This object is used to generate log files that capture information about events triggered by the Excel workbook that you integrate with your Fusion web application. To know the location of the log file, check the **Log Files** attribute in the Properties tab of the About dialog.

For more information about the `TraceSource` class, see Microsoft Developer Network documentation.

C.4 Common ADF Desktop Integration Error Messages and Problems

While using or configuring the ADF Desktop Integration enabled Fusion web application or workbooks, you might see error messages or have some problems. The following list describes the most common error messages, their cause, and solutions.

Error message: [ADFDI-00127] A version mismatch was detected for `SyncServletResponse`. Version *x* was found, version *y* was expected

Cause: The client version of ADF Desktop Integration does not match the ADF Desktop Integration version in the web application.

Action: Uninstall client ADF Desktop Integration, and install the web application specific ADF Desktop Integration version. For more information about installing

ADF Desktop Integration client, see [Section 3.4, "Installing ADF Desktop Integration."](#)

Error message: 404 Error - servlet not found

Cause: The `web.xml` deployment descriptor settings are not in sync with `Workbook.RemoteServletPath` property value.

Action: Open Workbook Properties editor and verify the `Workbook.RemoteServletPath` property value.

Error message: Programmatic access to Visual Basic Project is not trusted

Cause: Macro settings in Excel are not enabled.

Action: Verify that the **Trust access to the VBA project object model** checkbox in the Trust Center dialog is enabled. For more information, see [Section 3.3, "Configuring Excel to work with ADF Desktop Integration."](#)

Problem: Oracle ADF tab is not visible in your integrated Excel Workbook after installing ADF Desktop Integration

Cause: The ADF Desktop Integration add-in is not enabled in Excel.

Action: Enable the ADF Desktop Integration add-in in the Excel Options dialog. In Excel, click the Microsoft Office button, and then click **Excel Options** to open the Excel Options dialog. In the Add-Ins tab, open the **Manage** dropdown list, choose **COM Add-ins**, and click **Go**. In the COM Add-ins dialog, select the **Oracle ADF Desktop Integration Add-in for Excel** checkbox and click **OK**.

For information about all ADF Desktop Integration error messages, see *Oracle Fusion Middleware Error Messages Reference*.

If you are a system administrator, you should also see the "Common ADF Desktop Integration Error Messages and Problems" section in *Administering Oracle ADF Applications*.

ADF Desktop Integration Settings in the Web Application Deployment Descriptor

This appendix describes the values that you set for the ADF Desktop Integration servlet (`adfdiRemote`) so that the Fusion web application can use it. The appendix also describes the values in the deployment descriptor file that determine the behavior of the HTTP filter that ADF Desktop Integration provides, and provides a code sample from a deployment descriptor file that shows these values in use.

This appendix includes the following sections:

- [Section D.1, "Configuring the ADF Desktop Integration Servlet"](#)
- [Section D.2, "Configuring the ADF Desktop Integration Excel Download Filter"](#)
- [Section D.3, "Examples in a Deployment Descriptor File"](#)

D.1 Configuring the ADF Desktop Integration Servlet

A Fusion web application with integrated Excel workbooks must contain entries in its deployment descriptor file (`web.xml`) to use the `adfdiRemote` servlet. The Excel workbooks that you integrate with a Fusion web application call this servlet to synchronize data with the Fusion web application. The `adf-desktop-integration.jar` file stores the servlet in the following directory:

```
MW_HOME\oracle_common\modules\oracle.adf.desktopintegration_12.1.3
```

where *MW_HOME* is the Middleware Home directory.

When you add ADF Desktop Integration to your project as described in [Section 4.2, "Adding an Integrated Excel Workbook to a Fusion Web Application,"](#) ADF Desktop Integration automatically configures your deployment descriptor with the necessary entries to enable the servlet (`DIRemoteServlet`) on your Fusion web application. If required, then you can configure the servlet manually.

To configure the ADF Desktop Integration servlet:

1. In JDeveloper, locate and open the deployment descriptor file (`web.xml`) for your ADF Desktop Integration project.

Typically, this file is located in the `WEB-INF` directory of your project.

2. Click the Servlets page, and then click the **Add** icon to create a row entry in the Servlets table. The icon is in the top-right corner of the servlets table.

Enter the values as described in [Table D-1](#) to enable the `adfdiRemote` servlet on the Fusion web application.

Table D-1 Values to Enable adfdiRemote Servlet

For this property...	Enter this value...
Name	adfdiRemote
Type	Servlet Class
Servlet Class/JSP file	oracle.adf.desktopintegration.servlet.DIRemoteServlet

- In Servlets page, click the **Servlet Mappings** tab, and then click the **Add** icon to create a row in the Servlet Mapping table.

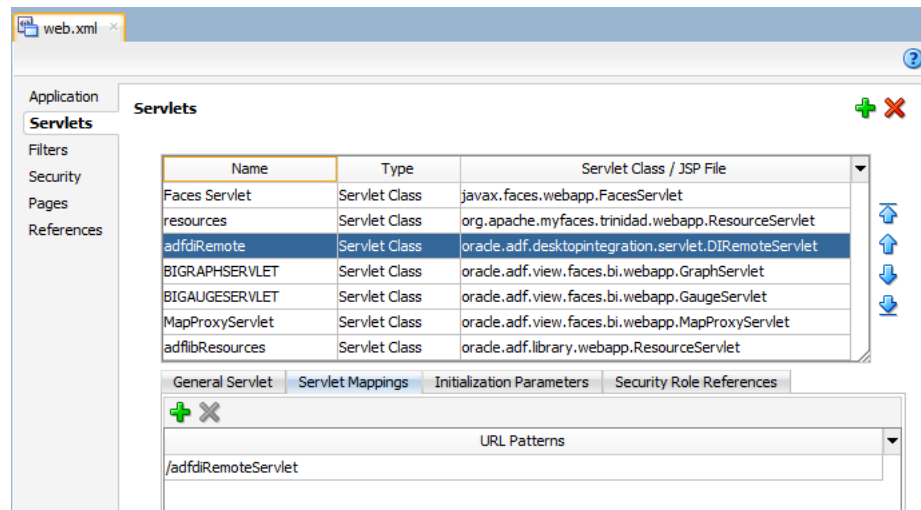
Enter the value as described in [Table D-2](#) to add a URL pattern for the adfdiRemote servlet in the Fusion web application. The value that you enter must match the value that you specify in the integrated Excel workbook for the RemoteServletPath workbook property. Note that values are case sensitive.

