

Oracle® Fusion Middleware

Developing Applications with Oracle ADF Desktop Integration



12c (12.2.1.4.0)

E80012-01

September 2019

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

Oracle Fusion Middleware Developing Applications with Oracle ADF Desktop Integration, 12c (12.2.1.4.0)

E80012-01

Copyright © 2014, 2019, Oracle and/or its affiliates. All rights reserved.

Primary Author: Walter Egan

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, then 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 about 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 unless otherwise set forth in an applicable agreement between you and Oracle. 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, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience	xvii
Documentation Accessibility	xvii
Related Documents	xvii
Conventions	xvii

What's New in This Guide for 12c (12.2.1.4.0)

New and Changed Features for 12c (12.2.1.4.0)	xix
---	-----

1 Introduction to ADF Desktop Integration

About ADF Desktop Integration	1-1
About ADF Desktop Integration with Microsoft Excel	1-2
Overview of Creating an Integrated Excel Workbook	1-2
Advantages of Integrating Excel with a Fusion Web Application	1-3

2 Introduction to the ADF Desktop Integration Sample Application

About the Summit Sample Application for ADF Desktop Integration	2-1
Setting Up and Running the Summit Sample Application for ADF Desktop Integration	2-1
How to Download the Application Resources	2-2
How to Run the Summit Sample Application for ADF Desktop Integration	2-2
Overview of the Fusion Web Application in the Summit Sample Application for ADF Desktop Integration	2-3
About the Fusion Web Application in the Summit Sample Application for ADF Desktop Integration	2-3
Downloading Integrated Excel Workbooks	2-4
Overview of the Integrated Excel Workbooks in the Summit Sample Application for ADF Desktop Integration	2-5
Log on to the Fusion Web Application from an Integrated Excel Workbook	2-5
Downloading Data Rows	2-5
Modify Customers and Warehouses Information in the Workbooks	2-6

Upload Modified Information to the Fusion Web Application	2-6
---	-----

3 Setting Up Your Development Environment

About Setting Up Your Development Environment	3-1
Required Oracle ADF Modules and Third-Party Software	3-1
Installing ADF Desktop Integration	3-2
How to Install ADF Desktop Integration	3-3
Removing ADF Desktop Integration	3-5
Upgrading ADF Desktop Integration	3-5

4 Preparing Your Integrated Excel Workbook

About Preparing Your Integrated Excel Workbooks	4-1
Working with Page Definition Files for an Integrated Excel Workbook	4-1
How to Create ADF Desktop Integration Page Definition File	4-2
What Happens When You Create a Page Definition File	4-4
How to Reload a Page Definition File in an Excel Workbook	4-4
What You May Need to Know About Page Definition Files in an Integrated Excel Workbook	4-5
Adding an Integrated Excel Workbook to a Fusion Web Application	4-6
How to Add an Integrated Excel Workbook to a Fusion Web Application	4-6
How to Configure a New Integrated Excel Workbook	4-8
How to Add Additional Worksheets to an Integrated Excel Workbook	4-12
Enabling ADF Desktop Integration in an Excel Workbook	4-12
How to Enable ADF Desktop Integration in an Existing Workbook	4-12
How to Manually Configure a New Integrated Excel Workbook	4-13
Enabling ADF Desktop Integration Manually	4-15
How to Manually Add ADF Desktop Integration In Fusion Web Application	4-15
What Happens When You Add ADF Desktop Integration to Your JDeveloper Project	4-17
Adding ADF Library Web Application Support	4-17
Using an Integrated Excel Workbook with Older Versions of ADF Desktop Integration	4-18

5 Getting Started with the Development Tools

About Development Tools	5-1
ADF Desktop Integration Development Tools Use Cases and Examples	5-2
Additional Functionality for ADF Desktop Integration Development Tools	5-3
Designer Ribbon Tab	5-3
ADF Desktop Integration Designer Task Pane	5-7

Using the Bindings Palette	5-8
Using the Components Palette	5-9
Using the Property Inspector	5-10
Using the Binding ID Picker	5-11
Using the Expression Builder	5-12
Using the Web Page Picker	5-13
Using the File System Folder Picker	5-14
Using the Page Definition Picker	5-15
Using the Collection Editors	5-16
Using the Cell Context Menu	5-17
Removing ADF Desktop Integration Components	5-18
Exporting and Importing Excel Workbook Integration Metadata	5-19
How to Export Workbook Integration Metadata	5-20
How to Import Workbook Integration Metadata	5-21
What You May Need to Know About Exporting and Importing Excel Workbook Integration Metadata	5-22

6 Working with ADF Desktop Integration Form-Type Components

About ADF Desktop Integration Form-Type Components	6-1
ADF Desktop Integration Form-Type Components Use Cases and Examples	6-2
Additional Functionality for ADF Desktop Integration Form-Type Components	6-3
Inserting an ADF Label Component	6-3
Inserting an ADF Input Text Component	6-5
Inserting an ADF Output Text Component	6-7
Inserting an ADF Input Date Component	6-9
Inserting an ADF Image Component	6-11
Displaying Output from a Managed Bean in an ADF Component	6-13
How to Display Output from a Managed Bean	6-14
What Happens at Runtime: How an ADF Component Displays Output from a Managed Bean	6-15
Displaying Concatenated or Calculated Data in Components	6-15
How to Configure a Component to Display Calculated Data	6-15
Using Form Components and Merged Cells	6-17

7 Working with ADF Desktop Integration Table-Type Components

About ADF Desktop Integration Table-Type Components	7-2
ADF Desktop Integration Table-Type Components Use Cases and Examples	7-2
Additional Functionality of Table-Type Components	7-3
Page Definition Requirements for an ADF Table Component	7-3
Inserting an ADF Table Component into an Excel Worksheet	7-5

How to Insert an ADF Table Component	7-5
How to Add a Column in an ADF Table Component	7-8
Downloading Data to an ADF Table Component	7-9
How to Download Data to an ADF Table Component	7-9
What Happens at Runtime: How an ADF Table Component Downloads Data	7-11
Downloading Pending Insert and Pending Update Rows to an ADF Table Component	7-12
What Happens at Runtime: Download Action is Invoked	7-13
Using STATUS_INITIALIZED Rows for Pending Inserts	7-13
What You May Need to Know About DownloadForInsert Action	7-13
Updating Existing Data in an ADF Table Component	7-13
How to Configure an ADF Table Component to Update Data	7-14
What Happens at Runtime: How the ADF Table Component Updates Data	7-15
Inserting Data in an ADF Table Component	7-15
How to Configure an ADF Table Component to Insert Data Using a View Object's Operations	7-15
Uploading Changes from an ADF Table Component	7-18
How to Configure an ADF Component to Upload Data from an ADF Table Component	7-18
What Happens at Runtime: How the ADF Table Component Uploads Data	7-20
What Happens at Runtime: How the ReadOnly EL Expression Is Evaluated During Upload	7-21
What Happens at Runtime: How Row Errors Are Handled During Upload	7-21
What You May Need to Know About Upload Options	7-22
How to Create a Custom Upload Dialog	7-23
What Happens at Runtime: Custom Upload Dialog	7-24
Uploading Changes from an ADF Table Component Using an UploadAllOrNothing Action	7-25
How to Configure an ADF Component to use UploadAllOrNothing Action	7-25
What Happens at Runtime: UploadAllOrNothing Action is Invoked	7-26
Limiting the Amount of Changed Data That Can Be Uploaded With UploadAllOrNothing Action	7-26
Deleting ADF Table Component Rows in the Fusion Web Application	7-27
How to Configure an ADF Table Component to Delete Rows in the Fusion Web Application	7-28
What Happens at Runtime: How the ADF Table Component Deletes Rows in a Fusion Web Application	7-29
Batch Processing in an ADF Table Component	7-30
How to Configure Batch Options for an ADF Table Component	7-30
Troubleshooting Errors While Uploading Data	7-31
Special Columns in the ADF Table Component	7-32
Row Flagging in an ADF Table Component	7-34
Configuring ADF Table Component Key Column	7-35

How to Configure the Key Column	7-35
How to Manually Add the Key Column At Design Time	7-36
Adding a Dynamic Column to Your ADF Table Component	7-37
How to Configure a Dynamic Column	7-38
What Happens at Runtime: How Data Is Downloaded or Uploaded In a Dynamic Column	7-39
How to Specify Header Labels for Dynamic Columns	7-39
How to Specify Styles for Dynamic Columns	7-40
Adding a ModelDrivenColumnComponent Subcomponent to Your ADF Table Component	7-41
Configuring an ADF Table Component to Resize Columns Based on Data at Runtime	7-41
How to Configure an ADF Table Component to Resize Columns at Runtime	7-42
How to Configure an Action Set to Resize Columns of an ADF Table Component at Runtime	7-44
What Happens at Runtime: How the ADF Table Columns are Resized	7-45
What You May Need to Know About Resizing Columns of an ADF Table Component at Runtime	7-45
Grouping Columns Together in an ADF Table Component	7-46
How to Group Columns in an ADF Table Component	7-47
How to Group Columns that Render in a Dynamic Column	7-50
What Happens at Runtime: How an ADF Table Component Groups Columns	7-52
Configuring an ADF Table Component to be Read-only	7-52
How to Configure an ADF Table Component to be Read-only	7-53
Creating an ADF Read-Only Table Component	7-54
How to Insert an ADF Read-only Table Component	7-54
Limiting the Number of Rows Your Table-Type Component Downloads	7-55
How to Limit the Number of Rows a Component Downloads	7-56
What Happens at Runtime: How the RowLimit Property Works	7-57
Tracking Changes in an ADF Table Component	7-57
Evaluating EL Expressions for ReadOnly Properties	7-58
What Happens at Runtime: Evaluating EL Expression While Downloading Data	7-58
What Happens at Runtime: Evaluating EL Expression While Uploading Data or Tracking Changes	7-58
What You May Need to Know About Evaluating EL Expression While Uploading Data or Tracking Changes	7-59
Displaying Error Messages in an Error Message Column	7-59
How to Display Error Messages in an Error Message Column	7-60

8 Working with Lists of Values

About List of Values in an Integrated Excel Workbook	8-1
--	-----

Adding Lists of Values to Integrated Excel Workbooks Use Cases and Examples	8-1
Additional Functionality for Adding List of Values to an Integrated Excel Workbook	8-2
Creating a List of Values in an Excel Worksheet	8-3
Creating a List of Values in an ADF Table Component Column	8-5
How to Create a List of Values in an ADF Table Component Column	8-5
What Happens at Runtime: How the ADF Table Column Renders a List of Values	8-7
Adding a Model-Driven List Picker to an ADF Table Component	8-8
What You May Need to Know About Model-Driven List Pickers in ADF Table Components	8-10
Creating Dependent Lists of Values in an Integrated Excel Workbook	8-11
How to Create Dependent Lists of Values in Excel Worksheets	8-14
What Happens at Runtime: How an Excel Worksheet Renders a Dependent List of Values	8-15
How to Create Dependent Lists of Values in ADF Table Component Columns	8-16
What Happens at Runtime: ADF Table Component Column Renders a Dependent List of Values	8-17

9 Adding Interactivity to Your Integrated Excel Workbook

About Adding Interactivity to an Integrated Excel Workbook	9-1
Adding Interactivity to Integrated Excel Workbook Use Cases and Examples	9-2
Additional Functionality for Adding Interactivity to an Integrated Excel Workbook	9-3
Using Action Sets	9-3
How to Invoke a Method Action Binding in an Action Set	9-5
How to Invoke Component Actions in an Action Set	9-6
What You May Need to Know About an Action Set Invoking a Component Action	9-8
How to Invoke an Action Set from a Worksheet Event	9-9
How to Display a Progress Bar while an Action Set Executes	9-10
What Happens at Runtime: How the Action Set Displays a Status Message	9-13
What You May Need to Know About Progress Bars	9-13
How to Allow End Users to Continue Working in Excel While an ActionSet Executes	9-14
What Happens at Runtime: How End Users Continue Working While an ActionSet Executes	9-15
What You May Need to Know About Canceling an Action	9-16
How to Provide an Alert After the Invocation of an Action Set	9-18
What Happens at Runtime: How the Action Set Provides an Alert	9-19
How to Configure Error Handling for an Action Set	9-20
How to Prompt the User for Confirmation in an Action Set	9-21

What Happens at Runtime: How the Action Set Prompts the User for Confirmation	9-23
Configuring the Runtime Ribbon Tab	9-24
How to Define a Workbook Ribbon Command for the Runtime Ribbon Tab	9-26
How to Configure a Worksheet Ribbon Command for the Runtime Ribbon Tab	9-27
What Happens at Runtime: Ribbon Commands in the Ribbon Tab	9-29
Displaying Web Pages from a Fusion Web Application	9-31
How to Display a Web Page in a Popup Dialog	9-32
How to Display a Web Page Search Form in a Popup Dialog	9-34
How to Display a Web Page in ADF Desktop Integration Runtime Task Pane	9-36
What You May Need to Know About Displaying Pages from a Fusion Web Application	9-37
Sending Data Between an Integrated Excel Worksheet and a Fusion Web Application Page	9-37
Sharing Data Control Frames Between Integrated Excel Worksheets and Fusion Web Application Pages	9-38
Configuring a Fusion Web Application for ADF Desktop Integration Frame Sharing	9-38
Localizing Fusion Web Application Pages	9-39
Using Row-Level Action Sets in a Table Column	9-39
How to Enable Row-Level Action Set Model Management	9-40
What Happens at Runtime: RowActionSetModelMgmtEnabled is Set to True	9-41
How to Synchronize Changes from ADF Table Component Using RowUpSyncNoFail	9-42
What Happens at Runtime: RowUpSyncNoFail Action is Invoked	9-43
How to Add a Custom Popup Picker Dialog to an ADF Table Column	9-43
Using EL Expression to Generate an Excel Formula	9-44
How to Configure a Cell to Display a Hyperlink Using EL Expression	9-45
What Happens at Runtime: How a Cell Displays a Hyperlink using an EL Expression	9-46
Using Calculated Cells in an Integrated Excel Workbook	9-47
How to Calculate the Sum of a Table-Type Component Column	9-48
What Happens at Runtime: How Excel Calculates the Sum of a Table-Type Component Column	9-49
Configuring Automatic Row Refresh in an ADF Table Component	9-49
How to Write Custom Methods for Use in Automatic Row Refresh	9-50
How to Configure Automatic Row Refresh in an ADF Table Component	9-50
What Happens at Runtime: Automatic Row Refresh in an ADF Table Component	9-52
What You May Need to Know About Automatic Row Refresh in an ADF Table Component	9-53
Using Macros in an Integrated Excel Workbook	9-54

10 Configuring the Appearance of Your Integrated Excel Workbook

About Configuring the Appearance of an Integrated Excel Workbook	10-1
Integrated Excel Workbook Configuration Use Cases and Examples	10-2
Additional Functionality for Configuring the Appearance of an Integrated Excel Workbook	10-2
Working with Styles	10-2
Predefined Styles in ADF Desktop Integration	10-3
Excel's Date Formats and Microsoft Windows' Regional and Language Options	10-4
How to Apply a Style to an Oracle ADF Component	10-5
What Happens at Runtime: How Style Is Applied to an ADF Component	10-6
Applying Styles Dynamically Using EL Expressions	10-7
What Happens at Runtime: How an EL Expression Is Evaluated	10-7
How to Write an EL Expression That Applies a Style at Runtime	10-8
What You May Need to Know About EL Expressions That Apply Styles	10-9
Using Labels in an Integrated Excel Workbook	10-10
Retrieving the Values of String Keys from a Resource Bundle	10-10
Retrieving the Values of Attribute Control Hints	10-11
How an Integrated Excel Workbook Evaluates a Label Property	10-12
Branding Your Integrated Excel Workbook	10-12
How to Brand an Integrated Excel Workbook	10-13
What Happens at Runtime: the BrandingItems Group of Properties	10-14
Displaying Tooltips in ADF Desktop Integration Components	10-15
How to Add a Tool Tip to an ADF Table Component	10-17
How to Add a Tool Tip to a Form-Type Component	10-18
What You May Need to Know About Tooltips for Form-Type Components	10-19
What You May Need to Know About Tooltips for Table Columns	10-20
Using Worksheet Protection	10-20
How to Enable Worksheet Protection	10-21
What Happens at Runtime: How the Locked Property Works	10-22
What You May Need to Know About Worksheet Protection	10-22
Using ADF Desktop Integration EL-based Properties with Custom Attribute Properties	10-23
How to Enable Custom Attribute Properties in Integrated Excel Workbook	10-23
What Happens at Runtime: CustomAttributePropertiesEnabled is Set to True	10-24
What You May Need to Know About the CustomAttributePropertiesEnabled Property	10-24

11 Internationalizing Your Integrated Excel Workbook

About Internationalizing Your Integrated Excel Workbook	11-1
Internationalizing Integrated Excel Workbook Use Cases and Examples	11-2

Additional Functionality for Internationalizing Integrated Excel Workbook	11-2
Using Resource Bundles in an Integrated Excel Workbook	11-3
How to Register a Resource Bundle in an Integrated Excel Workbook	11-3
What You May Need to Know About Resource Bundles	11-4
Localization in ADF Desktop Integration	11-5
Configuring Fusion Web Application to Override Server-Side Locale Settings	11-6
How to Create a User Preference Handler	11-7
How to Register the User Preference Handler	11-7

12 Securing Your Integrated Excel Workbook

About Security In Your Integrated Excel Workbook	12-1
Integrated Excel Workbook Security Use Cases and Examples	12-1
Additional Functionality for Integrated Excel Workbook in a Secure Fusion Web Application	12-3
Authenticating the Excel Workbook User	12-3
What Happens at Runtime: How the Login Method Is Invoked	12-3
What Happens at Runtime: How the Web Application Session is Terminated	12-4
Checking the Integrity of an Integrated Excel Workbook's Metadata	12-5
How to Reset the Workbook ID	12-5
What Happens When the Metadata Tamper-Check Is Performed	12-6
What You May Need to Know About Securing an Integrated Excel Workbook	12-7
Authorizing the Excel Workbook User	12-7
What You May Need to Know About ADF Desktop Integration-Disabled Worksheet	12-8

13 Adding Validation to Your Integrated Excel Workbook

About Adding Validation to an Integrated Excel Workbook	13-1
Integrated Excel Workbook Validation Use Cases and Examples	13-1
Additional Functionality for Adding Validation to an Integrated Excel Workbook	13-2
Using the Status Viewer to Report Error Messages to End Users	13-3
How to Manage the Automatic Display of the Status Viewer	13-4
Providing Data Entry Validation for an Integrated Excel Workbook	13-5
Providing Data Entry Validation Using ADF Desktop Integration	13-5
How to Enable or Disable ADF Desktop Integration Data Entry Validation	13-7
Providing Data Validation Using Excel	13-8
How to Upload Excel Cell Errors to the Fusion Web Application	13-8
Providing Server-Side Validation for an Integrated Excel Workbook	13-9
Providing a Row-by-Row Status on an ADF Table Component	13-10
Adding Detail to Error Messages in an Integrated Excel Workbook	13-11
Handling Data Conflicts When Uploading Data from a Workbook	13-12

How to Configure a Workbook to Handle Data Conflicts When Uploading Data	13-13
What Happens at Runtime: How Data Conflicts Are Handled	13-13

14 Testing Your Integrated Excel Workbook

About Testing Your Integrated Excel Workbook	14-1
Integrated Excel Workbook Testing Use Cases and Examples	14-1
Additional Functionality for Testing an Integrated Excel Workbook	14-1
Testing Your Fusion Web Application	14-2
Validating the Integrated Excel Workbook Configuration	14-3
How to Validate the Integrated Excel Workbook Configuration	14-3
What Happens When You Validate the Integrated Excel Workbook Configuration	14-4
How to Fix Validation Failures	14-4
How to Log the Integrated Excel Workbook Configuration Validation Failures at Runtime	14-6
Testing Your Integrated Excel Workbook	14-6

15 Deploying Your Integrated Excel Workbook

About Deploying Your Integrated Excel Workbook	15-1
Integrated Excel Workbook Deployment Use Cases and Examples	15-1
Additional Functionality for Deploying Your Integrated Excel Workbook	15-2
Making ADF Desktop Integration Available to End Users	15-2
Publishing Your Integrated Excel Workbook	15-3
How to Publish an Integrated Excel Workbook from Excel	15-3
How to Publish an Integrated Excel Workbook Using the Command Line Publish Tool	15-4
What Happens When You Publish an Integrated Excel Workbook	15-5
Deploying a Published Workbook with Your Fusion Web Application	15-6
What Happens When You Deploy an ADF Desktop Integration-Enabled Fusion Web Application from JDeveloper	15-8
Fusion Web Application is Deployed on Oracle WebLogic Server	15-8
What Happens at Runtime: End User Requests a Published Workbook	15-8
Passing Parameter Values from a Fusion Web Application Page to a Workbook	15-9
How to Configure the Fusion Web Application's Page to Pass Parameters	15-12
How to Configure Parameters Properties in the Integrated Excel Workbook	15-14
How to Configure the Page Definition File for the Worksheet to Receive Parameters	15-18
What Happens at Runtime: How Parameters Are Passed from a Fusion Web Application to the Integrated Excel Workbook	15-19
Customizing Workbook Integration Metadata at Runtime	15-19
How to Enable Workbook Customization at Runtime	15-19

What Happens at Runtime: Workbook Integration Metadata is Customized	15-20
What You May Need to Know About Customizing Workbook Integration Metadata	15-20
Integrating ADF Workbook Composer into Your Fusion Web Application	15-21
How to Integrate ADF Workbook Composer into Your Fusion Web Application	15-21
What Happens at Runtime: ADF Workbook Composer is Invoked	15-23
What You May Need to Know About ADF Workbook Composer	15-23

16 Using an Integrated Excel Workbook Across Multiple Web Sessions

About Using an Integrated Excel Workbook Across Multiple Web Sessions	16-1
Using an Integrated Excel Workbook Across Multiple Web Sessions Use Cases and Examples	16-2
Additional Functionality for Using an Integrated Excel Workbook Across Multiple Web Sessions	16-2
Restore Server Data Context Between Sessions	16-2
How to Configure an Integrated Excel Workbook to Restore Server Data Context	16-2
What Happens at Runtime: How the Integrated Excel Workbook Restores Server Data Context	16-4
Caching of Static Information in an Integrated Excel Workbook	16-4
Caching Lists of Values for Use Across Multiple Web Sessions	16-5
Using Explicit Worksheet Setup Action	16-6
How to Configure Explicit Worksheet Setup Action	16-6
What You May Need to Know About Explicit Worksheet Setup Action	16-7

17 Administering ADF Desktop Integration

Installing and Upgrading ADF Desktop Integration	17-1
Prerequisites for Installing ADF Desktop Integration Add-in	17-2
How to Install the ADF Desktop Integration Add-in From a Web Server	17-3
How to Upgrade the ADF Desktop Integration Add-in	17-4
How to Run ADF Desktop Integration Installer from Command Line	17-4
How to Manage the Display of the System Check to End Users	17-5
Running the Client Health Check Tool	17-7
ADF Desktop Integration Logs	17-8
Security in ADF Desktop Integration	17-9
End User Authentication	17-9
What You May Need to Know About Configuring Security in a Fusion Web Application	17-9
What You May Need to Know About Resource Grants for Web Pages	17-11
Verifying the Client Version of ADF Desktop Integration	17-12
How to Disable the Install Option on the Client-Server Version Check Dialog	17-14

Verifying Integrated Excel Workbook Metadata	17-15
How to Disable the Metadata Tamper-Check in the Fusion Web Application	17-15
Common ADF Desktop Integration Error Messages and Problems	17-16

A ADF Desktop Integration Component Properties and Actions

Frequently Used Properties in the ADF Desktop Integration	A-1
ADF Input Text Component Properties	A-2
ADF Output Text Component Properties	A-3
ADF Label Component Properties	A-3
ADF List of Values Component Properties	A-4
ADF Image Component Properties	A-4
ADF Input Date Component Properties	A-5
ModelDrivenColumnComponent Subcomponent Properties	A-5
TreeNodeList Subcomponent Properties	A-6
ADF Table Component Properties and Actions	A-7
ADF Table Component Properties	A-7
ADF Table Component Column Properties	A-13
ADF Table Component Actions	A-16
ADF Read-only Table Component Properties and Actions	A-19
Action Set Properties	A-20
Confirmation Action Properties	A-22
Dialog Action Properties	A-23
Workbook Actions and Properties	A-24
Worksheet Actions and Properties	A-28
ADF Desktop Integration Compatibility Properties	A-34

B ADF Desktop Integration EL Expressions

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

C Troubleshooting an Integrated Excel Workbook

Verifying That Your Fusion Web Application Supports ADF Desktop Integration	C-1
Generating ADF Desktop Integration Diagnostic Reports	C-2
How to Generate the ADF Desktop Integration Diagnostic Report	C-2
What You May Need to Know About the ADF Desktop Integration Diagnostic Report	C-3
Troubleshooting Connection Problems to Fusion Web Applications	C-4
Verifying End-User Authentication for Integrated Excel Workbooks	C-5

	Generating Log Files for an Integrated Excel Workbook	C-6
	About Server-Side Logging	C-6
	Using the Oracle Diagnostics Log Analyzer to Analyze ADF Desktop Integration Servlet Requests	C-6
	About Client-Side Logging	C-7
	What You May Need To Know About Always-On Logging	C-7
	Enabling Transient Verbose Logging	C-8
	How to Configure Logging in the Oracle ADF Tab	C-10
	About the ADF Desktop Integration Configuration File	C-12
	How to Configure Logging Using User Environment Variables	C-13
	What You May Need to Know About the adfdi-common Object	C-14
D	ADF Desktop Integration Settings in the Web Application Deployment Descriptor	
	Configuring the ADF Desktop Integration Servlet	D-1
	Configuring the ADF Desktop Integration Excel Download Filter	D-4
	Configuring the ADF Library Filter for ADF Desktop Integration	D-7
	Examples in a Deployment Descriptor File	D-8
E	Java Data Types Supported By ADF Desktop Integration	
	Primitive Java Types	E-1
	Object Java Types	E-1
F	Using the ADF Desktop Integration Model API	
	About the Temporary Row Object	F-1
	About ADF Desktop Integration Model API	F-2
	How to Add ADF Desktop Integration Model API Library to Your JDeveloper Project	F-2
G	End User Actions	
	Installing, Upgrading, and Removing ADF Desktop Integration	G-1
	How to Install ADF Desktop Integration on Your System	G-1
	How to Remove ADF Desktop Integration	G-2
	How to Upgrade ADF Desktop Integration On a Local System	G-3
	Removing Personal Information	G-4
	Limitations of an Integrated Excel Workbook at Runtime	G-4
	Using an Integrated Excel Workbook	G-5
	How to Insert or Paste Rows in an ADF Table Component	G-6

How to Sort ADF Table Data in an Integrated Excel Workbook	G-7
How to Delete a Row in ADF Table of an Integrated Excel Workbook	G-7
Handling Time Zone Conversion	G-8
Providing Diagnostic and Logging Information to Technical Support	G-8

Preface

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

Audience

This document is intended for enterprise developers who develop desktop applications to integrate with the Oracle Application Development Framework (Oracle ADF). It also provides information for system administrators and end users who work with these desktop applications.

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
monospace	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 for 12c (12.2.1.4.0)

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 (12.2.1.4.0)

Oracle ADF Desktop Integration 12c (12.2.1.4.0) includes the following new and changed development features for this document.

- You can now configure an ADF Table component to display an error message column where row-level error messages appear at runtime. See [Displaying Error Messages in an Error Message Column](#).
- The ADF Desktop Integration add-in installer now sets the `FEATURE_BROWSER_EMULATION` feature control key for Excel in the registry to IE 11 mode. This setting allows the embedded browser to function in IE 11 mode. See [How to Install ADF Desktop Integration](#) and the "ADFdi Notes on Embedded Browser" document that you can retrieve from My Oracle Support (<https://support.oracle.com>) if you search for Doc ID 2013081.1.
- ADF Desktop Integration has changed the algorithm that determines the number of rows it processes during the execution of `Upload` and `DeleteFlaggedRows` actions. It now adapts to the current performance and adjusts the number of rows it uploads per request to avoid timeouts and to show regular progress to users.
- ADF Desktop Integration now submits multiple requests to delete rows from an ADF Table component using the `DeleteFlaggedRows` action in a single HTTP request.

1

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:

- [About ADF Desktop Integration](#)
- [About ADF Desktop Integration with Microsoft Excel](#)

For definitions of unfamiliar terms found in this and other books, see the Glossary.

About ADF Desktop Integration

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

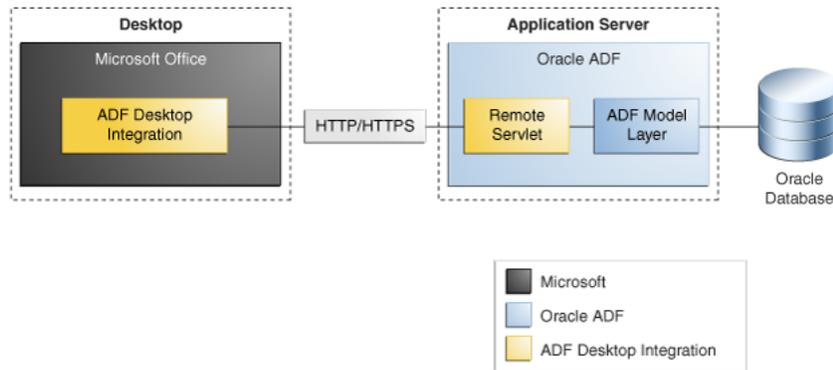
Many end users of Fusion web applications use desktop applications, such as Microsoft Excel, to manage information also used by their web application. ADF Desktop Integration provides a framework for Oracle Application Development Framework (Oracle ADF) developers to extend the functionality provided 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 *Oracle ADF Architecture 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



See the ADF Desktop Integration page on Oracle Technology Network (OTN) at:

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

About ADF Desktop Integration with Microsoft Excel

ADF Desktop Integration supports integration with Microsoft Excel. You create page definition files in your Fusion web application that expose the Oracle ADF bindings to use in the Excel workbooks that you integrate with the Fusion web application.

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.

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. See Working with Page Definition Files for an Integrated Excel Workbook.
Excel	<ul style="list-style-type: none"> • Create the Excel workbooks that you intend to configure with Oracle ADF functionality. See Adding an Integrated Excel Workbook to a Fusion Web Application. • 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. See the following sections and chapters: <ul style="list-style-type: none"> – Getting Started with the Development Tools This chapter provides an overview of the tools that ADF Desktop Integration provides to configure an Excel workbook with Oracle ADF functionality. – Working with ADF Desktop Integration Form-Type Components This chapter describes how to insert ADF Desktop Integration form-type components into Excel worksheets and configure their properties to determine behavior at runtime. – Working with ADF Desktop Integration Table-Type Components 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. – Adding Validation to Your Integrated Excel Workbook This chapter describes how to provide validation for your integrated Excel workbook. • Test your integrated Excel workbook. See Testing Your Integrated Excel Workbook. • After completing the integration of the Excel workbook with the Fusion web application, you deploy it to make it available to the end users. See Deploying Your Integrated Excel Workbook.

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 formulas and filters.

2

Introduction to the ADF Desktop Integration Sample Application

ADF Desktop Integration provides the Summit sample application that contains several Microsoft Excel workbooks which are integrated with a Fusion web application. This chapter includes the following sections:

- [About the Summit Sample Application for ADF Desktop Integration](#)
- [Setting Up and Running the Summit Sample Application for ADF Desktop Integration](#)
- [Overview of the Fusion Web Application in the Summit Sample Application for ADF Desktop Integration](#)
- [Overview of the Integrated Excel Workbooks in the Summit Sample Application for ADF Desktop Integration](#)

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 contains specific features that can be identified in the developer's guide. All of the samples use the same underlying database schema which makes it easy for accessing the source code, and also to experience the runtime behavior in a standalone way.

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

Set up the development environment before you download and run the Summit sample application for ADF Desktop Integration.

For information about how to set up the development environment, see [Setting Up Your Development Environment](#) .

Running the Summit sample application for ADF Desktop Integration requires you to:

1. Download the application resources, as described in [How to Download the Application Resources](#).
2. Run the Summit sample application, as described in [How to Run the Summit Sample Application for ADF Desktop Integration](#).

How to Download the Application Resources

You download the application resources from Oracle Technology Network.

To download the Summit sample application for ADF Desktop Integration:

1. Download and install Oracle JDeveloper.
For information, see *Installing the Oracle JDeveloper Software* in *Installing Oracle JDeveloper*.
2. Install ADF Desktop Integration. See [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 [Upgrading ADF Desktop Integration](#).

3. Download and install the Summit ADF schema. See *How to Install the Summit ADF Schema* in *Developing Fusion Web Applications with Oracle Application Development Framework*.
4. Download and install the Summit sample application for ADF Desktop Integration ZIP file from Oracle Technology Network.

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

See [How to Run the Summit Sample Application for ADF Desktop Integration](#).

How to Run the Summit Sample Application for ADF Desktop Integration

To run the Summit sample application, extract the contents of the zip file and open the .JWS file in JDeveloper.

To run the Summit sample application for ADF Desktop Integration:

1. Extract the contents the 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	c##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 Model1.jpx file.
8. Select the **ViewController** project and click the **Run** button in JDeveloper's main menu.

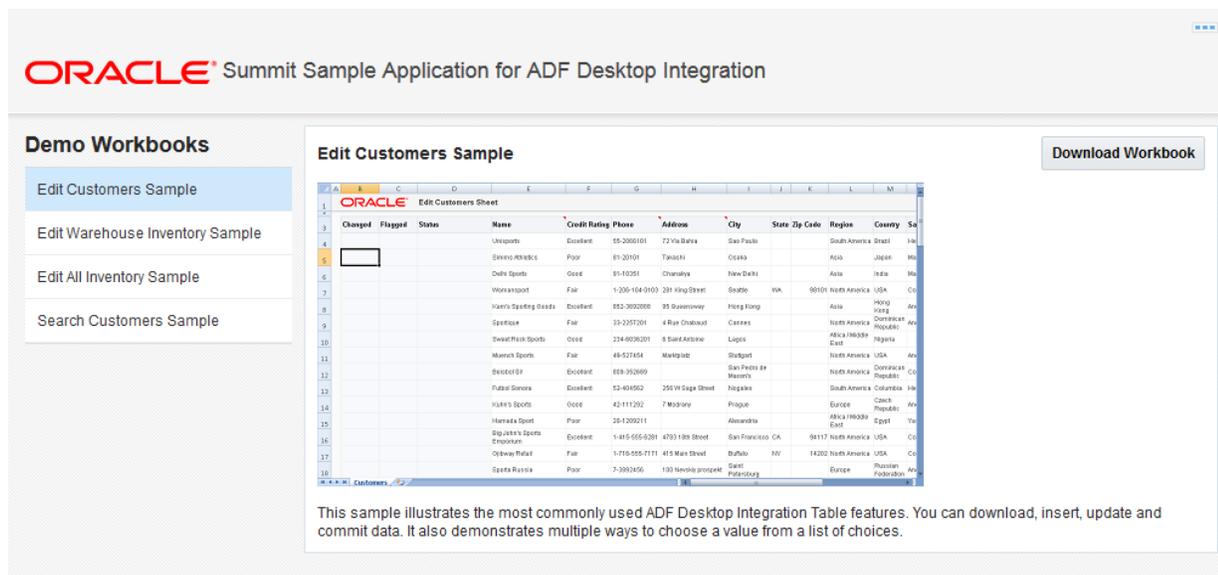
Overview of the Fusion Web Application in the Summit Sample Application for ADF Desktop Integration

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

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



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 from the `MainPage.jsf` of the application. When an end user clicks a link to download a workbook, a Java applet verifies that the ADF Desktop Integration add-in is present on the end user's machine. If the add-in is found, the workbook download begins automatically. Otherwise, ADF Desktop Integration prompts the end user to install the add-in. If Java is not installed on the end user's machine or is disabled by the end user's security settings, the Java applet is unable to verify the presence of the ADF Desktop Integration add-in. ADF Desktop Integration informs the end user that the installation of the add-in cannot be verified. It presents the end user with the option to download the workbook and/or install the add-in.

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
Edit Customers Sample	Downloads <code>EditCustomers.xlsx</code> workbook.
Edit Warehouse Inventory Sample	Downloads <code>EditWarehouseInventory.xlsx</code> workbook.
Edit All Inventory Sample	Downloads <code>EditAllInventory.xlsx</code> workbook.
Search Customers Sample	Downloads <code>CustomerSearch.xlsx</code> workbook.

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 `CustomerSearch.xlsx`, `EditAllInventory.xlsx`, `EditCustomers.xlsx`, and `EditWarehouseInventory.xlsx` integrated Excel workbooks.

The `CustomerSearch.xlsx` workbook demonstrates how a custom web page can be used to perform a search prior to downloading data into an ADF Table component configured to be read-only.

The `EditAllInventory.xlsx` workbook demonstrates how to create an editable table with a denormalized master-detail relationship. It also demonstrates how to use a date picker, group columns, and delete existing data records.

The `EditCustomers.xlsx` workbook illustrates the most commonly used ADF Desktop Integration ADF Table component features. You can download, insert, update and commit data. It also demonstrates multiple ways to choose a value from a list of choices.

The `EditWarehouseInventory.xlsx` workbook illustrates how to use ADF Desktop Integration form components with a detail table. You can download and update data in a master form and its detail table. This sample also demonstrates how to use workbook parameters to control the workbook initialization.

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

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 Customers tab in the Ribbon of the `EditCustomers.xlsx` workbook.

Figure 2-2 Runtime Customers Tab



Downloading Data Rows

Some workbooks, such as `EditCustomers.xlsx` workbook, use an ADF Table component to download information 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 [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 [Downloading 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 [How to Invoke an Action Set from a Worksheet Event](#).

Modify Customers and Warehouses Information in the Workbooks

The `EditCustomers.xlsx` and `EditWarehouseInventory.xlsx` workbooks enable end users to edit customers and warehouses information that the ADF Table component and form components download 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 [Inserting an ADF Table Component into an Excel Worksheet](#).
- For information about special columns, such as **Status** and **Changed**, see [Special Columns in the ADF Table Component](#).
- For information about action sets, see [Adding Interactivity to Your Integrated Excel Workbook](#).
- For information about lists of values, see [Working with Lists of Values](#).

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** ribbon command that invokes the ADF Table component's `Upload` action. For information about implementing this functionality, see [Uploading Changes from an ADF Table Component](#).

3

Setting Up Your Development Environment

Describes how to set up the development environment to integrate an Excel workbook with a Fusion web application, plus how to upgrade and remove ADF Desktop Integration.

This chapter includes the following sections:

- [About Setting Up Your Development Environment](#)
- [Required Oracle ADF Modules and Third-Party Software](#)
- [Installing ADF Desktop Integration](#)
- [Removing ADF Desktop Integration](#)
- [Upgrading ADF Desktop Integration](#)

About Setting Up Your Development Environment

Setting up the development environment involves making sure that you have the correct versions of JDeveloper and Microsoft Office installed.

After verifying that you have the required software, as described in [Required Oracle ADF Modules and Third-Party Software](#), complete the setup of your development environment by 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.

Required Oracle ADF Modules and Third-Party Software

Ensure that you have the required Oracle ADF modules and third-party software installed and configured before you begin to integrate your Excel workbook with a Fusion web application.

- 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 information about supported versions of Windows, click the "Certification Information" link for this release on the following OTN page:

<http://www.oracle.com/technetwork/developer-tools/jdev/documentation/index.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 information about supported versions of Excel, click the "Certification Information" link for this release on the following OTN page:

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

- 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.

See the "Supported Platforms for ADF Desktop Integration" document for definitive information regarding supported platforms for ADF Desktop Integration. You can retrieve this document from My Oracle Support (<https://support.oracle.com>) if you search for Doc ID 2242428.1.

Consider running the Client Health Check to determine if your environment is configured correctly after you run the ADF Desktop Integration installer. See [Running the Client Health Check Tool](#).

Installing ADF Desktop Integration

When you run the ADF Desktop Integration installer, it verifies whether required software is installed on the system where you want to install the add-in. If required software is not installed, ADF Desktop Integration installs it.

The following list identifies the list of required software that ADF Desktop Integration verifies is installed. ADF Desktop Integration installs missing software in the order specified.

1. Windows Installer 4.5
2. Microsoft .NET Framework

The Microsoft .NET Framework 4.5.2 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/>.

 **Note:**

- Microsoft .NET Framework 4.5.2 is the minimum required version. ADF Desktop Integration is also compatible with Microsoft .NET Framework 4.6.
- Installation of Microsoft .NET Framework may require you to restart the system where you install it. After the restart, the installer 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 `adfdi-excel-addin-installer.exe` installer available in the following directory:

```
MW_HOME\oracle_common\modules\oracle.adf.desktopintegration
```

For information about how to set up ADF Desktop Integration, see [How to Install ADF Desktop Integration](#). You can also install it from an ADF Desktop Integration-enabled Fusion web application. See [How to Install the ADF Desktop Integration Add-in From a Web Server](#).

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. See [How to Install ADF Desktop Integration](#).

How to Install ADF Desktop Integration

You can install the ADF Desktop Integration add-in from JDeveloper, and then create and test integrated Excel workbooks.

Although you do not require administrator privileges to install the ADF Desktop Integration add-in, administrator privileges may be required to run the installer 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.

Before you begin:

It may be helpful to have an understanding of ADF Desktop Integration requirements. See [Installing ADF Desktop Integration](#).

To install ADF Desktop Integration:

1. Open JDeveloper.
2. From the **Tools** menu, choose **Install ADF Desktop Integration**.

 **Note:**

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

3. In the ADF Desktop Integration Installer page of Oracle ADF Desktop Integration 12c Add-In for Excel wizard, click **Install**.