Table D-2 Values to Add A URL Pattern to adfdiRemote Servlet

For this property...	Enter this value...
URL Patterns	/adfdiRemoteServlet

[Figure D-1](#) displays the Servlets page of web.xml of Summit sample application for ADF Desktop Integration.

Figure D-1 Servlets Page of Deployment Descriptor



- Click the **Filters** page, and verify that whether a adfBindings filter exists in the Filters table. If an entry exists, select it and proceed to the next step. If there is no such entry, then click the **Add** icon to create a row entry in the Filters table. The icon is available in the top-right corner of the filters table.

Enter the values as described in [Table D-3](#) to add the ADF binding filter to the adfdiRemote servlet.

Table D-3 Values to Add Binding Filter to adfdiRemote Servlet

For this property...	Enter this value...
Name	adfBindings

Table D–3 (Cont.) Values to Add Binding Filter to adfdiRemote Servlet

For this property...	Enter this value...
Class	oracle.adf.model.servlet.ADFBindingFilter

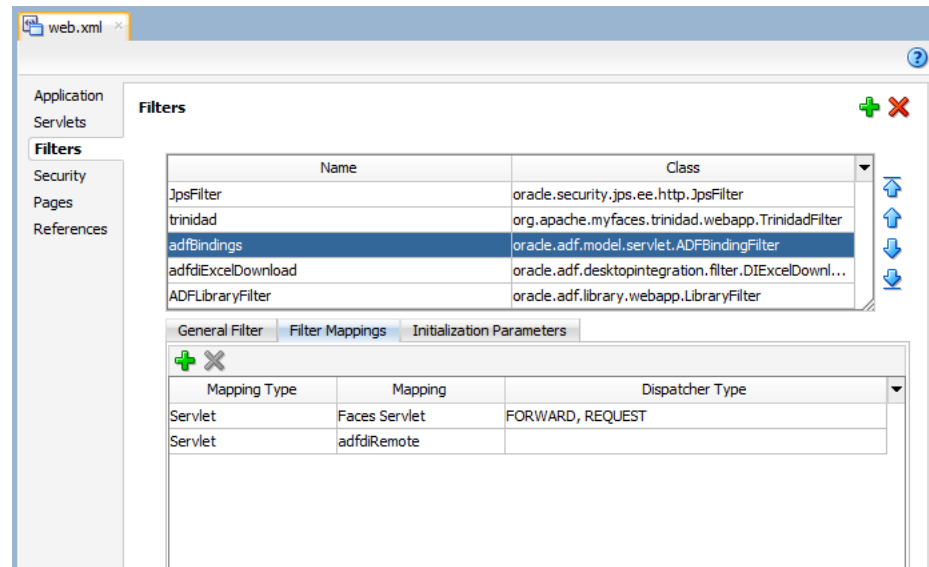
- In Filters page, click the **Filter Mappings** tab, and then click the **Add** icon to create a row in the Filter Mapping table.

Enter the values as described in [Table D–4](#) to add the mapping filter to the adfdiRemote servlet. The filter mapping must match with the Servlet name in Step 2.

Table D–4 Values to Add Mapping Filter to adfdiRemote Servlet

For this property...	Enter this value...
Mapping Type	Servlet
Mapping	adfdiRemote

[Figure D–2](#) displays the Filters page of web.xml of Summit sample application for ADF Desktop Integration.

Figure D–2 Filters Page of Deployment Descriptor

- Save the deployment descriptor file, and then rebuild your ADF Desktop Integration project to apply the changes you made.

D.2 Configuring the ADF Desktop Integration Excel Download Filter

ADF Desktop Integration includes an HTTP filter in the adf-desktop-integration.jar stored in the following directory:

`MW_HOME\oracle_common\modules\oracle.adf.desktopintegration_12.1.3`

where `MW_HOME` is the Middleware Home directory.

You configure an entry in the deployment descriptor file (`web.xml`) of your Fusion web application so that the application invokes the HTTP filter to make changes in an

integrated Excel workbook before the integrated Excel workbook is downloaded by the end user from the Fusion web application. These changes ensure that the integrated Excel workbook functions correctly when the end user opens it. The HTTP filter makes the following changes:

- **WebAppRoot**
Sets the value for this property to the fully qualified URL for the Fusion web application from which the end user downloads the integrated Excel workbook.
- **Workbook mode**
Changes the integrated Excel workbook mode to runtime mode in case the workbook was inadvertently left in design mode or test mode.

By default, JDeveloper adds the HTTP filter to your ADF Desktop Integration project when ADF Desktop Integration is enabled in your project.

To configure the HTTP filter:

1. In JDeveloper, locate and open the deployment descriptor file (*web.xml*) for your ADF Desktop Integration project.

Typically, this file is located in the `WEB-INF` directory of your project.
2. Click the **Filters** page, and verify that an `adfBindings` filter exists in the **Filters** table. If an entry exists, select it and proceed to the next step. If there is no such entry, then click the **Add** icon to create a row entry in the **Filters** table.

Enter the values as described in [Table D-5](#) to create a filter, or configure the values to modify the existing HTTP filter.

Table D-5 Properties to Configure HTTP Filter

For this property...	Enter this value...
Name	<code>adfdiExcelDownload</code>
Class	<code>oracle.adf.desktopintegration.filter.DIExcelDownloadFilter</code>
Display Name	(Optional) In General Filter tab, enter a display name for the filter that appears in JDeveloper.
Description	(Optional) In General Filter tab, enter a description of the filter.

3. In the **Filters** page, click the **Filter Mappings** tab, and then click the **Add** icon to create a row in Filter Mapping table.

Add a filter mapping for integrated Excel workbooks that use the default file format (`.xlsx`) by entering values as described in [Table D-6](#).

Table D-6 Properties to Configure Filter Mappings

For this property...	Enter this value...
Mapping Type	URL Pattern
Mapping	<code>*.xlsx</code>
Dispatcher Type	No value is required for this property.