Follow the instructions that appear in the dialog boxes to successfully install the required components. If you encounter an error during the installation process, an error message with a description appears, and installation is rolled back. For more details, check the `adfdi-installer-log.txt` error log file in the `temp` directory of the user profile.

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

It may also be necessary to open Microsoft Excel to accept additional prompts to allow the installation to complete.

Note the following points about installation:

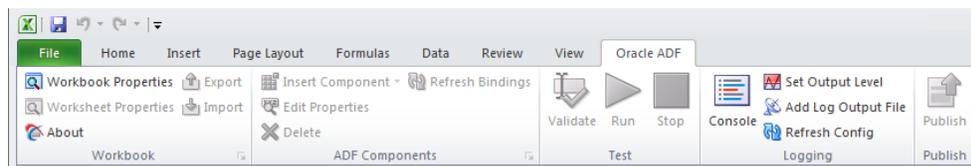
- You can also install ADF Desktop Integration by running `adfdi-excel-addin-installer.exe` available in the following directory:

```
MW_HOME/oracle_common/modules/oracle.adf.desktopintegration
```

Before you run the installer, remember that the ADF Desktop Integration add-in installer does not enable designer features by default. You must enable the designer features in the add-in to create and edit integrated Excel workbooks.

In the ADF Desktop Integration Installer page of the wizard, click **Developer Options**, and then in the Developer Options page select the **Enabled** option, and click **Install**. If you do not select the **Enabled** option, the Oracle ADF tab shown in [Figure 3-1](#) does not appear in Microsoft Excel.

Figure 3-1 Oracle ADF Tab in Microsoft Excel



- Designer features are automatically enabled if you install ADF Desktop Integration from JDeveloper.
- You can also install ADF Desktop Integration from the command line.
- The ADF Desktop Integration add-in installer sets the `FEATURE_BROWSER_EMULATION` feature control key for Excel in the registry to IE 11 mode. This setting allows the embedded browser to function in IE 11 mode. See also the "ADFDi Notes on Embedded Browser" document that you can retrieve from My Oracle Support (<https://support.oracle.com>) if you search for Doc ID 2013081.1.

If you want to install ADF Desktop Integration for end users, see [Installing, Upgrading, and Removing ADF Desktop Integration](#).

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 12c Add-in for Excel** program and click **Uninstall**.

 **Note:**

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

Upgrading ADF Desktop Integration

To upgrade to a new version, run the ADF Desktop Integration installer from the new version. It is not necessary to uninstall the old version first.

You can run the installer from:

- JDeveloper Tools menu
- Welcome page of the running Fusion web application (see [Verifying That Your Fusion Web Application Supports ADF Desktop Integration](#))
- File system (see the Notes section of [How to Install ADF Desktop Integration](#))

 **Note:**

- If you are upgrading from a previous release, you may receive a message that says that ADF Desktop Integration was installed with an incompatible installer. In this case, you must uninstall the ADF Desktop Integration add-in prior to running the installer.
- When you test your integrated Excel workbooks, ADF Desktop Integration may prompt you with a dialog to install a newer add-in version available from your test server. Although this dialog contains an option to skip, you, as a developer, should never skip the installation of the add-in version that the dialog proposes. It is important to keep the version of the add-in that the client uses synchronized with the ADF Desktop Integration servlet version.

4

Preparing Your Integrated Excel Workbook

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:

- [About Preparing Your Integrated Excel Workbooks](#)
- [Working with Page Definition Files for an Integrated Excel Workbook](#)
- [Adding an Integrated Excel Workbook to a Fusion Web Application](#)
- [Enabling ADF Desktop Integration in an Excel Workbook](#)
- [Enabling ADF Desktop Integration Manually](#)
- [Using an Integrated Excel Workbook with Older Versions of ADF Desktop Integration](#)

About Preparing Your Integrated Excel Workbooks

Having developed the Fusion web application, you integrate the Excel workbook with the Fusion web application.

This chapter describes the tasks that you perform to integrate an Excel workbook with the Fusion web application. The subsequent chapters of the guide enable you to configure the integrated workbook with Oracle ADF components that provide the functionality you require at runtime.

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

Note:

Before you start, ensure that designer tools of ADF Desktop Integration are enabled, as described in [Installing ADF Desktop Integration](#).

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 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	
ADF Read-only Table	Tree binding	
ADF Table	Tree binding	

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

For information about the elements and attributes in page definition files, see *pageNamePageDef.xml* in *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 *Using ADF Model in a Fusion Web Application* in *Developing Fusion Web Applications with Oracle Application Development Framework*.

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. See [Working with Page Definition Files for an Integrated Excel Workbook](#).

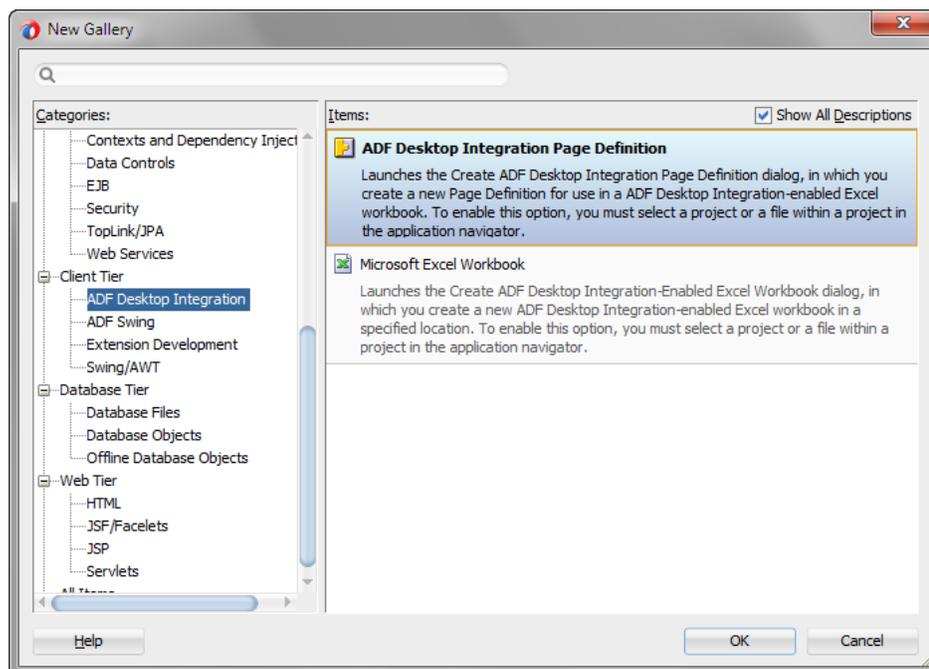
To create an ADF Desktop Integration page definition file:

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 4-1 shows the New Gallery with **ADF Desktop Integration** category and the **ADF Desktop Integration Page Definition** option.

Figure 4-1 New Gallery - ADF Desktop Integration Page Definition

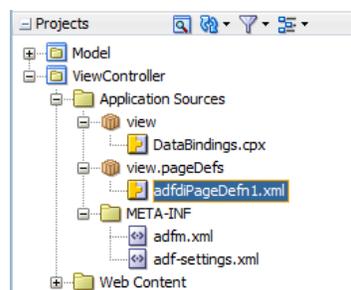


Click **OK**.

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 4-2 shows the **ViewController** project with the new page definition in the Applications window.

Figure 4-2 adfdiPageDefn.xml in Applications window



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

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 [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 Working with the `DataBindings.cpx` File of *Developing Fusion Web Applications with Oracle Application Development Framework*.

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 [How to Configure a New Integrated Excel Workbook](#).

The Oracle ADF tab provides a button (**Refresh Bindings**) that reloads all page definition files in an Excel workbook. For information about the Refresh Bindings button, see [About Development Tools](#).

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. See [Working with Page Definition Files for an Integrated Excel Workbook](#).

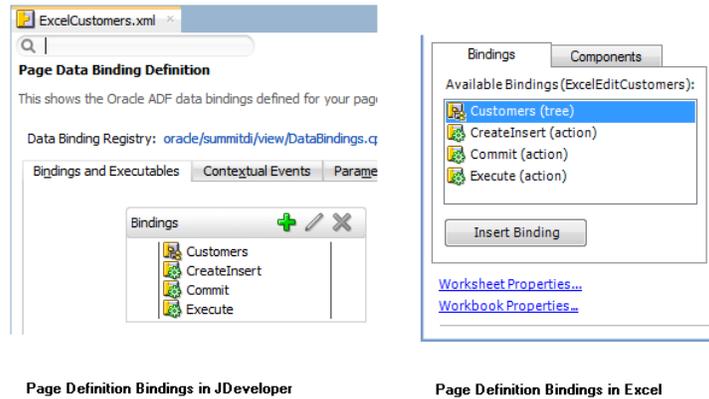
To reload page definition files in an Excel workbook:

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.

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 4-3](#) shows the bindings in the

ExcelCustomers.xml page definition file and the same bindings in the worksheet of the EditCustomers-DT.xlsx workbook.

Figure 4-3 Page Definition Bindings in JDeveloper and Integrated Excel Workbook



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 separate page definition file with each worksheet as described in [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.

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.

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.

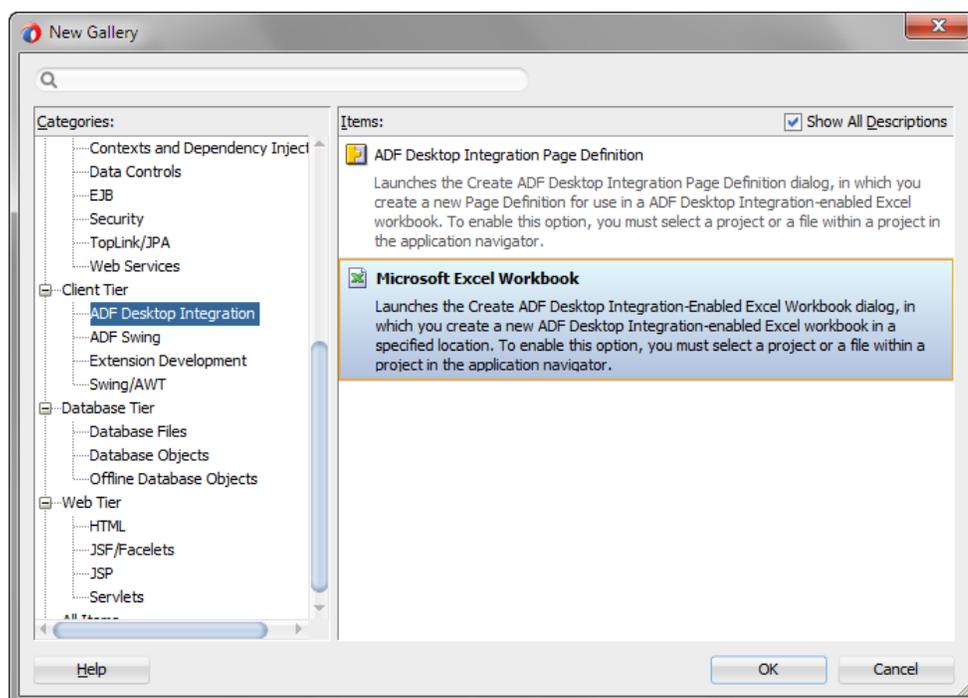
Before you begin:

It may be helpful to have an understanding of adding ADF Desktop Integration to a Fusion web application. See [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 4-4](#) shows the New Gallery with **ADF Desktop Integration** category and the **Microsoft Excel Workbook** option.

Figure 4-4 New Gallery - Microsoft Excel Workbook

Click **OK**.

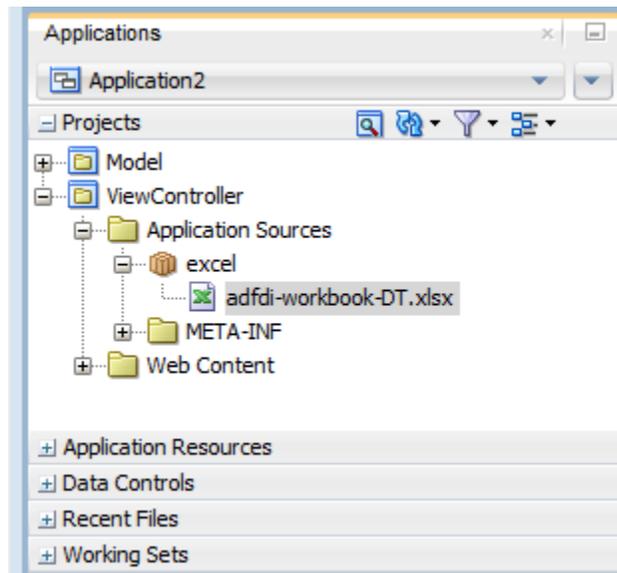
5. In the Create ADF Desktop Integration-Enabled Excel Workbook dialog, verify the desired location and type a unique workbook name. Consider adding a suffix of -DT to help with publishing later. For example, `MyWorkbook-DT.xlsx`.

By default, the integrated Excel workbook is saved as `adfdi-workbook-DT.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 applications.

Later, ADF Desktop Integration removes suffixes, such as `-DT`, when you publish the finalized integrated Excel workbook for distribution so that end users see meaningful filenames. For example, the Summit sample application publishes the workbook to edit customers using the `EditCustomers.xlsx` filename rather than `EditCustomers-DT.xlsx`.

6. Click **OK**.

JDeveloper adds the integrated Excel workbook into the Fusion web application, and automatically enables the project with ADF Desktop Integration. [Figure 4-5](#) shows the **ViewController** project.

Figure 4-5 adfdi-workbook-DT.xlsx in Applications window

How to Configure a New Integrated Excel Workbook

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

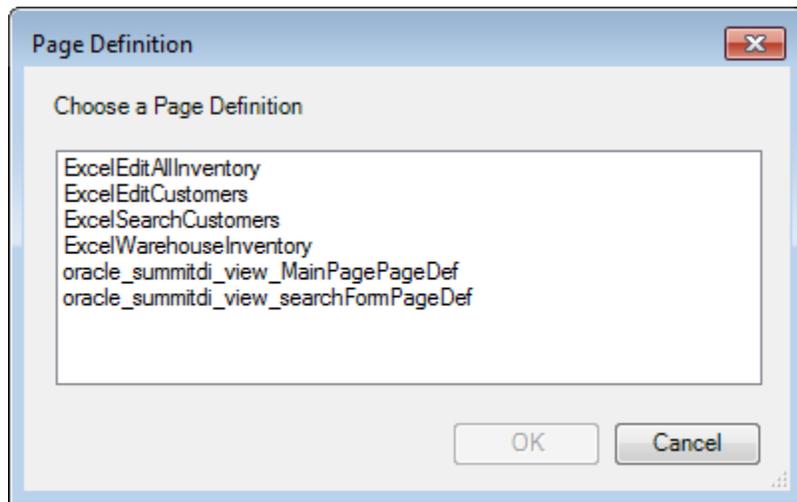
Before you begin:

It may be helpful to have an understanding of adding an integrated Excel workbook to a Fusion web application. See [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 4-6](#).

Figure 4-6 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 [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`.

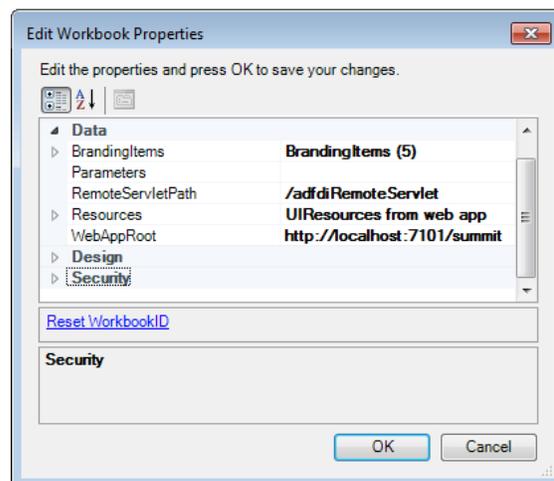
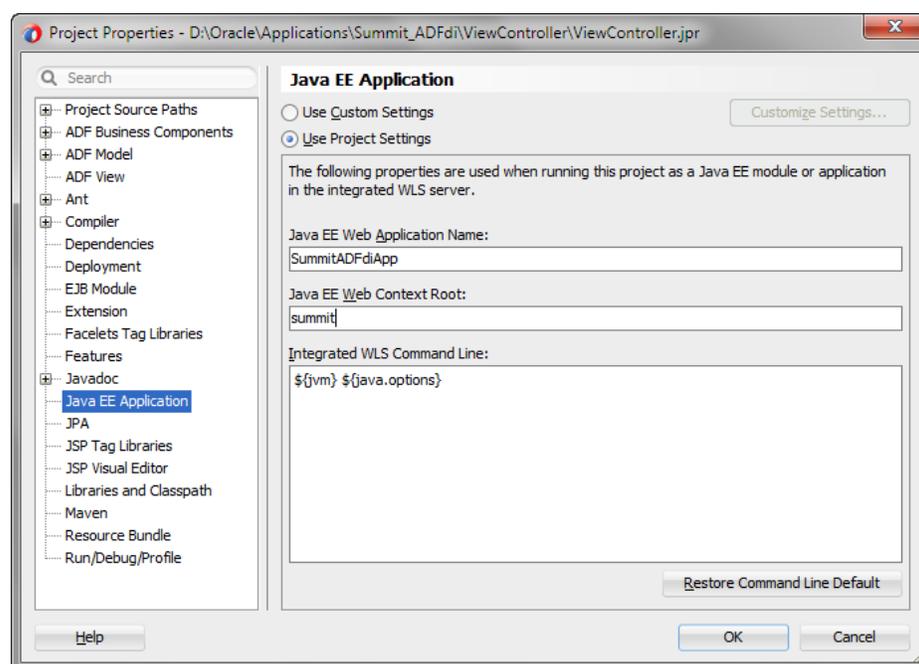
- WebAppRoot

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 4-7](#) shows the web context root used for the Summit sample application for ADF Desktop Integration in JDeveloper and integrated Excel workbook.

Figure 4-7 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 your system using the Summit sample application for ADF Desktop Integration:

```
http://localhost:7101/summit
```

For information about how to verify that the Fusion web application is online and that it supports ADF Desktop Integration, see [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`.

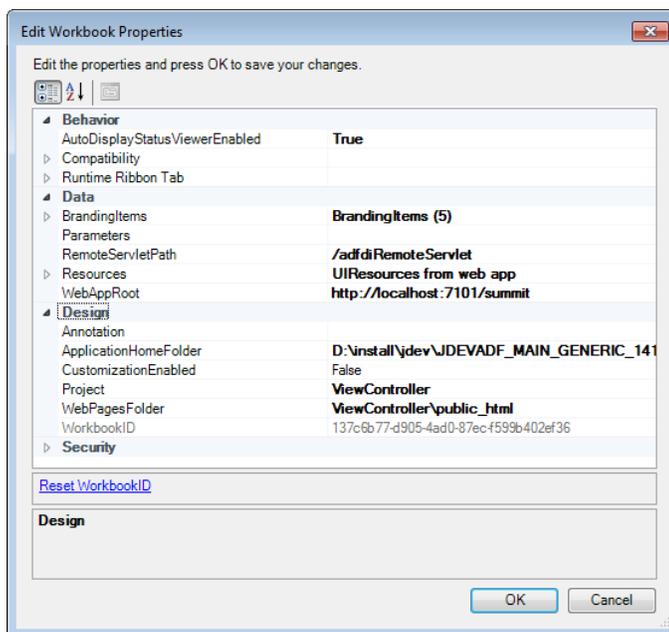
For information about securing the Fusion web application, see [Securing Web Applications in *Developing Applications with the WebLogic Security Service*](#).

- `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 4-8](#) 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 4-8 Edit Workbook Properties Dialog



4. Click **OK**.

 **Note:**

In Step 1, if the fully qualified path of the selected page definition file exceeds the Windows path length limit, a warning message appears when the Workbook Properties dialog is closed, and the page definition will not load.

5. Save the Excel workbook.

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. See [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.

Enabling ADF Desktop Integration in an Excel Workbook

ADF Desktop Integration automatically configures integrated Excel workbooks that you create by adding to a Fusion web application with desktop integration functionality. You must enable desktop integration functionality in a pre-existing workbook to make it an integrated Excel workbook and configure properties in the newly-integrated Excel workbook.

See [Adding an Integrated Excel Workbook to a Fusion Web Application](#).

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 supported file formats of Excel workbooks that you can use for integration with a Fusion web application, see [Required Oracle ADF Modules and Third-Party Software](#).

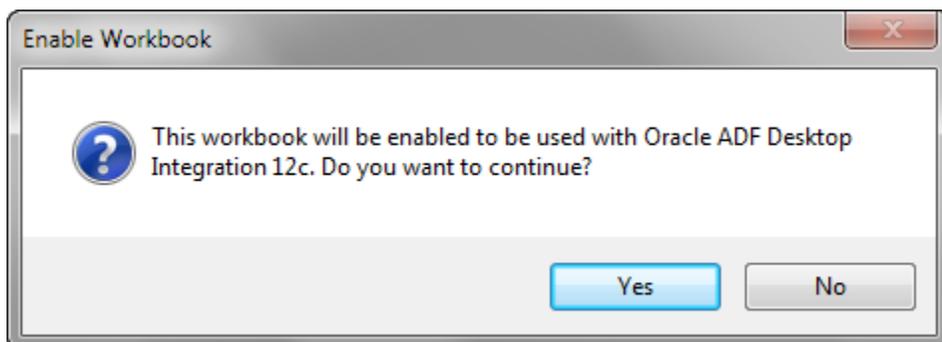
Before you begin:

It may be helpful to have an understanding of adding integrated Excel workbook to a Fusion web application. See [Adding an Integrated Excel Workbook to a Fusion Web Application](#).