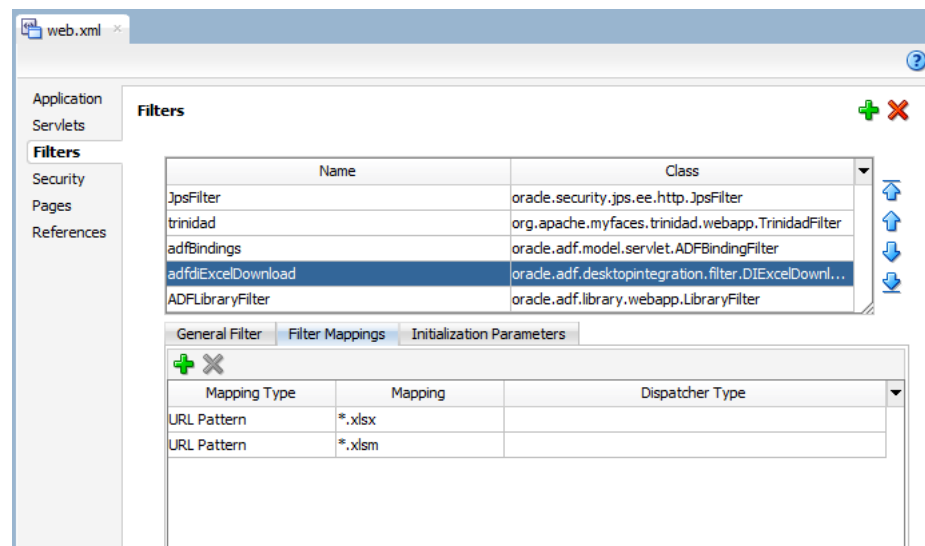
4. Add another filter mapping for integrated Excel workbooks that use the macro-enabled workbook format (`.xlsm`) by entering values as described in [Table D-7](#).

Table D-7 Properties to Configure Filter Mappings

For this property...	Enter this value...
Mapping Type	URL Pattern
Mapping	*.xlsm
Dispatcher Type	No value is required for this property.

Figure D-3 displays the Filters page of web.xml of Summit sample application for ADF Desktop Integration.

Figure D-3 Filters Page of Deployment Descriptor



- Click the Application page, expand MIME Mappings section, and click the Add icon.

Add a MIME type for integrated Excel workbooks that use the default file format (.xlsx) by entering values as described in Table D-8.

Table D-8 Properties to Add MIME Mappings

For this property...	Enter this value...
Extension	*.xlsx
MIME Type	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet

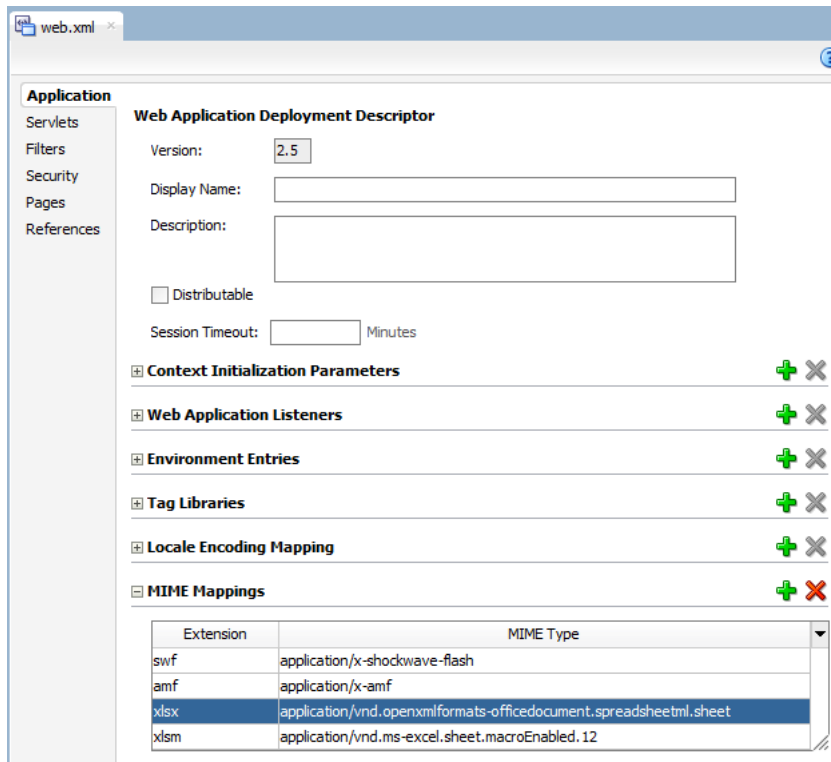
- Add another MIME type for integrated Excel workbooks that use the macro-enabled workbook format (.xlsm) by entering values as described in Table D-9.

Table D-9 Properties to Add MIME Mappings

For this property...	Enter this value...
Extension	*.xlsm
MIME Type	application/vnd.ms-excel.sheet.macroEnabled.12

Figure D–4 displays the Application page of `web.xml` of Summit sample application for ADF Desktop Integration.

Figure D–4 Application Page of Deployment Descriptor



7. Save the deployment descriptor file, and then rebuild your ADF Desktop Integration project to apply the changes you made.

While updating filter and filter mapping information in the `web.xml` file, ensure that the filter for ADF Library Web Application Support (`<filter-name>ADFLibraryFilter</filter-name>`) appears below the `adfdiExcelDownload` filter entries, so that integrated Excel workbooks can be downloaded from the Fusion web application.

D.3 Examples in a Deployment Descriptor File

The following extracts from the `web.xml` file of a Fusion web application with ADF Desktop Integration shows the entries that you configure for a desktop integration project. For more information ordering of filters, see [Section 4.4.4, "What Happens When You Add ADF Desktop Integration to Your JDeveloper Project."](#)

```
<filter>
  <filter-name>adfBindings</filter-name>
  <filter-class>
    oracle.adf.model.servlet.ADFBindingFilter</filter-class>
</filter>
<filter>
  <filter-name>adfdiExcelDownload</filter-name>
  <filter-class>
    oracle.adf.desktopintegration.filter.DIExcelDownloadFilter
  </filter-class>
</filter>
```

```
<filter-mapping>
  <filter-name>adfBindings</filter-name>
  <servlet-name>adfdiRemote</servlet-name>
</filter-mapping>
<filter-mapping>
  <filter-name>adfdiExcelDownload</filter-name>
  <url-pattern>*.xlsx</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>adfdiExcelDownload</filter-name>
  <url-pattern>*.xslm</url-pattern>
</filter-mapping>
<servlet>
  <servlet-name>adfdiRemote</servlet-name>
  <servlet-class>
    oracle.adf.desktopintegration.servlet.DIRemoteServlet
  </servlet-class>
  <init-param>
    <param-name>ClientServerVersionCheck.Precision</param-name>
    <param-value>2</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>adfdiRemote</servlet-name>
  <url-pattern>/adfdiRemoteServlet</url-pattern>
</servlet-mapping>
<mime-mapping>
  <extension>xlsx</extension>
  <mime-type>
    application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
  </mime-type>
</mime-mapping>
<mime-mapping>
  <extension>xslm</extension>
  <mime-type>
    application/vnd.ms-excel.sheet.macroEnabled.12
  </mime-type>
</mime-mapping>
```


String Keys in the Overridable Resources

This appendix describes the string keys in the reserved resource bundle that you can override.

[Table E-1](#) lists the string keys and their current English values. Create a resource bundle where you define the string keys in [Table E-1](#) and the values that you want to appear at runtime. For information about how to override the reserved resource bundle, see [Section 10.2.2, "How to Override Resources That Are Not Configurable."](#)

Table E-1 String Keys and Values in the Reserved Resource Bundle

Area where string key value appears at runtime	String key	English value in the ADF Desktop Integration reserved resource bundle	Comments
Upload Options	UPLOAD_OPTIONS_TITLE	Upload Options	
Upload Options	UPLOAD_OPTIONS_PROMPT	Specify options to use during the Upload operation	
Upload Options	UPLOAD_OPTIONS_CONTINUE_ON_FAIL_LABEL	On failure, continue to upload subsequent rows	
Upload Options	UPLOAD_OPTIONS_DOWNLOAD_AFTER_LABEL	Download all rows after successful upload	
Table.Download	DOWNLOAD_OVERWRITE_TITLE	Download	
Table.Download	DOWNLOAD_OVERWRITE_PROMPT	Do you wish to discard the pending changes?	
Table.Download	ROWLIMIT_WARNINGS_TITLE	Row limit exceeded	
Table.Initialize	INITIALIZE_OVERWRITE_TITLE	Initialize	
Table.Initialize	INITIALIZE_OVERWRITE_PROMPT	Do you wish to discard the pending changes?	
Workbook.ClearAllData	CLEARDATA_CONFIRM_TITLE	Clear all data	
Workbook.ClearAllData	CLEARDATA_CONFIRM_PROMPT	This command will log you out of your current session and clear all the data from all worksheets in the workbook. Are you sure?	
Workbook.Logout	LOGOUT_STATUS_TITLE	Logout	
Workbook.Logout	LOGOUT_STATUS_PROMPT	You have been logged out from your current session.	
Table.Upload	COMPONENTS_TABLE_DYN_COLS_NOT_AVAIL_TITLE	Upload	

Table E-1 (Cont.) String Keys and Values in the Reserved Resource Bundle

Area where string key value appears at runtime	String key	English value in the ADF Desktop Integration reserved resource bundle	Comments
Table.Upload	COMPONENTS_TABLE_DYN_COLS_NOT_AVAIL_PROMPT	One or more dynamic columns is no longer available, do you wish to continue?	
Table status	UPLOAD_STATUS_NO_UPDATES	No updates detected	
Table status	TABLE_UPLOAD_RECORD_NOT_FOUND	Record not found	
Table status	TABLE_UPLOAD_CANNOT_INSERT_MORE_THAN_ONCE	Cannot insert record more than once	
Table status	TABLE_COMMIT_FAILED_1	See Error Detail {0}	{0} is a batch number
Table status	TABLE_COMMIT_FAILURE_DETAILS_2	Error Detail {0};{1}	{0} is a batch number {1} is an error message
Table status	TABLE_UPLOAD_ROW_UPDATE_SUCCESS	Row updated successfully	
Table status	TABLE_UPLOAD_ROW_INSERT_SUCCESS	Row inserted successfully	
Table status	TABLE_UPLOAD_ROW_UPDATE_FAILURE	Update failed	
Table status	TABLE_UPLOAD_ROW_INSERT_FAILURE	Insert failed	
Table status	TABLE_DELETE_ROW_FAILURE	Delete failed	
Table status	TABLE_ROW_KEY_VALUE_INVALID	Key value invalid	
Table status	TABLE_DOWNLOAD_FAILURE	Download failed	
Table status	TABLE_DOWNLOAD_ROW_FAILURE	Row download failed	
Table status	TABLE_DOWNLOAD_FLAGGED_FAILURE	Download flagged rows failed	
Table status	TABLE_DOWNLOAD_FOR_INSERT_FAILURE	Download for insert failed	
Table status	MESSAGE_DETAILS_NONE	No error details available	
Table status	MESSAGE_DETAILS_ROW_TITLE	Row Errors	
Table status	MESSAGE_DETAILS_ROW_PROMPT	Errors for this row:	
Table status	MESSAGE_DETAILS_TABLE_TITLE	Table Errors	
Table status	MESSAGE_DETAILS_TABLE_PROMPT	Error details for this table:	
Table status Table errors Worksheet errors	MESSAGE_DETAILS_HELP_LABEL	Click on each error to reveal additional information	Appears in the error list.
Table status Table errors Worksheet errors	MESSAGE_LABEL_DEFAULT_CONTEXT	Action	
Worksheet errors	MESSAGE_DETAILS_WORKSHEET_TITLE	Worksheet Errors	
Worksheet errors	MESSAGE_DETAILS_WORKSHEET_PROMPT	Error details for this worksheet:	

Table E-1 (Cont.) String Keys and Values in the Reserved Resource Bundle

Area where string key value appears at runtime	String key	English value in the ADF Desktop Integration reserved resource bundle	Comments
Worksheet errors	MESSAGE_DETAILS_PARSE_FAILURE	A problem has occurred while retrieving the error details. The information is no longer available.	
Worksheet errors	MESSAGE_LABEL_FAILED_1	{0} failed	{0} is a context label
Workbook.EditOptions	SETTINGS_EDIT_TITLE	Edit Options	
Workbook.EditOptions	SETTINGS_EDIT_PROMPT	Enter a value for WebAppRoot. For example: http://localhost:1234/MyApp.	
Workbook.EditOptions	SETTINGS_CONFIRM_TITLE	Web App Root	
Workbook.EditOptions	SETTINGS_CONFIRM_PROMPT	Changing the Web App Root will log you out of your current session and clear all the data from all worksheets in the workbook. Are you sure?	