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 4-9](#).

Figure 4-9 Enable Workbook Dialog



ADF Desktop Integration prepares your workbook, displays the ADF Desktop Integration Designer task pane, and opens the Browse For Folder dialog. See [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.

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:

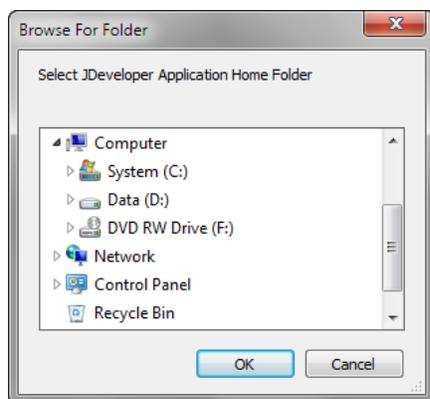
It may be helpful to have an understanding of adding an integrated Excel workbook to a Fusion web application. See [Enabling ADF Desktop Integration Manually](#).

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 4-10](#).

Figure 4-10 Browse For Folder Dialog



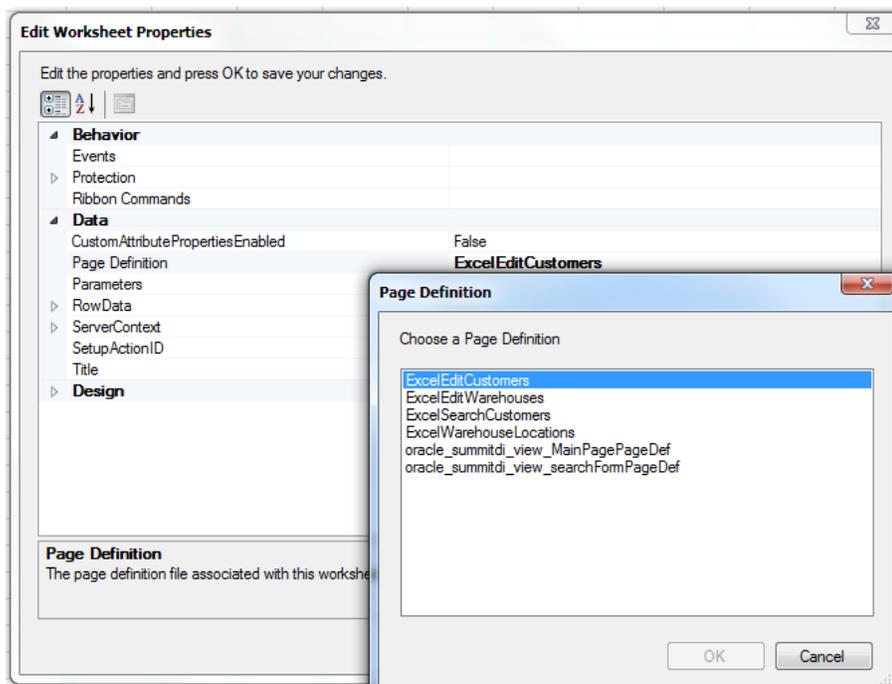
Use the Browse for Folder dialog to select the JDeveloper application home directory. In a typical JDeveloper project, the JDeveloper application home directory stores the *application_name.jws* file. The value you select is assigned to the `ApplicationHomeFolder` workbook property.

 **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 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 4-11](#).

Figure 4-11 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.

Enabling ADF Desktop Integration Manually

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

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. See [Enabling ADF Desktop Integration Manually](#).

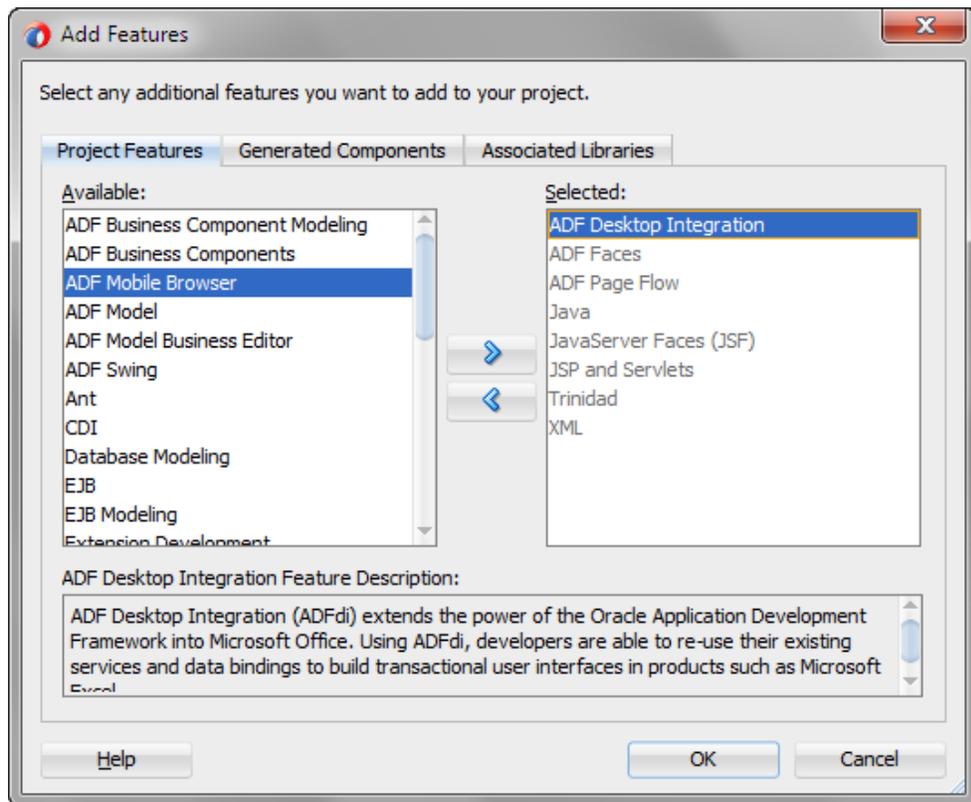
To manually add ADF Desktop Integration to your project:

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 4-12](#).

Figure 4-12 Add Features Dialog



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

For information about what happens when you add ADF Desktop Integration, see [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. See [Adding ADF Library Web Application Support](#).

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-19](#).

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

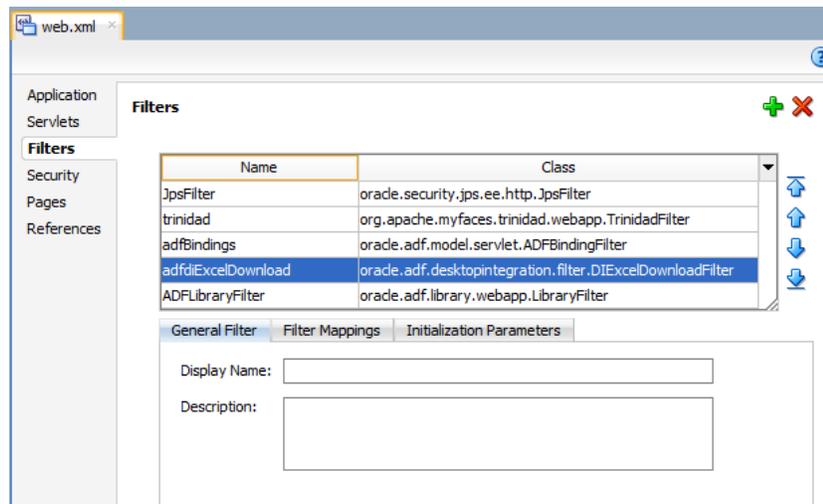
The previous list is not exhaustive. Adding ADF Desktop Integration to a project makes other changes to `web.xml`. Note that some entries in `web.xml` are added only if they do not already appear in the file.

Adding ADF Library Web Application Support

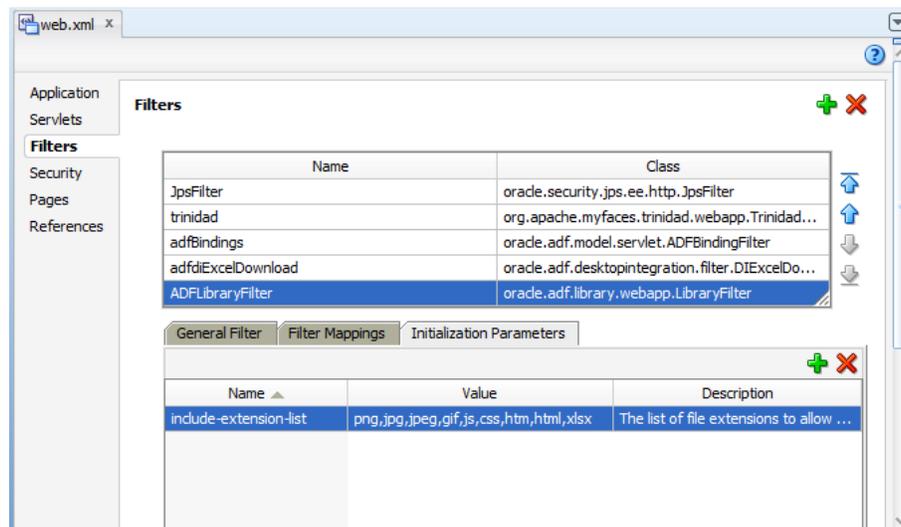
If you want to distribute integrated workbooks by adding them to ADF library files, add ADF Library web application support to the Fusion web application. See *Packaging a Reusable ADF Component into an ADF Library* in *Developing Fusion Web Applications with Oracle Application Development Framework*.

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 4-13](#) shows the **Filters** tab of the overview editor of the `web.xml` in JDeveloper.

Figure 4-13 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 4-14](#).

Figure 4-14 ADFLibraryFilter Using include-extension-list Parameter

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

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

Integrated Excel workbooks can be used with older versions of ADF Desktop Integration if the workbook does not contain features incompatible with the older version.

When you or your end users open an integrated Excel workbook created, or last updated, by a newer version of ADF Desktop Integration on a system running an older version of ADF Desktop Integration, a dialog appears if the integrated Excel workbook contains features that are incompatible with the older version of ADF Desktop Integration.

When you click **OK** on this dialog, ADF Desktop Integration disables the integrated Excel workbook and the end user cannot interact with the ADF Desktop Integration features in the workbook. The data in the workbook is not removed, but ADF Desktop Integration treats the workbook as a non-integrated workbook.

If the integrated Excel workbook does not contain incompatible features, no dialog appears and the workbook functions normally. For integrated Excel workbooks that contain incompatible features, upgrade the client version of ADF Desktop Integration, as described in [Upgrading ADF Desktop Integration](#). End users can upgrade their client version, as described in [How to Upgrade ADF Desktop Integration On a Local System](#).

Integrated Excel workbooks created using 12c (12.2.1) or earlier of ADF Desktop Integration do not have features that are incompatible with the ADF Desktop Integration 12c (12.2.1) client. A future release of ADF Desktop Integration may introduce features that will be incompatible with clients using 12c (12.2.1) or earlier of ADF Desktop Integration.

 **Note:**

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.

5

Getting Started with the Development Tools

ADF Desktop Integration provides a range of development tools (Bindings Palette, Components Palette, Property Inspector, and Expression Builder) and exposes a development environment in the Excel Ribbon of integrated Excel workbooks. This chapter includes the following sections:

- [About Development Tools](#)
- [Designer Ribbon Tab](#)
- [ADF Desktop Integration Designer Task Pane](#)
- [Using the Bindings Palette](#)
- [Using the Components Palette](#)
- [Using the Property Inspector](#)
- [Using the Binding ID Picker](#)
- [Using the Expression Builder](#)
- [Using the Web Page Picker](#)
- [Using the File System Folder Picker](#)
- [Using the Page Definition Picker](#)
- [Using the Collection Editors](#)
- [Using the Cell Context Menu](#)
- [Removing ADF Desktop Integration Components](#)
- [Exporting and Importing Excel Workbook Integration Metadata](#)

About Development Tools

ADF Desktop Integration provides several tools to configure Excel workbooks so that they can integrate with your Fusion web application.

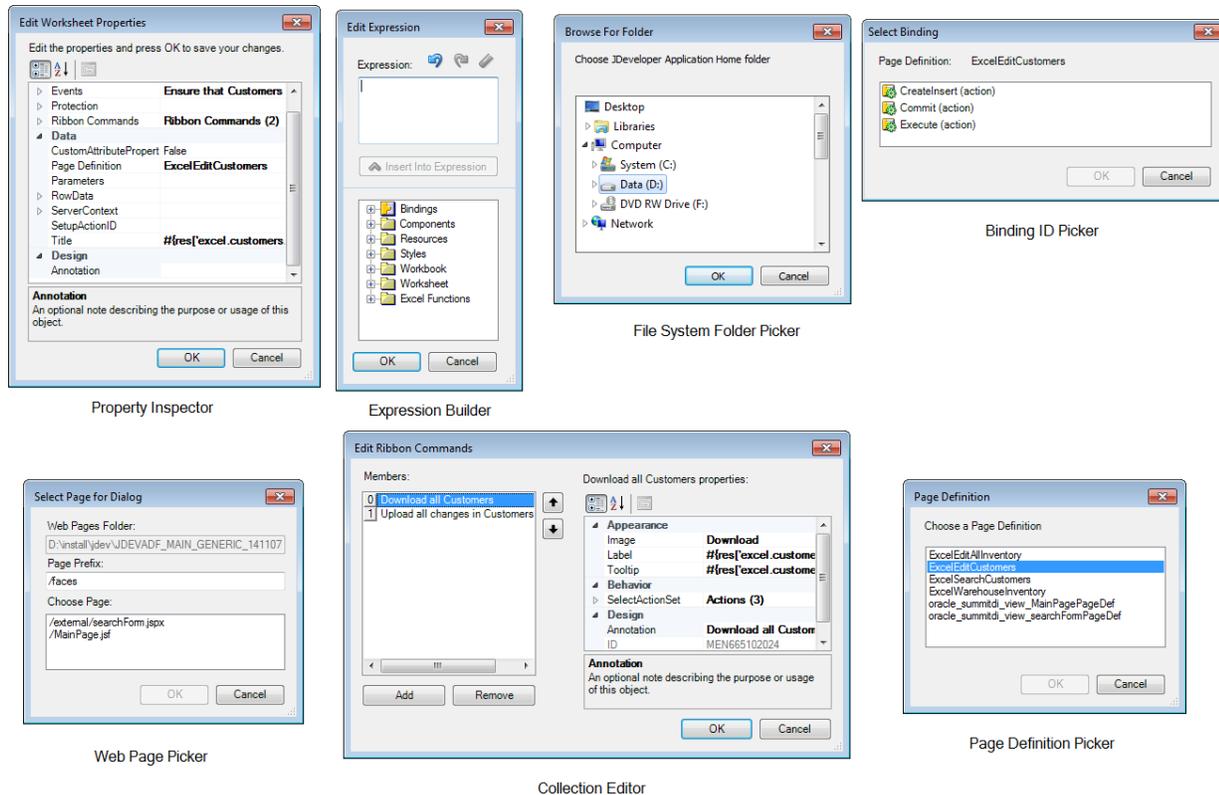
Using these tools you configure the workbook and corresponding worksheets to display, and edit, data from the Fusion web application in the integrated Excel workbook. The tools are available in the **Oracle ADF** tab and in the ADF Desktop Integration Designer task pane.

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.

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` an 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 Input Text, ADF Input Date and ADF Label, can be inserted from the Components palette, and configured using the Property Inspector and Expression Builder.

Additional Functionality for ADF Desktop Integration Development Tools

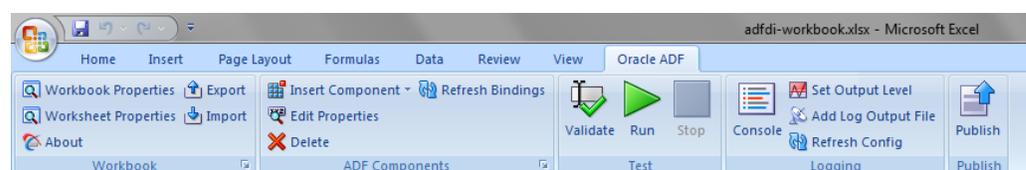
After adding the desired components and configuring your workbook, you may find that you need additional functionality such as changing the appearance of the workbook, and localizing it. The following sections describe other functionality that you can use:

- **Interactivity:** You add one or more action sets to your integrate Excel workbook for it to integrate with your Fusion web application. See [Adding Interactivity to 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. See [Internationalizing Your Integrated Excel Workbook](#).
- **Styles:** You can configure the display of your components using several predefined Excel styles. See [Working with Styles](#).
- **EL Expressions:** You can use EL expressions with the ADF Desktop Integration components. See [ADF Desktop Integration EL Expressions](#).

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, inserting Oracle ADF components and editing their properties, running the workbook in test mode, and publishing 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:**

Press Alt+C to access the **Oracle ADF** tab and view the shortcut keys for Oracle ADF tab ribbon commands from the keyboard.

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

Table 5-1 Oracle ADF Tab Options

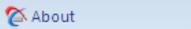
In this group...	Click this button...	To...	Mode when the button is available...
Workbook		Display the Edit Workbook Properties dialog to view and edit integrated Excel workbook properties. The button is also used to enable ADF Desktop Integration in a non-integrated Excel workbook.	Design
Workbook		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 dialog also provides access to the diagnostic report described in Generating an ADF Desktop Integration Diagnostic Report . 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

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

In this group...	Click this button...	To...	Mode when the button is available...
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
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. See 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 information about validating a workbook, see 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

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

In this group...	Click this button...	To...	Mode when the button is available...
Test		Switch the Excel workbook from test mode to design mode. This button is active only when you are in test mode. For information about switching between design mode and test mode, see Testing Your Integrated Excel Workbook .	Test
Logging		Display a window that shows the most recent client-side log entries. See About Client-Side Logging .	Design, Test
Logging		Display the Set Output Level dialog to choose client-side log output level. See About Client-Side Logging .	Design, Test
Logging		Create a new temporary logging listener to act as a client-side log output file. See About Client-Side Logging .	Design, Test
Logging		Reload the ADF Desktop Integration configuration file. See 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 information about publishing an integrated Excel workbook, see Deploying Your Integrated Excel Workbook .	Design

 **Tip:**

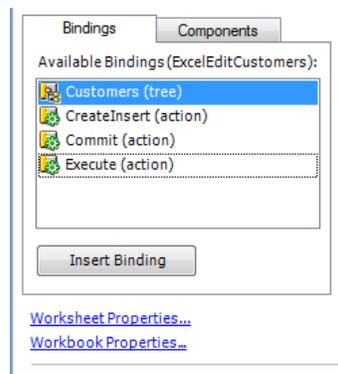
For quick and easy access, you can add **Oracle ADF** tab buttons to the Excel Quick Access toolbar.

ADF Desktop Integration Designer Task Pane

The ADF Desktop Integration Designer task pane organizes development tools into 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 can hide or show 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 and provides a brief description of each item.

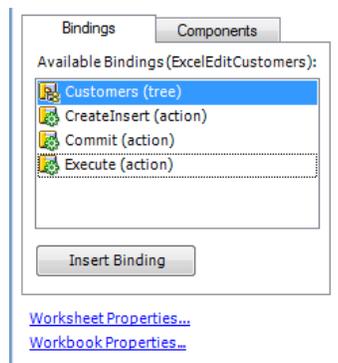
Table 5-2 Overview of ADF Desktop Integration Designer Task Pane

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. The dialog also provides access to the diagnostic report described in Generating an ADF Desktop Integration Diagnostic Report . After a successful login, it also provides access to the server's current client installer.

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 [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 prompts you to select and insert an Oracle

ADF component appropriate for 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 **Customers (tree)** binding illustrated in [Figure 5-6](#), a Select Component dialog appears where you can select and insert an ADF Table component.

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 Select Component dialog appears that prompts you 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

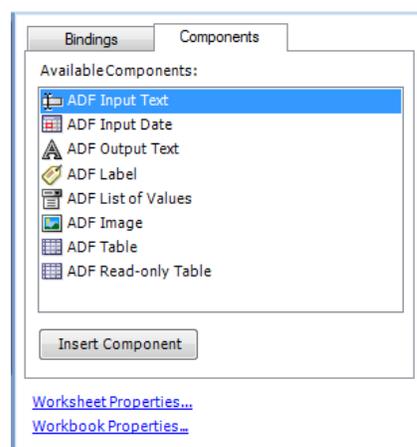
If you choose the Oracle ADF component as ADF Input Text, ADF Output Text, or ADF Label, the binding name is assigned to the **Value** property. If you choose the Oracle ADF component as ADF Ribbon Command, the binding name is assigned to the **Label** property. If you choose the Oracle ADF component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

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.