Note: The keys listed in [Table E-1](#) cannot be used in EL expressions of the following syntax:

```
#{_ADFDIRes['key']}
```

Java Data Types Supported By ADF Desktop Integration

This appendix lists the Java data types that an ADF Desktop Integration project supports.

Primitive Java Types

- `boolean`
- `double`
- `float`
- `int`
- `long`
- `short`

Object Java Types

- `java.lang.Boolean`
- `java.lang.Double`
- `java.lang.Float`
- `java.lang.Integer`
- `java.lang.Long`
- `java.lang.Short`
- `java.lang.String`
- `java.math.BigDecimal`
- `java.sql.Date`
- `java.sql.Time`
- `java.sql.Timestamp`
- `java.util.Date`
- `oracle.jbo.domain.Date`
- `oracle.jbo.domain.Number`
- `oracle.jbo.domain.RowID`
- `oracle.jbo.domain.Timestamp`
- `oracle.jbo.domain.TimestampLTZ`

-
- `oracle.jbo.domain.TimestampTZ`

Note: Using data types not listed in this appendix will generate error at runtime.

Using the ADF Desktop Integration Model API

There may be certain use cases where you want to allow uploading ADF Table data even when there are no rows available in a tree binding. This appendix describes how to use the ADF Desktop Integration Model API library in custom Java code to access the attribute values sent from the client during the upload process when there are no actual rows available.

This appendix includes the following sections:

- [Section G.1, "About the Temporary Row Object"](#)
- [Section G.2, "About ADF Desktop Integration Model API"](#)
- [Section G.3, "ADF Desktop Integration Model API Classes and Methods"](#)

G.1 About the Temporary Row Object

Each ADF Table component is bound to a tree binding defined within a page definition. Each tree control binding has one (or more) tree nodes defined. For parent-child relationships, the tree binding has two nodes, one for parent table and another for child table. At runtime, the ADF Table component displays both parent and child attributes within each worksheet row. On upload, ADF Desktop Integration sets attribute values to both the parent and child nodes.

In certain situations, a particular tree node may not have actual data rows available during `Table.Upload` request processing. Two common scenarios where a tree node may not have data are:

- The tree node's iterator result set does not have any data rows available. This could be because of a query returning zero rows.
- In a parent-child relationship, if the foreign key has not been populated in the parent table, the link between parent and child tree node may not contain actual rows.

There may be certain cases when, even though there is no actual row available on the server, you still want to allow the end user to enter values in the worksheet and upload them to the server. During upload, ADF Desktop Integration creates a temporary row object and stores the values uploaded from the worksheet row. Using the ADF Desktop Integration Model API, you can write custom Java code to access the temporary row object and collect its values.

To call your custom Java code during upload, you must expose your custom Java code through a `pageDef` action binding and then configure the ADF Table component's

UpdateRowActionID or InsertAfterRowActionID to point to the pageDef action binding.

G.2 About ADF Desktop Integration Model API

While data is being uploaded, if a tree node of the ADF Table component contains no actual rows, the ADF Desktop Integration remote servlet creates a temporary row object to store the attribute values. If you want to access the temporary row object and its attribute values, you must write custom Java code that uses the ADF Desktop Integration Model API library.

Note: The ADF Desktop Integration Model API is not supported for EJB or Toplink data controls.

For more information about the classes and methods available in the API, see [Section G.3, "ADF Desktop Integration Model API Classes and Methods."](#)

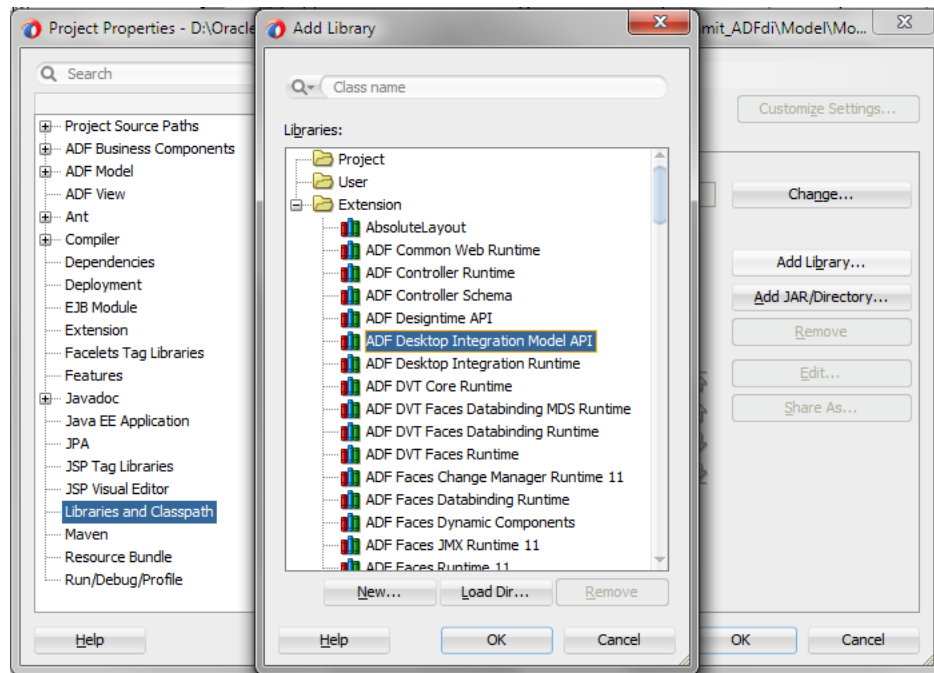
G.2.1 How to Add ADF Desktop Integration Model API Library to Your JDeveloper Project

You typically add the ADF Desktop Integration Model API Library to your application's data model project. The library is an independent library, not included with any feature. You can add it through Project Properties dialog box.

To add ADF Desktop Integration Model API library to your project:

1. In the Applications window, right-click the data model project and choose **Project Properties**.
2. In the Project Properties dialog, select **Libraries and Classpath** to view the list of libraries available.
3. Click **Add Library** and in the Add Library dialog, select the **ADF Desktop Integration Model API** library.

Figure G-1 Add Library Dialog



4. Click **OK**. The library name adds to the **Classpath Entries** list.
5. Click **OK** to close the Project Properties dialog box.

G.3 ADF Desktop Integration Model API Classes and Methods

The ADF Desktop Integration Model API library contains one public class that contains APIs for retrieving temporary row objects.

G.3.1 The `oracle.adf.desktopintegration.model.ModelHelper` Class

The `ModelHelper` class is a public class that exposes Model APIs. The following sections describe the methods available in the class.

G.3.1.1 The `getAdfdiTempChildRow` Method

The method is used to lookup temporary child row object (`ViewRowImpl` object) associated with a particular master row. When required, the servlet code creates the temporary `ViewRowImpl` object and stores attribute values when there are no actual `ViewRowImpl` objects available.

The method returns the temporary child `ViewRowImpl` object containing any attribute values sent from worksheet.

Method Syntax

```
public static final ViewRowImpl getAdfdiTempChildRow(ViewRowImpl masterRow,
java.lang.String childAccessor)
```

Parameters

- `masterRow` – master row object
- `childAccessor` – child attribute name

G.3.1.2 The `getAdfdiTempRowForView` Method

The method is used to lookup temporary child row object (`ViewRowImpl` object) associated with a particular view. When required, the servlet code creates the temporary `ViewRowImpl` object and stores attribute values when there are no actual `ViewRowImpl` objects available.

The method returns the temporary child `ViewRowImpl` object containing any attribute values sent from worksheet.

Method Syntax

```
public static final ViewRowImpl getAdfdiTempRowForView(ApplicationModuleImpl  
am, java.lang.String viewDefName)
```

Parameters

- `am` – application module instance
- `viewDefName` – view definition name

G.3.1.3 The `getChildViewDef` Method

The method is used to lookup polymorphic child view definition if the view link destination attributes specify one or more child discriminator attributes. The master row source attributes lookup the correct polymorphic child view definition through `ViewObjectImpl.findViewDefFromDiscrValues` API. If no child discriminator attributes are defined, or the child view is non-polymorphic, the default child `ViewDefImpl` object is returned.

The method returns the temporary child `ViewRowImpl` object containing any attribute values sent from worksheet, or returns null if the object is not found.

Method Syntax

```
public static final ViewDefImpl getChildViewDef(ViewRowImpl  
masterRow, java.lang.String childAccessor)
```

Parameters

- `masterRow` – master row object
- `childAccessor` – child attribute name

End User Actions

This appendix describes the actions the end user would be performing while using your application and integrated Excel workbook, such as installing runtime edition of ADF Desktop Integration, importing data from non-integrated Excel workbook, making changes in the workbook at runtime, and handling time zone conversion of date-time values in the workbook.

The actions described in this appendix assume that you have developed a functioning Fusion web application.

This appendix includes the following sections:

- [Section H.1, "Installing, Upgrading, and Removing the Runtime Edition of ADF Desktop Integration"](#)
- [Section H.2, "Importing Data from a Non-Integrated Excel Worksheet"](#)
- [Section H.3, "Removing Personal Information"](#)
- [Section H.4, "Editing an Integrated Excel Workbook at Runtime"](#)
- [Section H.5, "Limitations of an Integrated Excel Workbook at Runtime"](#)
- [Section H.6, "Using An Integrated Excel Workbook"](#)
- [Section H.7, "Handling Time Zone Conversion"](#)

H.1 Installing, Upgrading, and Removing the Runtime Edition of ADF Desktop Integration

End users can install the runtime edition of ADF Desktop Integration using the client installer.

When the ADF Desktop Integration client installer runs, it verifies whether the required software is installed on the system. For more information about the required software, see the following:

- [Section 3.2, "Required Oracle ADF Modules and Third-Party Software"](#)
- [Section 3.3, "Configuring Excel to work with ADF Desktop Integration"](#)

Notes:

- You cannot install the Runtime edition of ADF Desktop Integration from JDeveloper.
 - JDeveloper is not required to install the runtime edition of ADF Desktop Integration.
-
-

H.1.1 How to Install Runtime Edition of ADF Desktop Integration

To install the runtime edition, the client installer must be made available to the end user on the end user's local system. Ask your system administrator to make the client installer files available to the end user.

To install the Runtime edition of ADF Desktop Integration:

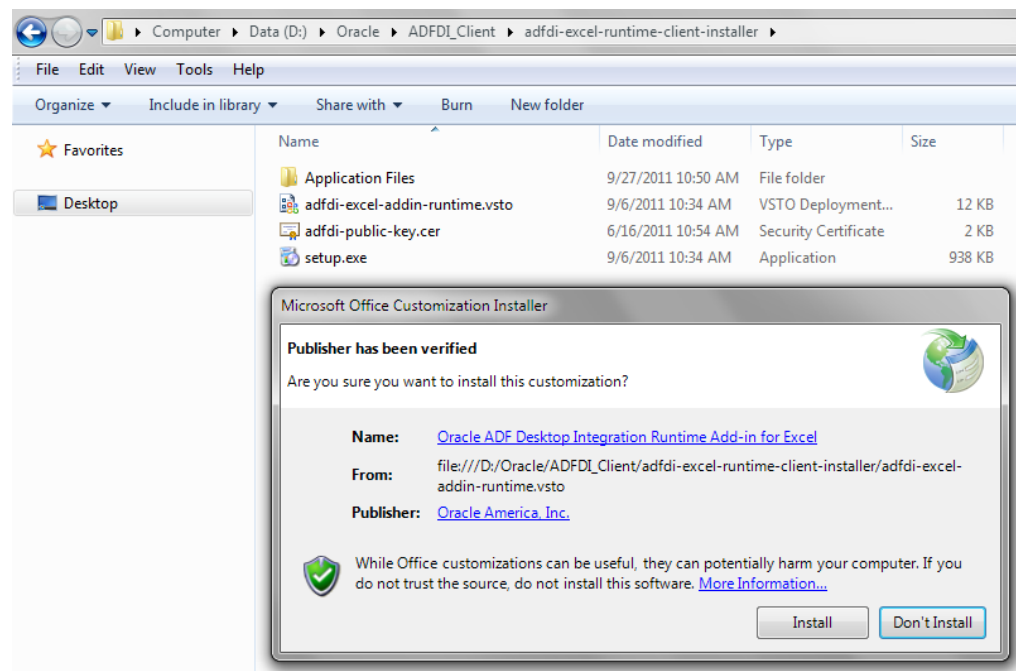
1. Navigate to the `MW_HOME\oracle_common\modules\oracle.adf.desktopintegration_12.1.3` directory, where `MW_HOME` is the Middleware Home directory.
2. Extract the contents of `adfdi-excel-runtime-client-installer.zip` to a temporary directory on the end user's local file system (for example, `D:\Oracle\ADFDI_Client`).