Using the Property Inspector

The property inspector is a dialog that enables you to view and edit the properties of 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 the property inspector of an ADF Table or ADF Read-only Table, double-click any cell that is part of the table.

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 ADF Label 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.

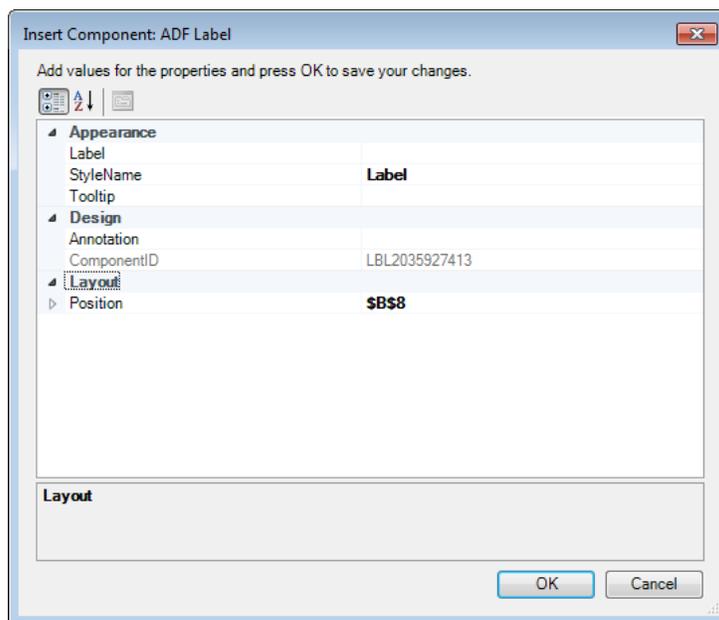
You can display the properties in an alphabetical list or in a list where the properties are grouped by categories such as Behavior, Data, and so on. [Table 5-3](#) describes the buttons that you can use to change how properties display in the property inspector.

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 Label Component



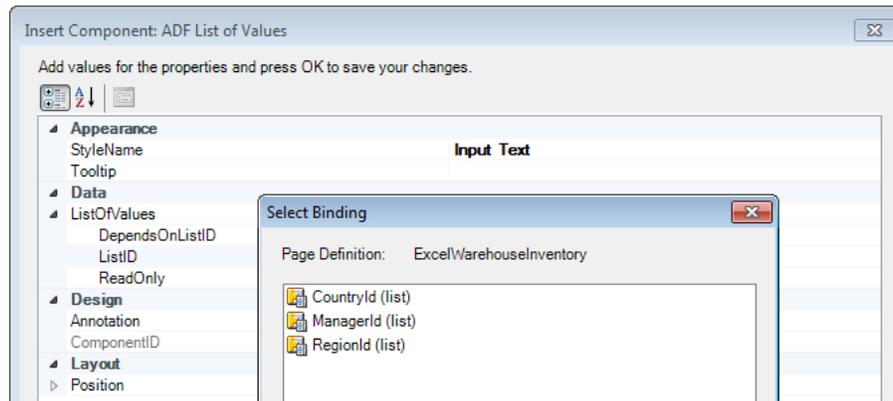
Using the Binding ID Picker

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 `ListID` property for an ADF List of Values component supports list bindings. Therefore, the binding ID picker filters the bindings from the page definition file so that only list bindings appear, as illustrated in [Figure 5-9](#).

Figure 5-9 Binding ID Picker



For information about ADF Desktop Integration component properties and the bindings they support, see [ADF Desktop Integration Component Properties and Actions](#).

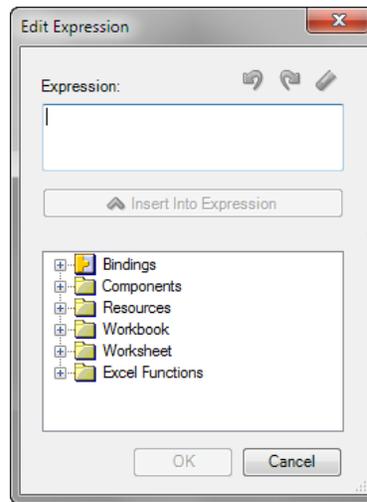
Using the Expression Builder

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. See Working with Styles .
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. See Excel's documentation.

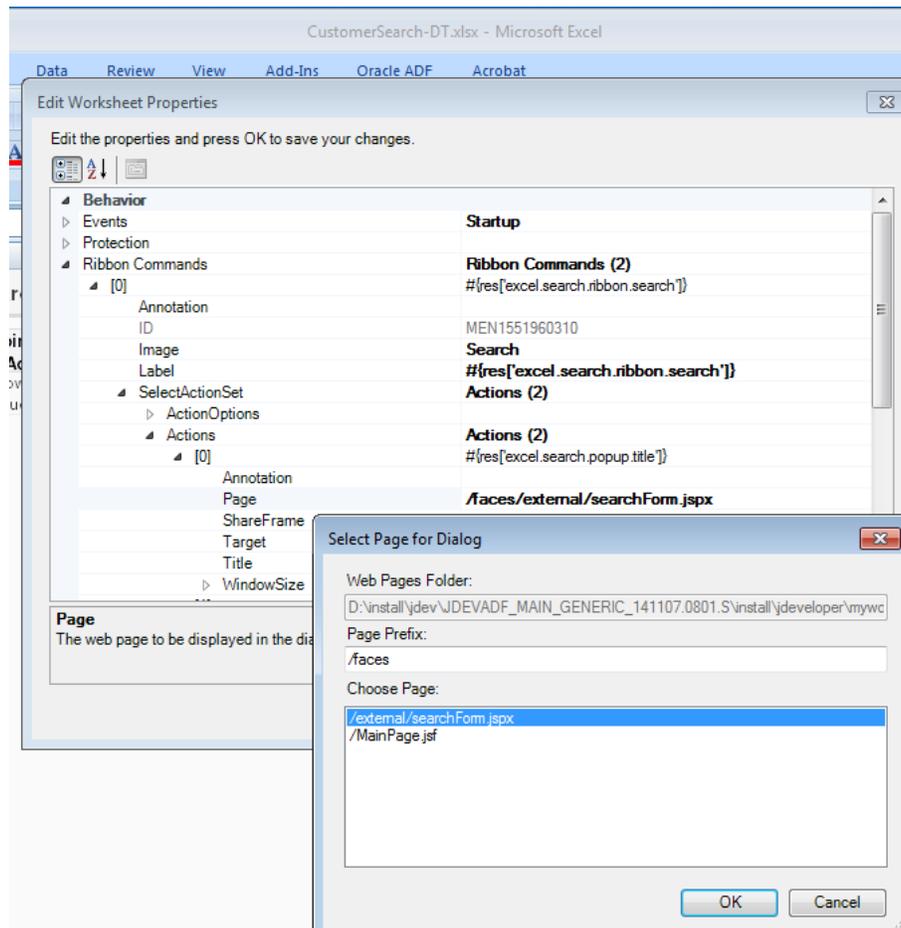
For information about using the expression builder, see [Applying Styles Dynamically Using EL Expressions](#). For information about the syntax of EL expressions in ADF Desktop Integration, and guidelines on how you write these expressions, see [ADF Desktop Integration EL Expressions](#).

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 a ribbon command, 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 information about displaying web pages in your integrated Excel workbook, see [Displaying Web Pages from a Fusion Web Application](#).

Using the File System Folder Picker

Use the file system folder picker to navigate over the Windows file system and select folders.

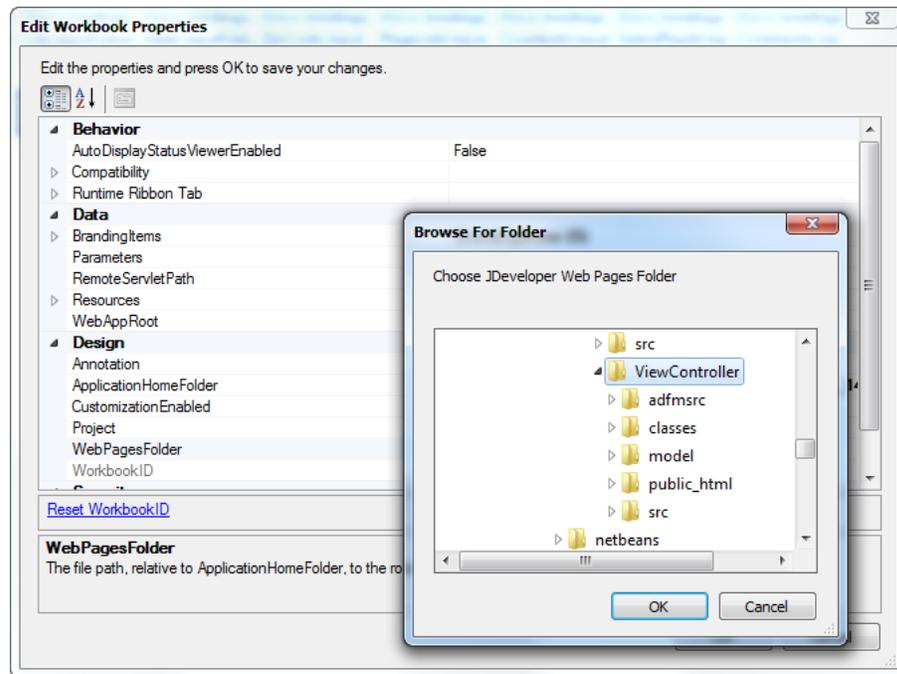
You use this picker to specify values for the following workbook properties:

- 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 information about opening an Excel workbook for the first time and the properties you set, see [How to Configure a New Integrated Excel Workbook](#).

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

Figure 5-12 File System Folder Picker

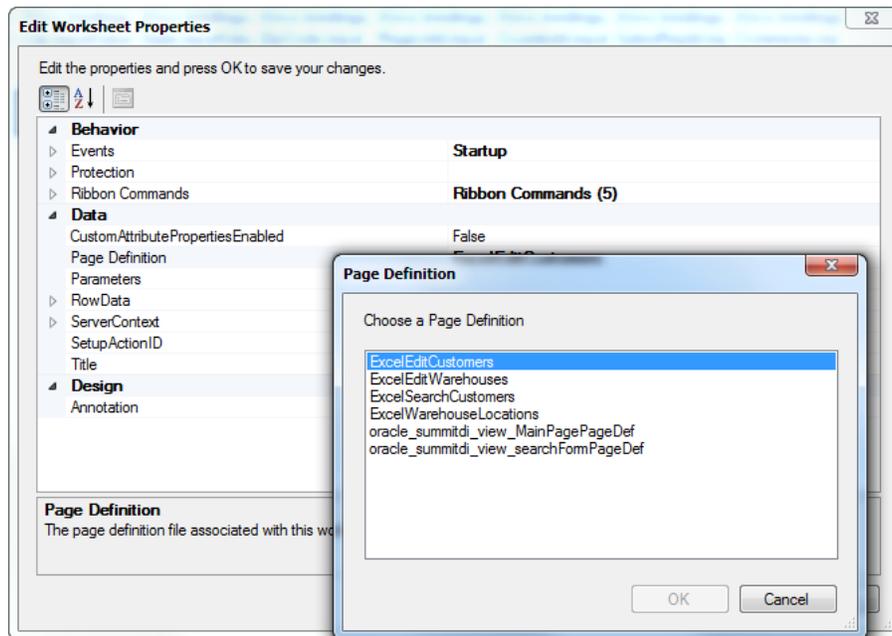


Using the Page Definition Picker

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 non-integrated 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



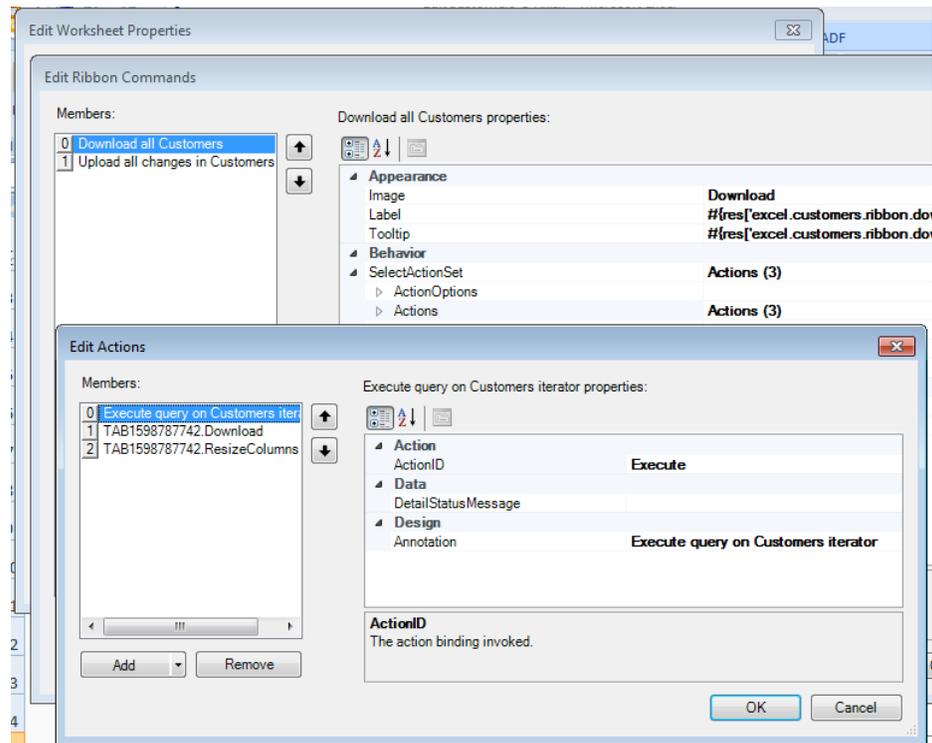
For information about page definition files, see [Working with Page Definition Files for an Integrated Excel Workbook](#).

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.

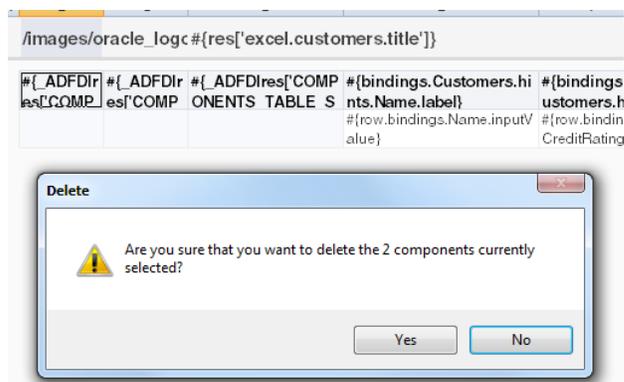
Using the Cell Context Menu

ADF Desktop Integration provides a cell context menu that enables you to edit ADF components in the integrated Excel workbook.

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 (Edit and Delete) of an ADF Table 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:

- 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.
- If you delete a cell adjacent to a cell that contains an ADF component and this shifts the latter cell into the position of the deleted cell, ADF Desktop Integration offers to delete the ADF component.
- ADF Desktop Integration context menu options are not available if multiple cells are selected when the context menu is invoked.
- After removing ADF Desktop Integration components, you should validate the integrated Excel workbook configuration in order to find any references to the deleted components. For information about validating the workbook, see [Validating the Integrated Excel Workbook Configuration](#).

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 integration metadata. For example, you might write a program to search the xml file for custom policy violations.
- 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.

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. See [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.

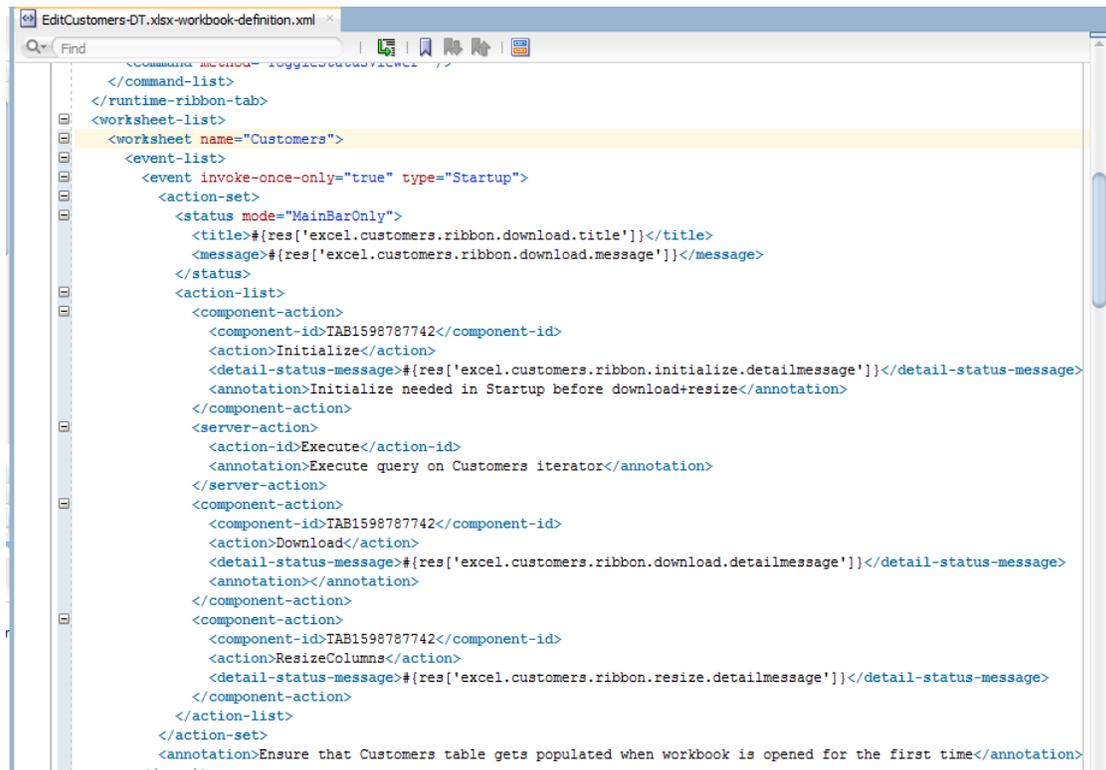
Note:

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-defintion.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 information about publishing a workbook, see [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



How to Import Workbook Integration Metadata

After editing, you can import the workbook definition file into the original workbook from which it was exported, or 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. See [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:

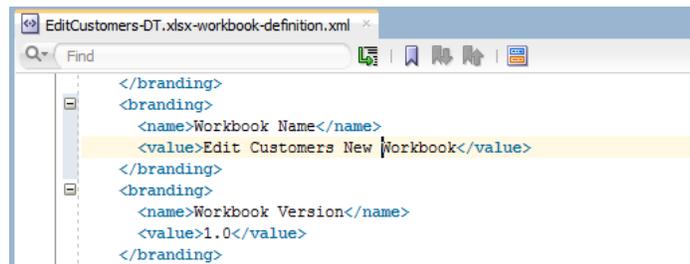
1. Open the integrated Excel workbook.
2. Click **Import** in the **Oracle ADF** tab.
The Choose a Workbook Definition file to Import dialog box appears.
3. Select the XML file that stores the workbook integration metadata, and click **Open**.

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. If you use this method to create a new (and independent) integrated Excel workbook from an existing one, make sure to reset the workbook ID after the import is complete so that the two integrated Excel workbooks do not share the same workbook ID. See [How to Reset the Workbook ID](#).

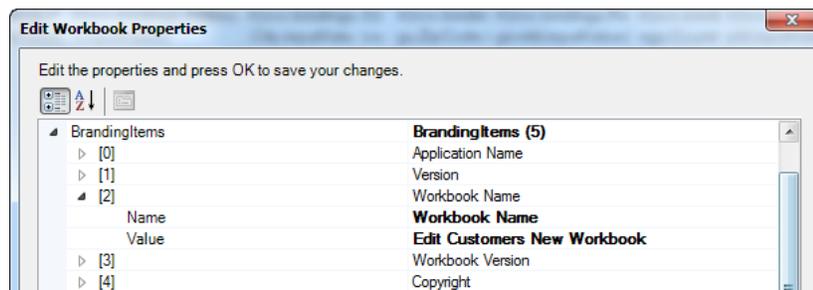
For example, [Figure 5-19](#) shows the branding value of workbook changed to `Edit Customers New Workbook` in the workbook definition file.

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



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.

6

Working with ADF Desktop Integration Form-Type Components

ADF Desktop Integration provides form components (such as labels, input and output text, and list of values) 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:

- [About ADF Desktop Integration Form-Type Components](#)
- [Inserting an ADF Label Component](#)
- [Inserting an ADF Input Text Component](#)
- [Inserting an ADF Output Text Component](#)
- [Inserting an ADF Input Date Component](#)
- [Inserting an ADF Image Component](#)
- [Displaying Output from a Managed Bean in an ADF Component](#)
- [Displaying Concatenated or Calculated Data in Components](#)

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.

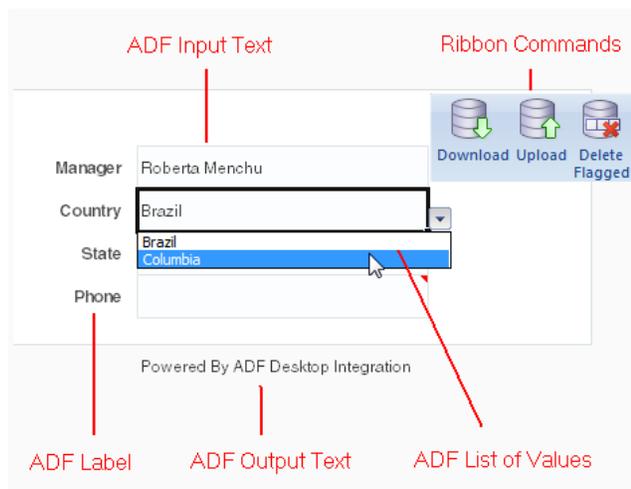
ADF Desktop Integration uses the following components to create form-type functionality in an integrated Excel workbook:

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

[Figure 6-1](#) shows some of these components. Note that the ribbon commands shown in [Figure 6-1](#) are worksheet-level ribbon commands that appear in the Excel Ribbon of your integrated Excel workbook at runtime. See [Configuring the Runtime Ribbon Tab](#).

Use of the ADF List of Values component is described in [Creating a List of Values in an Excel Worksheet](#).

Figure 6-1 ADF Desktop Integration Form-Type Components



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 the navigation form of `EditWarehouseInventory-DT.xlsx` enable end users to navigate and update data.

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

ORACLE® Edit Warehouse Inventory

Warehouse

Warehouse Id.	301	Region	Africa / Middle East
Address	6921 King Way	Country	Nigeria
City	Lagos	Manager	Ben Biri
State		Phone	
Zip Code			

Inventory

Changed	Status	Product	Amount in Stock	Reorder Point	Max in Stock	Out of Stock Explanation	Restock Date
		20510	69	40	100		
		20512	28	20	50		
		30321	85	80	140		
		30421	100	80	140		

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:

- **Lists of values:** You can use an ADF List of Values component to create a list of values in your integrated Excel workbook. See [Working with Lists of Values](#).
ADF Label or ADF Output Text components to display output from a managed bean. See [Displaying Output from a Managed Bean in an ADF Component](#).
- **Displaying output from a managed bean:** You can use ADF Label or ADF Output Text components to display output from a managed bean. See [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. See [Working with Styles](#).
- **EL Expressions:** You can use EL expressions with form-type components. See [ADF Desktop Integration EL Expressions](#).

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:

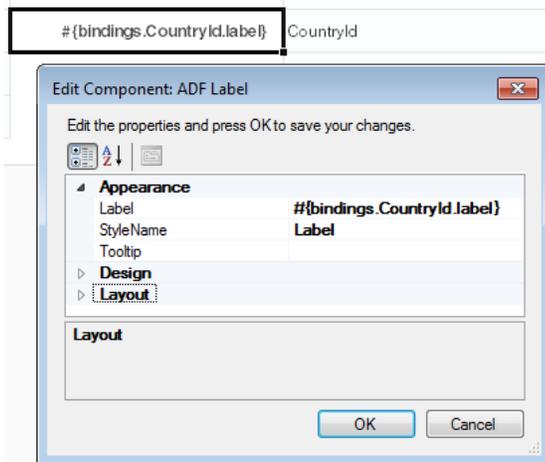
```
#{bindings.CountryId.label}
```

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 6-3](#) 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 6-3 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 6-4 shows an example of the ADF Label component (in black box) at runtime.

Figure 6-4 ADF Label Component at Runtime

ORACLE Edit Warehouse Inventory

Warehouse

Warehouse Id.	301	Region	Africa / Middle East
Address	6921 King Way	Country	Nigeria
City	Lagos	Manager	Ben Biri
State		Phone	
Zip Code			

 **Note:**

An ADF Label component renders only once, and is not updated after a call to `Worksheet.DownSync`. Consider using an ADF OutputText component instead if you want the displayed value to change after a call to `Worksheet.DownSync`.

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

Inserting an ADF Input Text Component

The ADF Input Text component is a component that you insert into the active worksheet using the components palette.

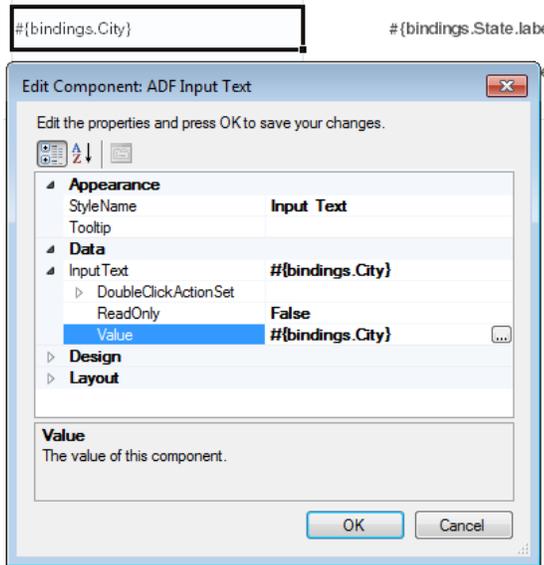
At runtime, the active cell in the worksheet where you inserted the component displays the current value from the component's binding after the worksheet `DownSync` action is invoked. End users can edit this value at runtime. Configure the worksheet `UpSync` action to transfer changes end users make to the value to the Fusion web application. Configure 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 information about `DoubleClickActionSet`, see [Using Action Sets](#).

[Figure 6-5](#) 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.

For information about the ADF Input Text component's properties, see [ADF Input Text Component Properties](#).

Figure 6-5 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 6-1](#) outlines some properties that you must specify values for.

Table 6-1 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. Note that if you specify an Excel formula in the Value property, the component behaves as if its ReadOnly property were True. The component ignores the actual value of the ReadOnly property.
InputText.ReadOnly	An EL expression that resolves to False so that changes the end user makes are uploaded. Write an EL expression that resolves to True if you want the component to ignore changes. False 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. See [Removing ADF Desktop Integration Components](#).

Figure 6-6 shows an example of the ADF Input Text component (in black box) at runtime.

Figure 6-6 ADF Input Text Component at Runtime

Warehouse	
Warehouse Id.	301
Address	6921 King Way
City	Lagos
State	
Zip Code	
Region	Africa / Middle East
Country	Nigeria
Manager	Ben Eiri
Phone	

Inserting an ADF Output Text Component

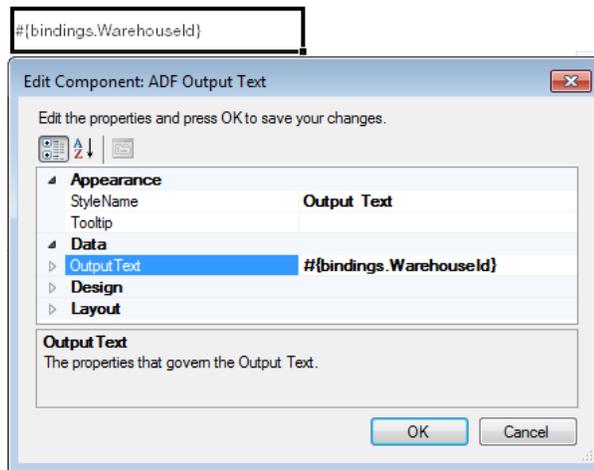
The ADF Output Text component is a component that you can insert into the active worksheet using the components palette. The active cell in the worksheet when you insert the component displays the current value from the component's binding after you invoke the worksheet `DownSync` action.

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. To prevent end users from altering the cell contents, enable automatic sheet protection in worksheet properties, as described in [Using Worksheet Protection](#).

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 6-7 shows an ADF Output Text component with its property inspector in the foreground.

Figure 6-7 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 information about the values that you specify for the properties of the ADF Output Text component, see [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. See [Removing ADF Desktop Integration Components](#).

Figure 6-8 shows an example of the ADF Output Text component (in black box) at runtime.

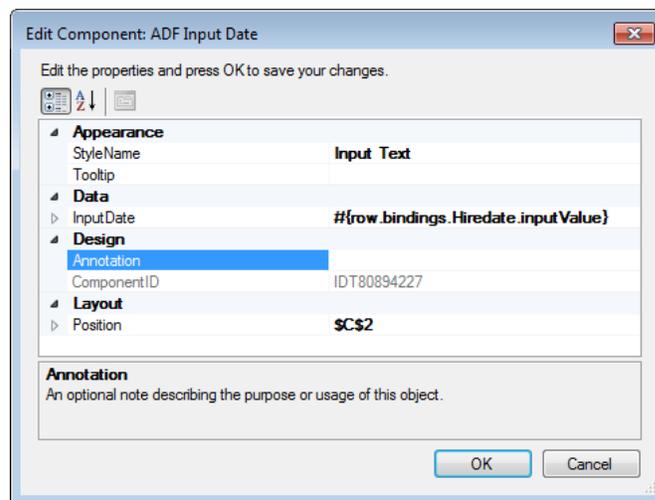
Figure 6-8 ADF Output Text Component at Runtime

Inserting an ADF Input Date Component

The ADF Input Date component displays a date picker at runtime that enables the end user to choose a date value for a date-type field.

At design-time, you can specify an attribute binding or an EL expression that resolves to a date-time value at runtime in the input field for **Value**. Other date-time values are not supported.

Figure 6-9 shows an ADF Input Date component at design-time.

Figure 6-9 ADF Input Date Component in Design Mode

You can insert the ADF Input Date component as a field in a form, or as a column in a table component. You can also add the ADF Input Date component as a model-driven column with date attribute.

To insert an ADF Input Date 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 Date** and click **Insert Component**. Alternatively, in the Oracle ADF tab, select **ADF Input Date** 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 6-2](#) outlines some properties you must specify values for. For information about the component's other properties, see [ADF Input Date Component Properties](#).

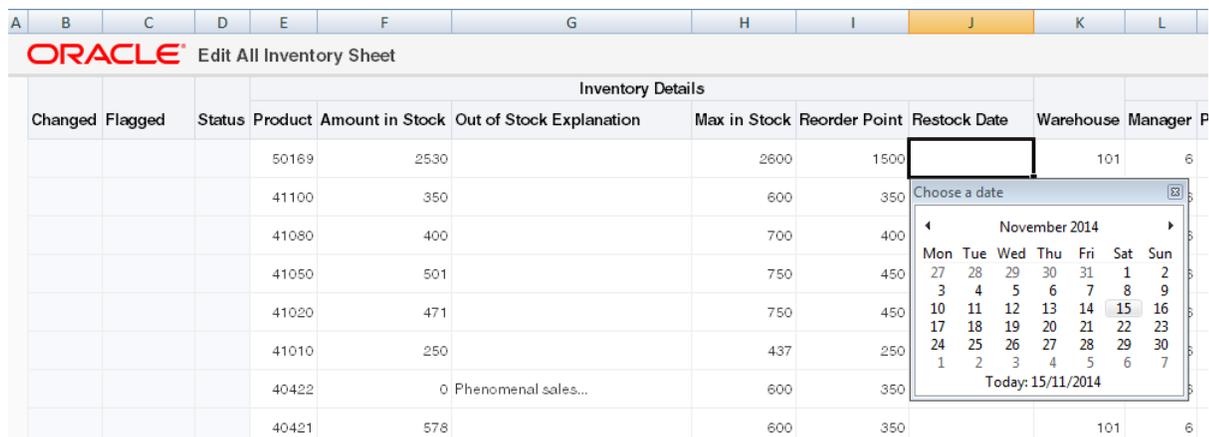
Table 6-2 ADF Input Date Component Properties

For this property...	Specify...
ReadOnly	To upload end user's changes, set this property to (or write an EL expression) <code>False</code> . To ignore end user's changes during upload, set the property to <code>True</code> . If set to <code>True</code> , the date picker does not appear at runtime. False is the default value.
Value	A date attribute. You can also specify an EL expression that resolves to a date-time value at runtime. An attribute or an EL expression that does not resolve to a date-time value at runtime will cause an error. If no date-time value is specified at design-time, the calendar shows the date that corresponds to the cell's current value at runtime. If the cell is empty (or does not contain a date value), the calendar defaults to today's date.

5. Click **OK**.

[Figure 6-10](#) shows an example of the ADF Input Date component at runtime.

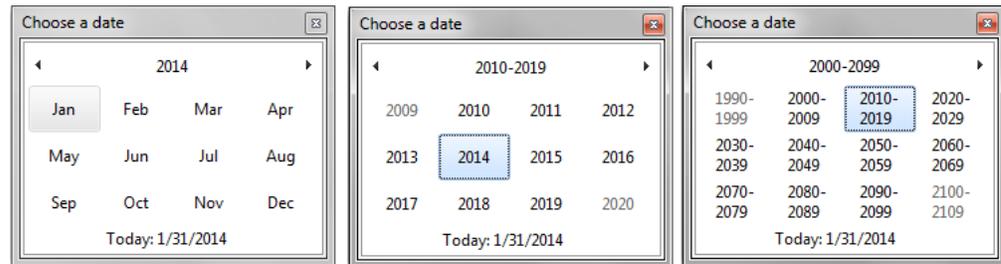
Figure 6-10 ADF Input Date Component at Runtime



At runtime, when selected, the ADF Input Date component displays a calendar in a modeless window. The end user can pick a date from the displayed month, or use the arrow icons to navigate to other months. You can also click the month or the year to navigate to another month, year, or decade (see [Figure 6-11](#)).

End users can enter a time manually in the cell that hosts the ADF Input Date component. To accept this input from your end users, configure the Excel's Format Cells properties to permit entry of a time value along with a date value in the cell that hosts the ADF Input Date component. The ADF binding type determines whether the time value will be used. By default, the time value defaults to 0:00.

Figure 6-11 Navigation in ADF Input Date Component at Runtime



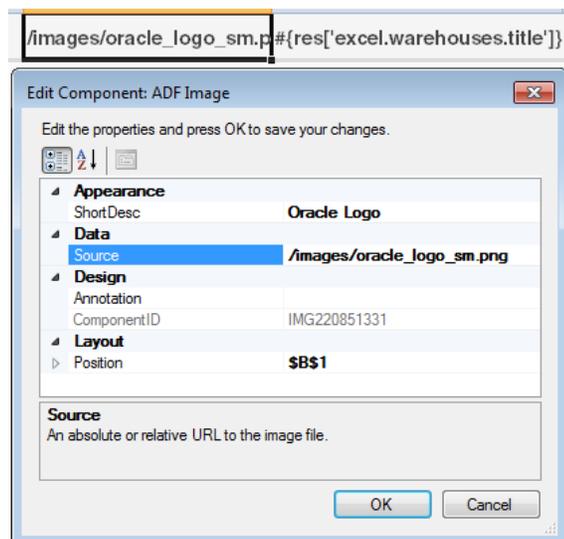
Inserting an ADF Image Component

Using the ADF Image component, you can insert an image 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 ADF Image component renders, 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 image does not render at runtime for any reason (for example, an invalid URL), the short description text that you configure at design time appears instead of the image and ADF Desktop Integration creates a log entry. The technical details regarding the failure are reported in the client logs. ADF Desktop Integrate does not interrupt the worksheet initialization and does not present a warning or error message to the end user. The end user sees the short description of the image in the cell location where the image would have displayed in the case of success.

Figure 6-12 shows an ADF Image component at design time.

Figure 6-12 ADF Image Component in 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 [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 WebAppRoot.</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 Source property does not support EL expressions.</p> <p>For the list of supported image formats, see Microsoft Excel documentation.</p>

Table 6-3 (Cont.) ADF Image component properties

For this property...	Specify...
ShortDesc	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. Note that the ShortDesc property does not support binding expressions.

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. See [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.

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.

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.

Information about how to use managed beans in a Fusion web application can be found in *Using a Managed Bean in a Fusion Web Application of Developing Fusion Web Applications with Oracle Application Development Framework*.

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. See [Displaying Output from a Managed Bean in an ADF Component](#).

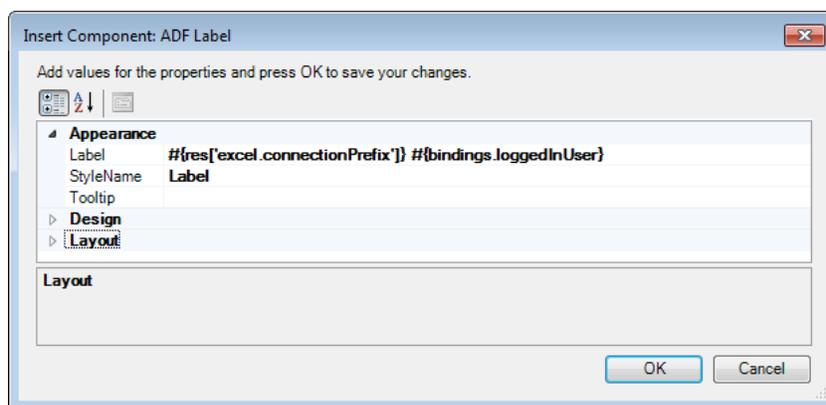
You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for ADF Desktop Integration Form-Type Components](#).

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 6-14](#) 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 6-14 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 6-14](#) 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.

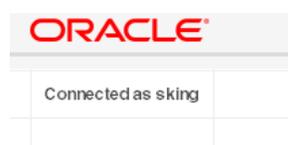
4. Click **OK**.

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 6-15](#) displays a string similar to the following at runtime:

```
Connected as sking
```

Figure 6-15 Output from a Managed Bean at Runtime



In [Figure 6-15](#), `sking` is the user name of the user that is logged on to the Fusion web application through the integrated Excel workbook.

Displaying Concatenated or Calculated Data in Components

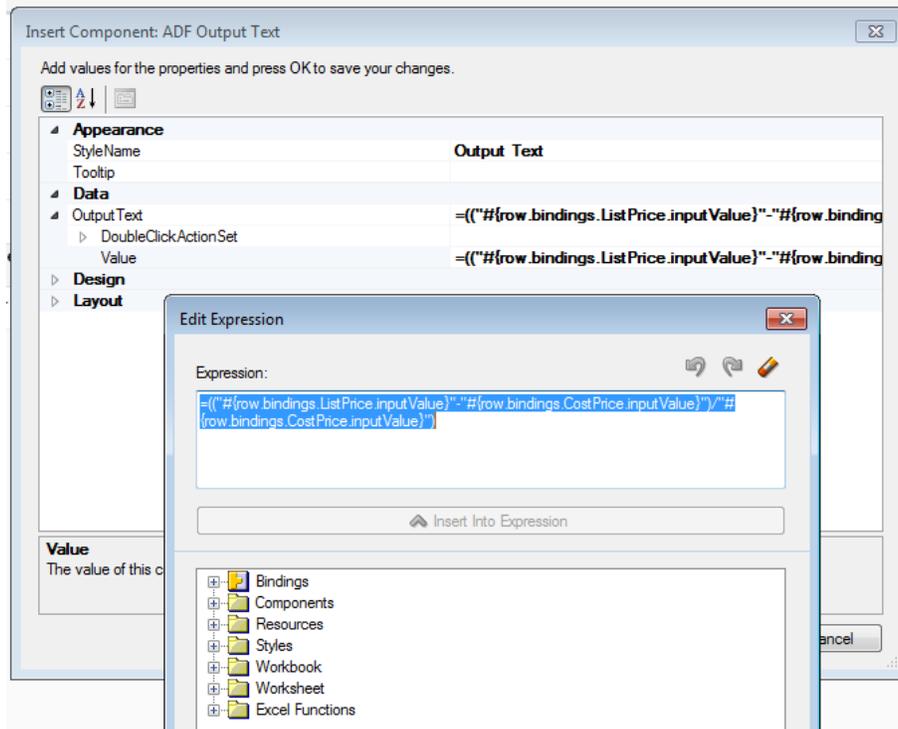
The ADF Desktop Integration module supports EL expressions within components that allow a single component to display data that is based on a calculation or concatenation of multiple binding expressions.

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 6-16](#) 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 6-16 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. See [Displaying Concatenated or Calculated Data in Components](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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.

The following example 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.

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

5. Click **OK**.

For information about EL expressions, see [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.

Using Form Components and Merged Cells

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.

7

Working with ADF Desktop Integration Table-Type Components

ADF Desktop Integration provides table-type components that you configure to download, insert, update, and delete data from the Fusion web application. This chapter includes the following sections:

- [About ADF Desktop Integration Table-Type Components](#)
- [Page Definition Requirements for an ADF Table Component](#)
- [Inserting an ADF Table Component into an Excel Worksheet](#)
- [Downloading Data to an ADF Table Component](#)
- [Downloading Pending Insert and Pending Update Rows to an ADF Table Component](#)
- [Updating Existing Data in an ADF Table Component](#)
- [Inserting Data in an ADF Table Component](#)
- [Uploading Changes from an ADF Table Component](#)
- [Uploading Changes from an ADF Table Component Using an UploadAllOrNothing Action](#)
- [Deleting ADF Table Component Rows in the Fusion Web Application](#)
- [Batch Processing in an ADF Table Component](#)
- [Special Columns in the ADF Table Component](#)
- [Configuring ADF Table Component Key Column](#)
- [Adding a Dynamic Column to Your ADF Table Component](#)
- [Adding a ModelDrivenColumnComponent Subcomponent to Your ADF Table Component](#)
- [Configuring an ADF Table Component to Resize Columns Based on Data at Runtime](#)
- [Grouping Columns Together in an ADF Table Component](#)
- [Configuring an ADF Table Component to be Read-only](#)
- [Creating an ADF Read-Only Table Component](#)
- [Limiting the Number of Rows Your Table-Type Component Downloads](#)
- [Tracking Changes in an ADF Table Component](#)
- [Evaluating EL Expressions for ReadOnly Properties](#)
- [Displaying Error Messages in an Error Message Column](#)

About ADF Desktop Integration Table-Type Components

ADF Desktop Integration provides the ADF Table component to display structured data. It provides end users with the functionality to download rows of data. It also enables end users to edit or delete downloaded data, insert new rows of data, and to upload new and edited rows of data.