Note that the fully qualified path of the temporary directory must be less than 248 characters.

3. Run the `setup.exe` file located in the extracted directory of the `adfdi-excel-runtime-client-installer.zip` file.

Figure H-1 shows an example of the `setup.exe` file location on a local system, and the installation of runtime edition.

Figure H-1 Installing Runtime Edition of ADF Desktop Integration



4. Follow the instructions that appear in the dialog boxes launched by `setup.exe` to successfully install the required components.
5. If prompted, click **Yes** to restart the system and complete the setup of ADF Desktop Integration.

WARNING: After you install the runtime edition of ADF Desktop Integration, do not delete the directory where you copied the client installer files. You can delete the files after removing the runtime edition of ADF Desktop Integration from the system.

H.1.2 How to Remove the Runtime Edition of ADF Desktop Integration

Use the Microsoft Windows Control Panel to remove the runtime edition of ADF Desktop Integration from the system.

To remove the ADF Desktop Integration add-in:

1. Click the Windows **Start** button, and then choose **Control Panel**.
2. In the Control Panel, select and open **Programs and Features**.
3. Select the Oracle ADF Desktop Integration Runtime add-in for Excel entry in the Uninstall or change a program window, and click **Uninstall**.

H.1.3 How to Upgrade the Runtime Edition of ADF Desktop Integration On a Local System

To upgrade the runtime edition of ADF Desktop Integration, uninstall the old client version, and install the new version. For more information about uninstalling the runtime edition, see [Section H.1.2, "How to Remove the Runtime Edition of ADF Desktop Integration,"](#) and for more information about installing the runtime edition, see [Section H.1.1, "How to Install Runtime Edition of ADF Desktop Integration."](#)

If the runtime edition of ADF Desktop Integration is installed from a shared file server or a web server, by default, Excel checks the server for ADF Desktop Integration updates when it is launched and if 24 hours have elapsed since the last check for updates. If you do not wish to wait 24 hours and want to check the update manually, you can do so from the Microsoft Windows Control Panel.

To upgrade the Runtime edition of ADF Desktop Integration manually:

1. Click the Windows Start button, and then choose **Control Panel**.
2. In the Control Panel, select and open **Programs and Features**.
3. Select the **Oracle ADF Desktop Integration Runtime Add-in for Excel** entry in the Uninstall or change a program window.
4. With the entry selected, click the **Update Information** link.

Note: If the runtime edition of ADF Desktop Integration is installed from a web server, open the **Update Information** link in Internet Explorer to run the installer. Other browsers are not supported.

5. Follow the instructions that appear in the dialog boxes to successfully upgrade ADF Desktop Integration.

H.2 Importing Data from a Non-Integrated Excel Worksheet

End users who use the ADF Table component in an integrated Excel workbook to upload large batches of data rows to the Fusion web application can prepare these

rows of data in a non-integrated Excel worksheet. They can then insert the data into the ADF Table component prior to invoking the ADF Table component's Upload action.

To prepare data in a non-integrated Excel workbook:

1. Arrange the layout of data in a non-integrated Excel worksheet to match the layout of the ADF Table component in the integrated Excel workbook.

For example, if an ADF Table component contains columns such as Product, Price, and Description, reproduce this layout in the non-integrated Excel worksheet.

Tip: Copy the column headers from the ADF Table component to the non-integrated Excel worksheet.

2. Use functionality of Excel to import the rows of data into the non-integrated Excel worksheet in rows under the columns arranged in Step 1.
3. Row values that will be inserted into ADF Table component columns that use the TreeNodeList subcomponent must match a choice from the list of values.

Tip: Copy an ADF Table component row from the integrated Excel workbook to another worksheet of the same workbook, as the proper constraints will be defined for such a row and can be reproduced.

To insert data into the ADF Table component from a non-integrated Excel workbook:

1. In the ADF Table component, highlight n existing downloaded rows or new rows at the end of the ADF Table component where n is the number of rows to insert.
2. Right-click and choose **Insert** from the context menu.
3. In the non-integrated Excel worksheet, select the cells to insert into the rows of the ADF Table component created in Step 2.

WARNING: Select the cells in the non-integrated Excel worksheet and not the rows or columns.

4. In the Excel Ribbon, choose **Home > Copy**.
5. In the ADF Table component, select the upper left corner cell of the rows inserted in Step 2.
6. In the Excel Ribbon, choose **Home > Paste**.
7. End users can now invoke the ADF Table component's Upload action using whatever functionality you configured for them as described in [Section 7.8, "Configuring an ADF Component to Upload Changes from an ADF Table Component."](#)

H.3 Removing Personal Information

If the Fusion web application that you integrate an Excel workbook with uses a security mechanism, such as single sign-on, personally identifying information may be stored in cookies on the system where the end user accesses the integrated Excel workbook. End users can remove this information using Microsoft Internet Explorer. End users must log out and close all integrated Excel workbooks to invalidate all active cookie-based web sessions.

For information about removing personal information, see Microsoft Internet Explorer documentation.

H.4 Editing an Integrated Excel Workbook at Runtime

Once you publish and deploy a finalized integrated Excel workbook, as described in [Chapter 14, "Deploying Your Integrated Excel Workbook."](#) end users can make the following changes to a workbook at runtime:

- Delete a column from an ADF Table or ADF Read-only Table component.
- Drag and drop cells to move ADF components other than an ADF Button component.
- Insert new rows into an ADF Table component.
- Change the order of columns in an ADF Table or ADF Read-only Table component.
- Insert non-integrated columns between the columns of an ADF Table or ADF Read-only Table component.