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. [Figure 7-1](#) shows the ADF Table component.

Figure 7-1 ADF Desktop Integration Table-Type Components

The screenshot shows a Microsoft Excel window titled 'EditCustomers-DT.xlsx - Microsoft Excel'. The ribbon includes 'Oracle ADF' and 'Customers'. The ADF Table component is embedded in the worksheet, displaying a table with the following data:

Changed	Status	* Name	Credit Rating	Address	City	Zip Code	Country	Sales Rep.	Key
		Zebra's Bicycles	Fair	3910 Colley Ave	Norfolk	23508	USA	Magee
		Superior Bicycle	Fair	538 Superior Ave E	Cleveland	44114	USA	Magee
		Bicycle World	Fair	5300 SW 17th St	Topeka	66604	USA	Magee
		Schindler's Sports	Fair	4479 Forest Park Ave	St Louis	63108	USA	Magee
		Barry's Basketball	Fair	56 E Superior St	Chicago	60611	USA	Magee
		Gavin Sporting Goods	Fair	1935 SE Hawthorne Blvd	Portland	97214	USA	Magee
		Hot Stuff	Fair	25613 Dollar St	Hayward	94544	USA	Magee
		Acme Sporting Goods	Fair	770 4th Ave	San Diego	92101	USA	Magee

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 information about the **Key** column, see [Configuring ADF Table Component Key Column](#).

The other ADF Desktop Integration components that you can use with these table-type components are described in [Working with ADF Desktop Integration Form-Type Components](#) and [Working with Lists of Values](#).

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	Region	Country
			Unisports	Good	55-2066101	72 Via Bahia	Sao Paulo			South America	Brazil
			Simms Athletics	Poor	81-20101	Takashi	Osaka			Asia	Japan
			Delhi Sport	Excellent	91-10351	Chanakya	New Delhi			Asia	India
			Soccer-N-More	Poor		1749 W 5th Ave	Columbus	OH	43212	North America	USA
			Ball and Glove	Excellent		14527 Madison Ave	Cleveland	OH	44107	North America	USA

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 functionality that table components can use.

- **Search and Select dialog:** You can configure a `ModelDrivenColumnComponent` subcomponent in a table column, as described in [Adding a Model-Driven List Picker to an ADF Table Component](#), to display a dialog where end users can search and select data.
- **Dependent List of Values:** You can add dependent list of values components in your table component. See [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. See [Working with Styles](#).
- **Tooltips:** You can configure tooltips to display additional information or instructional text to your end users. See [Displaying Tooltips in ADF Desktop Integration Components](#).
- **EL Expressions:** You can use EL expressions with table-type components. See [ADF Desktop Integration EL Expressions](#).

Page Definition Requirements for an ADF Table Component

ADF Table components require that you expose specific control bindings in the page definition file in order to provide data-entry functionality.

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 [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 and range paging access modes for view objects. The other access modes are not supported.

Consider using the range paging access mode when your integrated Excel workbook has to download large amounts of data. See Efficiently Scrolling Through Large Result Sets Using Range Paging in *Developing Fusion Web Applications with Oracle Application Development Framework*.

If you decide to use the range paging access mode, make sure that the application's view object supports this access mode before using it with ADF Desktop Integration. For example, the view object must work properly with TOP-N queries described in Understanding How Oracle Supports "TOP-N" Queries of *Developing Fusion Web Applications with Oracle Application Development Framework*.

In addition, note that view objects with range paging access mode cannot be scrolled with unposted rows. For this reason, make sure that ADF Desktop Integration action sets commit or roll back any pending changes as expected. If pending changes are not committed or rolled back before invoking an ADF Table component's `Download` action, the application reports the following exception:

```
An attempt has been made to navigate a rowset in range paging mode when the rowset has pending changes.
```

Before inserting new rows, the iterator repositions to the first row, if necessary. This is because inserting new rows after the first row can result in unexpected scrolling. This behavior applies to the ADF Table component's `Upload` action as well as double-click action sets for insert rows.

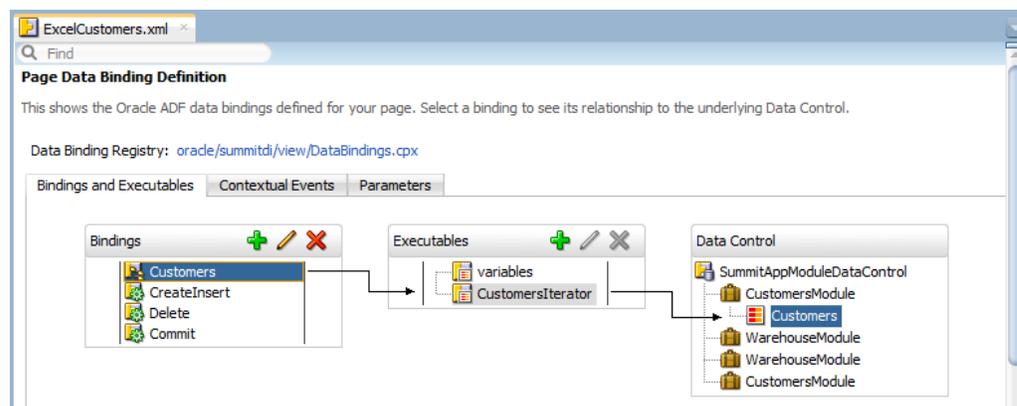
 **Note:**

Deleting rows may also result in unexpected scrolling with a view object in range paging access mode. Therefore, range paging access mode should not be used if the ADF Table component's `DeleteFlaggedRows` action is exposed in the integrated Excel worksheet.

- Method action bindings and action bindings (such as `Execute`, `Commit`, and `CreateInsert`) 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:
 - [Inserting an ADF Table Component into an Excel Worksheet](#)
 - [Inserting Data in an ADF Table Component](#)
 - [Downloading Pending Insert and Pending Update Rows to an ADF Table Component](#)

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



Inserting an ADF Table Component into an Excel Worksheet

After you configure a page definition file correctly, you can insert an 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.

How to Insert an ADF Table Component

You insert an ADF Table component using one of the following methods:

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

The following procedure describes how to insert an ADF Table component using the bindings palette. One benefit of this method over the other two is that you do not have to manually add each column that you want to appear in the component at runtime.

- In the **Oracle ADF** tab, select **ADF Table** from the **Insert Component** dropdown list.
- In the components palette of the ADF Desktop Integration Designer task pane, select **ADF Table** and click **Insert Component**.

Before you begin:

It may be helpful to have an understanding of ADF Table component. See [Inserting an ADF Table Component into an Excel Worksheet](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality of Table-Type Components](#).

To insert an ADF Table component into an 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 selecting a cell, make sure that the:

- Data of two tables do not overlap at runtime
 - 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**.
 4. In the dialog that appears, select **ADF Table** and click **OK**.

 **Note:**

- By default, the `ModelDrivenColumnComponent` subcomponent is defined as the subcomponent type for all columns when you insert an ADF Table component using the bindings palette.

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**.

- For tree bindings with multiple `<nodeDefinition>` elements (child nodes), attribute names used in the expressions for `Value` properties of `UpdateComponent` and `InsertComponent` must be unique across all `<nodeDefinitions>`.

5. Configure properties for the ADF Table component, as described in [Table 7-1](#), using the property inspector shown in [Figure 7-4](#).

Figure 7-4 ADF Table Property Inspector

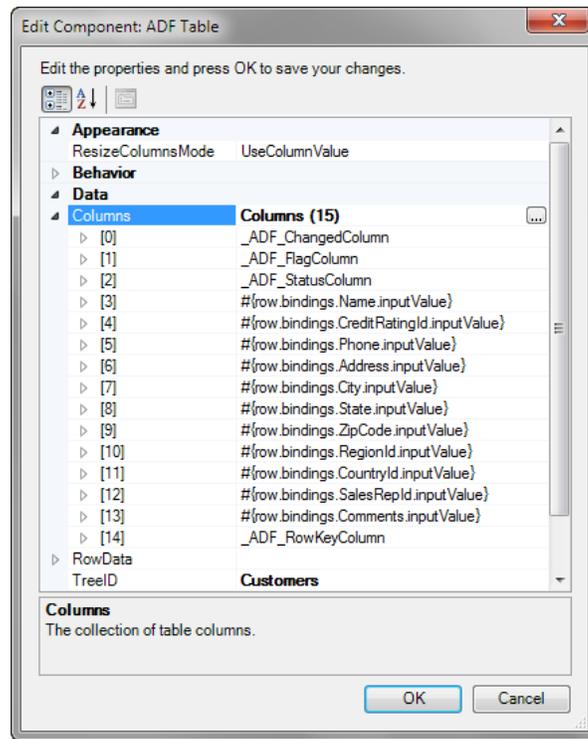


Table 7-1 ADF Table Component Properties

Set this property to...	This value...
BatchOptions.CommitBatchActionID	The Commit action binding that the page definition file exposes.
UniqueAttribute	Specify a binding expression that uniquely identifies each row in the iterator associated with the tree binding. A UniqueAttribute property value should only be specified if the tree binding's iterator does not support row keys.
RowLimit	(Optional) configure this group of properties to determine the number of rows that the ADF Table component downloads. See Limiting the Number of Rows Your Table-Type Component Downloads.

6. Click **OK**.
7. Choose the appropriate option in the Insert Component: ADF Table dialog:
 - **Yes** to create default ribbon commands for the new table to download and upload data. You can delete or edit these ribbon commands at a later time. We recommend that you change the default ribbon command label. See [How to Configure a Worksheet Ribbon Command for the Runtime Ribbon Tab](#).
 - **No** if you want to configure the download and upload of data at a later time or use one of the other available methods (for example, a worksheet event), as

described in [Downloading Data to an ADF Table Component](#), and [Uploading Changes from an ADF Table Component](#).

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	Status	* Name	Address	City	State	Zip Code	Region	Country	Sales Rep.	Key
		Zebra's Bicycles	3910 Colley Ave	Norfolk	VA	23508	North America	USA	Magee
		Superior Bicycle	538 Superior Ave E	Cleveland	OH	44114	North America	USA	Magee
		Bicycle World	5300 SW 17th St	Topeka	KS	66604	North America	USA	Magee
		Schindler's Sports	4479 Forest Park Ave	St Louis	MO	63108	North America	USA	Magee
		Barry's Basketball	56 E Superior St	Chicago	IL	60611	North America	USA	Magee
		Gavin Sporting Goods	1935 SE Hawthorne Blvd	Portland	OR	97214	North America	USA	Magee
		Hot Stuff	25613 Dollar St	Hayward	CA	94544	North America	USA	Magee
		Acme Sporting Goods	770 4th Ave	San Diego	CA	92101	North America	USA	Magee
		MoreAndMoreStuffz	3501 McKinney Ave	Dallas	TX	75204	North America	USA	Magee
		BuyMyJunk	5610 E Mockinbird Ln	Dallas	TX	75206	North America	USA	Magee

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

To remove the table component, use the Delete ribbon command. See [Removing ADF Desktop Integration Components](#).

How to Add a Column in an ADF Table Component

If you inserted the table without using the tree binding (for example, you inserted the table from the component palette) you add columns to the table to display the data for each attribute that you want to appear in the table. For example, a customers' table will have columns that displays customer name, phone, credit rating, and so on.

The procedure is the same if you want to add a column to table you inserted using the tree binding.

Before you begin:

It may be helpful to have an understanding of ADF Table component. See [Inserting an ADF Table Component into an Excel Worksheet](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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**.

ADF Desktop Integration does not limit the number of columns you can add to an ADF Table component. You can add as many columns as your version of Excel supports. However, a wide table can result in a poor user experience and slow performance. If you experience slow performance, try to reduce the number of table columns before investigating other causes. ADF Desktop Integration recommends configuring less than 30 columns per table when possible to optimize performance and user experience.

Downloading Data to an ADF Table Component

After you add an ADF Table component to a worksheet, configure an Oracle ADF component, such as a worksheet ribbon command, to invoke an action set. The action set must include the ADF Table component `Download` action among the actions that it invokes.

The number of rows that an ADF 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.

How to Download Data to an ADF Table Component

Configure a ribbon command to invoke the ADF Table component `Download` action.

Before you begin:

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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. See [Downloading Data to an ADF Table 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 information about adding a ribbon command in a worksheet, see [How to Define a Workbook Ribbon Command for the Runtime Ribbon Tab](#).

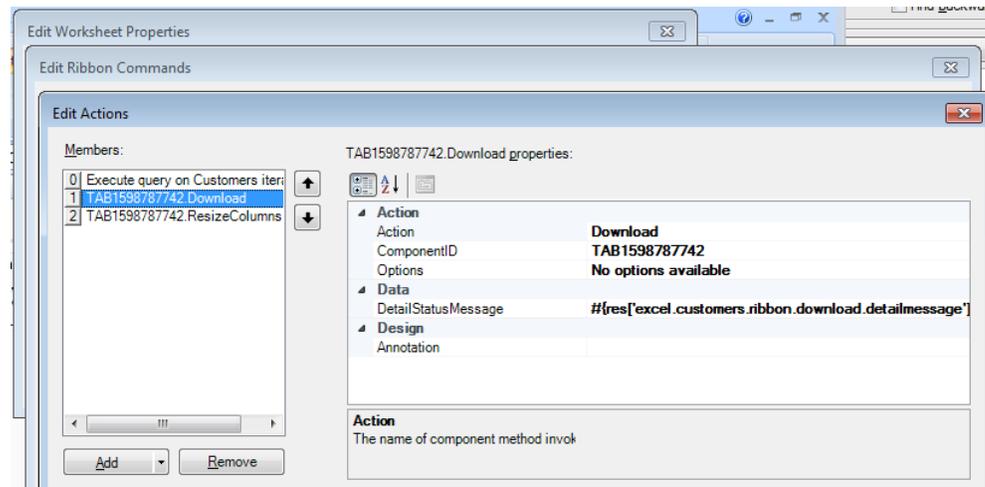
 **Note:**

Instead of adding a ribbon command, you can configure a worksheet event to invoke the action set at runtime.

3. Open the Edit Action dialog to configure an action set. For information about invoking action sets, see [Using Action Sets](#).
4. Add the following actions in the following order to the action set that invokes at runtime:
 - ADFmAction `Execute` action binding to execute the query on the iterator binding referenced by the ADF Table component `TreeID` property. This makes sure the binding is up-to-date before the action set invokes the ADF Table component `Download` action.
 - ADF Table component `Download` action.

The ADF Table component `Download` action downloads the current state of the binding referenced by the ADF Table component `TreeID` property.

[Figure 7-7](#) shows the Edit Action dialog in the `EditCustomers-DT.xlsx` workbook where the action set invoked by the **Download** ribbon command in the Excel ribbon is configured.

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

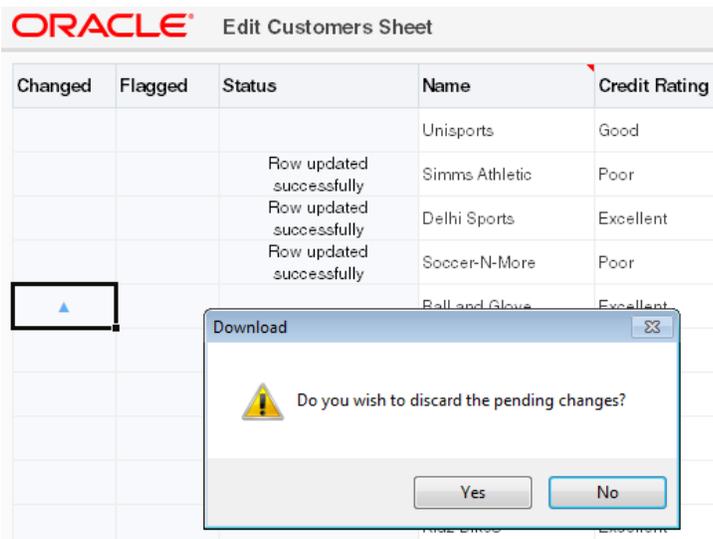
5. Click **OK**.

What Happens at Runtime: How an 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 the following points when the `Download` action is invoked at runtime:

- If any rows are 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.
- Any criteria that has been applied to the worksheet using Excel's Filter functionality is automatically cleared prior to the Upload action.
- Excel's precision for a specified number is confined to 15 significant figures. If server data has a higher precision, Excel silently changes the precision. For example, 1234567890123456789012345678901234567890 is downloaded as 1.23456789012345E+39.

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. See [Limiting the Number of Rows Your Table-Type Component Downloads](#).

Downloading Pending Insert and Pending Update Rows to an ADF Table Component

ADF Desktop Integration provides pending insert rows and pending update rows that facilitate the insertion and update of data from the 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 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 (if the `InsertUsesUpdate` column property is set to `False`), 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.

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

What Happens at Runtime: Download Action is Invoked

When the `Download` action is invoked, it examines the state of each row 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.

Using STATUS_INITIALIZED Rows for Pending Inserts

You can use `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 `STATUS_INITIALIZED` rows are not automatically removed from the iterator during download. You can configure another action to remove `STATUS_INITIALIZED` rows after download. For example, you can configure an action set with the following actions:

1. `ADFmAction` that creates `STATUS_INITIALIZED` rows
2. `Table.Download` action
3. `ADFmAction` that cleans up `STATUS_INITIALIZED` rows

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.

Updating Existing Data in an ADF Table Component

Describes how you configure an ADF Table component so that end users can edit and upload changes to existing data rows in the table.

[Uploading Changes from an ADF Table Component](#) describes how you can configure the ADF Table component so that end users can upload modified data rows.

How to Configure an ADF Table Component to Update Data

If you want the end user to be able to edit existing data, but want to restrict the addition or deletion of data rows, no additional configuration is required. Make sure that your project and the ADF Table component is configured as shown in the following procedure.

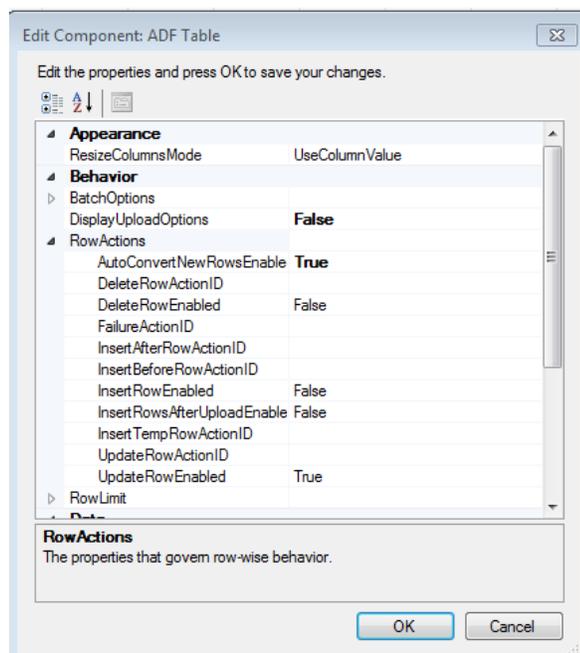
To configure an ADF Table component to update data:

1. Open the project in JDeveloper.
2. If not present, add a `Commit` action binding to the page definition file that is associated with the Excel worksheet that hosts the ADF Table component.
See [Working with Page Definition Files for an Integrated Excel Workbook](#) and [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. Make sure 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



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.

 **Note:**

You can also configure an ADF Table component to automatically refresh a row when an end user edits a value. See [Configuring Automatic Row Refresh in an ADF Table Component](#).

After updating the existing data, the end user initiates the upload process to save the changes. For information about the ADF Table component's upload process, see [Uploading 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 number of rows uploaded for each batch as well as the actions an ADF Table component invokes when it uploads and commits a batch of rows. For information about batch processing, see [Batch Processing in an ADF Table Component](#).

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

 **Note:**

Any criteria that has been applied to the worksheet using Excel's Filter functionality is automatically cleared prior to the `upload` action.

Inserting Data in an ADF Table Component

You can configure an ADF Table component to allow end users to insert new data rows.

Once you complete this task, you may want to also configure the component to allow end users to upload new and modified data rows, as described [Uploading Changes from an ADF Table Component](#).

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

To commit the changes that an end user makes in an ADF Table component, you add action bindings to the page definition file that is associated with the Excel worksheet that hosts the ADF Table component and configure a number of ADF Table component properties.