However, some changes to a workbook at runtime can corrupt the integration and are not supported. For more information about what changes are not allowed at runtime, see [Section H.5, "Limitations of an Integrated Excel Workbook at Runtime."](#)

H.5 Limitations of an Integrated Excel Workbook at Runtime

There are some known limitations on changing ADF Desktop Integration components at runtime.

- **Moving a column in ADF Table and ADF Read-only Table components** – If the end user moves a column (for example, using native Excel cut-insert operations) to the right of the last table column, the column is considered to be outside of the table boundaries. ADF Desktop Integration will no longer recognize the column as being part of the table component during subsequent table operations.

Instead, to change the last column of a table component at runtime, the end user can perform either of the following actions:

- Move all columns that are to the right of the desired last column to the left of the desired last column.
- First, move the desired last column to the immediate left of the current last column, and then move the current last column to the left of the desired last column.

Similar actions should be performed to move a column so that it becomes the left-most column of the table.

- **Deleting an integrated Excel worksheet** – If the end user deletes an integrated Excel worksheet, ADF Desktop Integration generates an exception when the end user tries to save the integrated Excel workbook.

To resolve the problem, the end user must close and reopen the workbook without saving changes.

Additional known limitations:

- Excel's Conditional Formatting feature cannot be used effectively with ADF Desktop Integration table components.

- The ADF Button components are disabled when the end user zooms in or out on an integrated Excel worksheet. The ADF Button components are active at 100% zoom only.
- Due to native Excel behavior, sorting a dependent List of Values column at runtime does not sort the cell list validations properly.

H.6 Using An Integrated Excel Workbook

End users who are new to the ADF Desktop Integration technology and integrated Excel workbook must be made aware of the following common actions:

- To download all rows after uploading the changed data, ensure that **Download all rows after successful upload** checkbox is selected in Upload Options dialog box.
- Before uploading the changes, ensure that the **Changed** column of all modified rows is marked with an upward pointing triangle. A double-click on the upward pointing triangle character removes it, and the data of the relevant row is not uploaded.
- To have Excel retain the format of a numeric or date value in a cell formatted with a text style while uploading data, add an apostrophe symbol (') before entering the value. The apostrophe symbol acts as an escape character and is not displayed with the value.
- Do not delete, edit, or clear any cells in the **Key** column of the table. Any change to these values can lead to upload failures and data corruption.
- Do not change Excel's settings for Protect Sheet or Protect Workbook. These settings are available in the Changes group of the **Review** tab.
- To erase a value from a cell that is integrated with the web application, clear the cell value instead of deleting the Excel cell.
- If the Fusion web application is running on the `https` protocol and you have not installed the security certificate on the client, the integrated Excel workbook gives an error on login and the connection is not established. To establish a connection, you must install the security certificate. If you cannot install the certificate from Excel, open Internet Explorer and navigate to the same website. You will be prompted to install the certificate.
- Some ADF components may have cells that are configured to respond to a double-click to perform some action. For example, the Status column cells of the ADF Table component. You can also right-click in these cells and select **Invoke Action**.

Some common actions, such as inserting or deleting a row, and sorting data in ADF Table, are described in the subsequent sections.

H.6.1 How to Insert a Row in an ADF Table Component of an Integrated Excel Workbook

To insert a row in an ADF Table component, insert a full row in the worksheet, and add data in all mandatory columns. For more information, see [Section 7.7, "Configuring an ADF Table Component to Insert Data."](#)

An ADF Table, with one or more data rows, does not have a placeholder row after the last row of the table. Hence, any data inserted in the cells of the workbook after the last row of the table is not uploaded. To upload data of the row after the last row, you must insert a full Excel row.

To insert a row in an ADF Table component:

1. In the ADF Table component, click the row header, and select the entire row above which you want to insert the new row.
2. With the row selected, right-click and choose **Insert**.

A new row is inserted above the selected row.

Notes:

- Insert a full Excel row between the table headers and the last row of the table. Do not insert the new row after the last row of the table.
 - If the ADF Table has no data rows, the first row of the table acts as a placeholder row.
-
-

H.6.2 How to Sort ADF Table Data in an Integrated Excel Workbook

To sort table data, choose Excel's Sort and Filter command.

To sort ADF Table data based on a particular column:

1. Select the header, or any cell, of the column you want to sort.
2. In the Editing group of the **Home** tab, click **Sort and Filter**. Choose the desired sort order from the dropdown list options.

To sort table data based on multiple columns:

1. Select any cell of the table.
2. In the Editing group of the **Home** tab, click **Sort and Filter**, and choose **Custom Sort**.
3. In the Sort dialog, add the columns, and their order preference. Ensure that the **My data has headers** checkbox is enabled.
4. Click **OK**.

Note: While sorting the columns in an ADF Table component, ensure that you always choose **Expand the selection** in the Sort Warning dialog, when prompted, in order to maintain the integrity of the data in all the table rows.

H.6.3 How to Delete a Row in ADF Table of an Integrated Excel Workbook

Clearing the cell values of a row does not remove the row, and deleting the row from the Excel worksheet does not delete the row from the web application.

To delete a row in an ADF Table component, flag the row by double-clicking the respective cell of the **Flagged** column, and click the respective delete button. For more information about row flagging, see [Section 7.11.2, "Row Flagging in an ADF Table Component."](#)

Note: If your table does not contain a **Flagged** column, you will not be able to delete rows from that table.

H.7 Handling Time Zone Conversion

You can configure integrated Excel workbooks to retrieve, edit, and submit data values that represent dates and times. As Excel does not provide native support for managing date or time data when the system time zone changes, ADF Desktop Integration tracks and detects the time zone changes for a workbook. It informs the end user about the time zone update when the workbook is opened, and then converts the date-time data of the workbook to the current time zone setting of the system.

For example, an end user in Arizona (GMT -07:00) downloads the data from the server to the integrated Excel workbook, edits the date-time data in the workbook, saves the data, but does not upload it. Later, the end user travels to Seoul and changes the time zone preference of the computer to GMT +09:00. When the workbook is opened after changing to the Seoul time zone, the end user receives a message, and then all date-time data values in the ADF components are converted from GMT -07:00 (Arizona) to GMT +09:00 (Seoul).