Before you begin:

It may be helpful to have an understanding of how to configure ADF Table component to insert data. See [Inserting Data in an ADF Table Component](#).

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

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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.
See [Working with Page Definition Files for an Integrated Excel Workbook](#) and [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 [Table 7-3](#):

Table 7-3 RowActions properties of ADF Table component

Set this property to...	This value...
<code>AutoConvertNewRowsEnabled</code>	True. When True, end users can edit the rows under the ADF Table component or paste new data directly into the rows under the component to convert them to rows in the ADF Table component provided that the worksheet <code>Protection.Mode</code> property is set to <code>Off</code> (the default value). For information about worksheet properties, see Worksheet Actions and Properties .
<code>InsertRowEnabled</code>	True
<code>InsertBeforeRowActionID</code>	The <code>CreateInsert</code> action binding that the page definition file exposes.
<code>InsertRowsAfterUploadEnabled</code>	True, to upload the inserted rows again regardless of whether they have been previously uploaded. By default, this property is set to <code>False</code> . The property is ignored if <code>InsertRowEnabled</code> is set to <code>False</code> .

6. Configure the `BatchOptions` properties of the ADF Table component as described in [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.

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



Note:

ADF Desktop Integration automatically sets the appropriate property values if you selected a tree binding when inserting the ADF Table component, as described in [How to Insert an ADF Table Component](#). (Optional) Review and adjust the other Columns property values as needed. See [ADF Table Component Column Properties](#).

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. Set ReadOnly to False if you do want users to edit the values in the column, set to True otherwise. The default value of the ReadOnly property is False. <p>See Adding a ModelDrivenColumnComponent Subcomponent to Your ADF Table Component.</p>

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

 **Note:**

- 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.

Uploading 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 other 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 [Uploading Changes from an ADF Table Component Using an 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. See [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.

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. See [Uploading Changes from an ADF Table Component](#) .

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality of Table-Type Components](#).

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

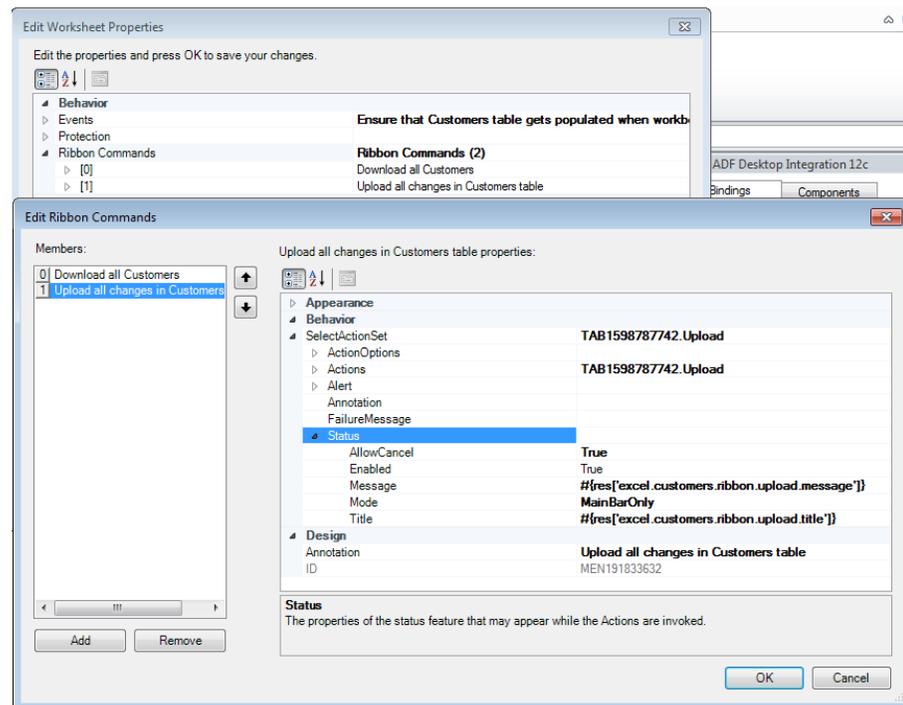
1. Open the integrated Excel workbook.
2. Open the Edit Action dialog to configure the action set that invokes the ADF Table component Upload action.

For information about action sets, see [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**.
5. If you do not want the Upload Options dialog to appear, select the cell in the Excel worksheet that references the ADF Table component and click the **Edit Properties** button in the **Oracle ADF** tab.

Set `DisplayUploadOptions` to `False` in the Table Properties dialog and 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`. See [How to Configure Batch Options for an ADF Table Component](#).

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 command, or 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. 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 information about batch options for the ADF Table component, see [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. For edited rows, position the tree binding iterator to the correct row.
- c. Set attributes from the worksheet into the model, including any cached row attribute values.
- d. For edited rows, invoke the `UpdateRowActionID` action; and for inserted rows, invoke the `InsertAfterRowActionID` action, if specified.
- e. For each uploaded row, displays a status message indicating success or failure in the `Status` column. If a row fails to upload, the `Status` column displays a message (for example, `Update Failed`). More detailed information about the failure is shown in the `Status Viewer` when the end user clicks in any cell on the row with the failure. See [Using the Status Viewer to Report Error Messages to End Users](#).

For information about the `Status` column, see [Special Columns in the ADF Table Component](#).

- f. For any row failure, the ADF Table component verifies the value of `AbortOnFail`. If `AbortOnFail` is set to `False`, it continues the upload process. Otherwise the component stops uploading data and invokes the `Commit` action.

4. 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 commits all batches successfully, it returns the success status to the executing action set. If `Table.DisplayUploadOptions` property is set to `True` and the end user has selected the **Download all rows after successful upload** option in Upload Options dialog, the ADF Table component then downloads all rows from the Fusion web application.
 - If the ADF Table component did not commit 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 the `Table.DisplayUploadOptions` property is set to `True` and the **On failure, continue to upload subsequent rows** checkbox is selected in the Upload Options dialog, the Upload action returns a success code to the action set even if some individual rows encountered validation failures.

 **Note:**

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.

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 information about change tracking, see [Evaluating EL Expressions for ReadOnly Properties](#).

What Happens at Runtime: How Row Errors Are Handled During Upload

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 error, ADF Desktop Integration reverts back to the 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.

Since ADF Desktop Integration uses savepoints to manage model state during upload, any row-level action implementations should avoid calling `commit` or `rollback` actions.

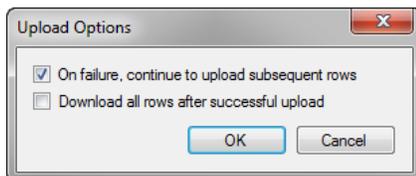
For information about savepoints, see *Using Trees to Display Master-Detail Objects in Developing Fusion Web Applications with Oracle Application Development Framework*.

What You May Need to Know About Upload Options

At runtime, when the end user uploads data from the integrated Excel workbook to the Fusion web application, ADF Desktop Integration continues to upload subsequent data rows in case of any row failure, and does not refresh or download data of all rows after a successful upload.

If required, you can enable or disable the Upload Options dialog, as shown in [Figure 7-11](#), by setting the `Table.DisplayUploadOptions` property. When `DisplayUploadOptions` is set to `True`, ADF Desktop Integration presents the Upload Options dialog.

Figure 7-11 Default Upload Dialog



 **Note:**

The `Table.DisplayUploadOptions` property is set to `True` by default in ADF Table components of integrated Excel workbooks created with versions of ADF Desktop Integration that did not include `Table.DisplayUploadOptions` property.

Using the Upload Options dialog, end users can enable or disable the following options:

- Continue to upload subsequent rows on failure. This is the default behavior. When disabled, ADF Desktop Integration aborts the upload process in case of any row failure.

- Download all data rows after a successful upload. This behavior is disabled by default.

When enabled, ADF Desktop Integration downloads the latest data from the view object cache after the successful upload.

 **Note:**

If the **Download all data rows after a successful upload** checkbox is selected, ADF Desktop Integration downloads the data from the view object cache, not from the database.

Therefore, if another user happens to update the same rows that the end user has updated, the end user will not see the updates made by the other user after downloading data rows.

If the end user clicks **Cancel** in the Upload Options dialog, 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.

You may also create a custom upload dialog. See [How to Create a Custom Upload Dialog](#).

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.

 **Note:**

You can prevent the appearance of the standard Upload Options dialog by setting the `DisplayUploadOptions` property to `False`, as described in [What You May Need to Know About Upload Options](#).

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. See [Uploading Changes from an ADF Table Component](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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 [Displaying Web Pages from a Fusion Web Application](#).

- In addition to the `ADFdi_CloseWindow` element (for example, a span element) described in [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. Ensure that you specify a boolean value for each element (`True` or `False`).

- 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 information about displaying pages from a Fusion web application, see [Displaying Web Pages from a Fusion Web Application](#).

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 information about displaying a page from the Fusion web application, see [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 [What Happens at Runtime: How the ADF Table Component Uploads Data](#).

Uploading Changes from an ADF Table Component Using an UploadAllOrNothing Action

The `UploadAllOrNothing` action provides an alternative method to upload data from the ADF Table component where the upload process commits all changed rows only if all rows are successfully uploaded.

ADF Desktop Integration commits all row changes that are successfully uploaded during a `Table.Upload` operation, even when one or more rows have failures. For example, if 100 rows are uploaded and only three rows contain failures, 97 rows are still committed to the database. For information, see [Uploading 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.

Uploading a large number of changed worksheet rows with the `UploadAllOrNothing` action can result in significant memory consumption on the application server. This is because the `UploadAllOrNothing` action commits only after all rows are processed successfully. For this reason, the `UploadAllOrNothing` action is not intended for use with large data sets. You can limit the amount of data that the `UploadAllOrNothing` action can upload using the `UploadAllOrNothing.ChangedDataLimit` servlet parameter. For information about the `UploadAllOrNothing.ChangedDataLimit` servlet parameter, see [Limiting the Amount of Changed Data That Can Be Uploaded With UploadAllOrNothing Action](#).

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. See [Uploading Changes from an ADF Table Component Using an UploadAllOrNothing Action](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality of Table-Type Components](#).

To configure an ADF component to use `UploadAllOrNothing` action:

1. Open the integrated Excel workbook.
2. Click the **Worksheet Properties** button in the **Oracle ADF** tab, and add a ribbon command that the end user uses to invoke the action set at runtime. For information about adding a ribbon command in a worksheet, see [How to Configure a Worksheet Ribbon Command for the Runtime Ribbon Tab](#).
3. Open the Edit Action dialog to configure the action set that invokes the ADF Table component actions.

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

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

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 information about how data gets uploaded during `Upload` as well as `UploadAllOrNothing`, see [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 the action 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 values in the `Changed` column remain unchanged. The `Status` column displays failure messages for the rows that contain errors, but remains empty for all rows without errors. When all rows successfully commit, the `Changed` column values are cleared and the `Status` column for the uploaded rows reports success.

Note:

- 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.
- The `UploadAllOrNothing` action treats all update and insert rows as a single batch. This means that the action bindings specified by the ADF Table component `RowData.BatchOption`'s `StartBatchActionID` and `CommitBatchActionID` properties get invoked one time per operation.

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

Uploading a large number of changed worksheet rows with the `UploadAllOrNothing` action can result in significant memory consumption on the application server. For this reason, it is not intended for use with large data sets. 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 you specify a value for this servlet parameter larger than the default, performance and scalability testing and analysis should be performed to measure the impact on the application server.

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.

To alter the limit for the amount of changed data that can be uploaded:

1. Open the `web.xml` file of your Fusion web application.
2. Add the `UploadAllOrNothing.ChangedDataLimit` servlet parameter, as described in [Table 7-7](#).

Table 7-7 Limiting the Amount of Changed Data That Can be Uploaded

Property	Value
Name	Enter the name of the servlet parameter as follows <code>UploadAllOrNothing.ChangedDataLimit</code>
Value	Specify the total amount of changed data in Kb that can be uploaded.

3. Save the `web.xml` file.
4. Rebuild and restart your Fusion web application.

Example 7-1 `web.xml` File With `UploadAllOrNothing.ChangedDataLimit` Servlet Parameter

```
<servlet>
  <servlet-name>adfdiRemote</servlet-name>
  <servlet-class>oracle.adf.desktopintegration.servlet.DIRemoteServlet</servlet-class>
  <init-param>
    <param-name>UploadAllOrNothing.ChangedDataLimit</param-name>
    <param-value>10240</param-value>
  </init-param>
</servlet>
```

[Example 7-1](#) shows the entry for `UploadAllOrNothing.ChangedDataLimit` in the Summit sample application for ADF Desktop Integration's `web.xml` file.

Deleting ADF Table Component 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

[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.

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 the `RowActions` group of ADF Table component properties, 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. See [Deleting ADF Table Component Rows in the Fusion Web Application](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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.
See [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 7-8](#).

Table 7-8 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. The specified <code>Delete</code> action binding is expected to delete the current row in the iterator.
<code>DeleteRowEnabled</code>	True to enable the ADF Table component to delete rows in the Fusion web application. False is the default value.

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

4. Click **OK**.
5. Open the integrated Excel workbook.
6. Click the **Worksheet Properties** button in the **Oracle ADF** tab, and add a ribbon command that the end user uses to invoke the action set at runtime. For

information about adding a ribbon command in a worksheet, see [How to Configure a Worksheet Ribbon Command for the Runtime Ribbon Tab](#).

7. Add the ADF Table component's `DeleteFlaggedRows` action to the list of actions that the action set invokes at runtime.

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

8. (Optional) Set the `DeleteFlaggedRows` action's `Options.AbortOnFailure` property to `False` if you want the action set to continue processing even if it encounters failures. The default value is `True`.
9. Click **OK**.

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

2. For each flagged row in the ADF Table component, the action set positions the tree binding iterator to the correct row, then it invokes the delete-type 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. For each flagged row in the ADF Table component, if the delete-type action binding specified by `RowActions.DeleteRowActionID` fails, the next event depends on the value you specified for the `DeleteFlaggedRows` action's `Options.AbortOnFailure` property. If `False`, the action set attempts to delete all flagged rows without stopping at the first failure it encounters. If the action set fails to delete a flagged row, that row:
 - Remains in the ADF Table component
 - Is marked as `Failed` in the ADF Table component's `Status` column
 - Is skipped while the action set commits the batch of successfully deleted flagged rows
 - Remains flagged in the `Flagged` column cell

If the `DeleteFlaggedRows` action's `Options.AbortOnFailure` property is set to `True` (the default value), the ADF Table component stops invocation of the `DeleteFlaggedRows` action.

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-type 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 relevant error messages are available in the Status Viewer. See [Using the Status Viewer to Report Error Messages to End Users](#).

 **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.

Batch Processing in an ADF Table Component

The ADF Table component's `Upload` and `DeleteFlaggedRows` actions both commit changes in batches rather than row-by-row in order to optimize performance and scalability. 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.

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 [ADF Table Component Properties and Actions](#).

Before you begin:

It may be helpful to have an understanding of how ADF Table components upload data, delete data, and batch process both tasks. See [What Happens at Runtime: How the ADF Table Component Uploads Data](#), [What Happens at Runtime: How the ADF Table Component Deletes Rows in a Fusion Web Application](#), and [Batch Processing in an ADF Table Component](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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 7-9 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.
<code>LimitBatchSize</code>	True When <code>True</code> , the ADF Table component processes rows in batches determined by the value of <code>BatchSize</code> . <code>True</code> is the default value. When <code>False</code> , the ADF Table component uploads all modified rows in a single batch. Note that it is not sufficient to set this property to <code>False</code> if you want the ADF Table component to upload all rows or no rows in the case of failure. Instead, you need to invoke the <code>UploadAllOrNothing</code> action, as described in Uploading Changes from an ADF Table Component Using an UploadAllOrNothing Action .
<code>StartBatchActionID</code>	(Optional) 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.

Troubleshooting Errors While Uploading Data

End users may see 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`. Rows from a batch that experiences a failure executing the action specified by the `CommitBatchActionID` display the details of the failure in the Status Viewer. Any rows in the batch that had changes posted successfully on the server before the failure show `Batch Failed` in the Status column.

Errors that occur during the commit action might continue to be reported on subsequent batch commit actions, even though subsequent batches of records do not contain errors. 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.

Special Columns in the ADF Table Component

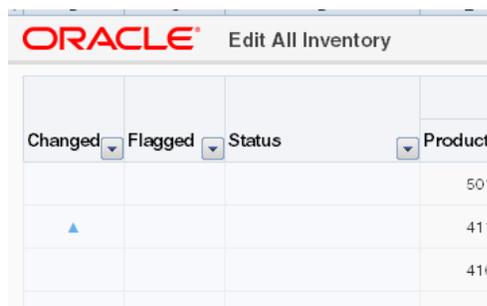
By default, the ADF Table component includes some columns when you insert an ADF Table component in a worksheet. You can retain or remove these columns, if required.

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 7-12](#) shows an example.

Figure 7-12 Changed Column in an ADF Table Component



Changed	Flagged	Status	Product
			501
▲			411
			410
			...

 **Note:**

If the end user does not want the ADF Table component's `Upload` action to upload changes in the rows marked by this column, the user must clear the entry that appears in the corresponding cell.

When an ADF Table component invokes its Upload action, it uploads all rows with non-empty cells in the `_ADF_ChangedColumn` column. See [Uploading Changes from an ADF Table Component](#).

- `_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 information about the use of this column, see [Row Flagging in an ADF Table Component](#).

 **Note:**

By default, the solid circle character indicates a row flagged for flagged-row processing. However, any nonempty cell in a `_ADF_FlagColumn` flags the corresponding row for flagged-row processing.

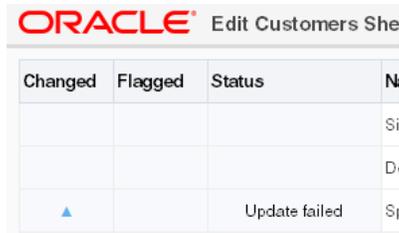
- `_ADF_StatusColumn`

This column reports the results of invocation of ADF Table component actions, such as `DeleteFlaggedRows` and `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 7-13](#) shows an example of a Status column message for a row where an update failed. More detailed information about status appears in the Status Viewer, as described in [Using the Status Viewer to Report Error Messages to End Users](#).

Figure 7-13 Status Column in an ADF Table Component



Changed	Flagged	Status	Name
			Si
			D.
▲		Update failed	Si

- `_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. Do not remove the **Key** column because it is required for the proper functioning of the ADF Table component. You can configure its appearance-related properties.

For information about the `_ADF_RowKeyColumn`, see [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-12](#). It reads the values for properties related to style and appearance, for example `CellStyleName` and `HeaderStyleName`.

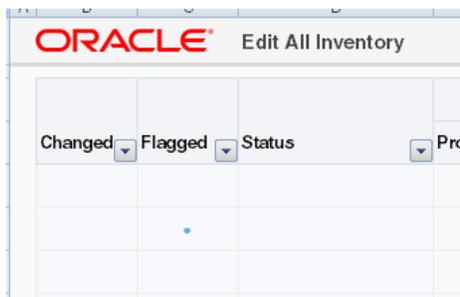
Row Flagging in an ADF Table Component

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 7-14](#) shows an example of a flagged column.

Figure 7-14 Flagged Column in ADF Table Component



Note:

By default, the solid circle character indicates a row flagged for flagged-row processing. However, any nonempty 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.

 **Note:**

- 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 information about these component actions, see [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 information about configuring an action set to invoke a component action, see [How to Invoke Component Actions in an Action Set](#).

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.

How to Configure the Key Column

You can configure the **Key** column's position, style properties, and header label. By default, the `Key Cell` 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. See [Configuring ADF Table Component Key Column](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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 and the values of properties that determine the appearance of the column (`Label`, `Tooltip`, and `Style`).
7. Click **OK** to close Edit Columns dialog.
8. Click **OK** to close the Edit Component: ADF Table dialog.

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. See [Configuring ADF Table Component Key Column](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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 7-10](#). For information about adding a column, see [How to Add a Column in an ADF Table Component](#).

Table 7-10 Key Column Properties

Set this property...	To ...
<code>CellStyleName</code>	Key Cell
<code>HeaderStyleName</code>	Column Header

Table 7-10 (Cont.) Key Column Properties

Set this property...	To ...
DynamicColumn	False
HeaderLabel	<code>#{_ADFDIres[COMPONENTS_TABLE_ROWKEY_COL_LABEL]}</code>
ID	<code>_ADF_RowKeyColumn</code>
InsertUsesUpdate	True
UpdateComponent	OutputText The Value property must be empty.
Visible	True

If desired, change the position of the column using the **Up** and **Down** arrow keys and the values of properties that determine the appearance of the column (`Label`, `Tooltip`, and `Style`).

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.

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:

- `ModelDrivenColumnComponent`
- Input Text
- Output Text

 **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. See [Adding a ModelDrivenColumnComponent Subcomponent to Your ADF Table Component](#) and [Creating a List of Values in an ADF Table Component Column](#).

Note:

In cases where the ADF Table component uses a tree binding containing multiple `<nodeDefinition>` elements, model-driven lists used in dynamic columns must have unique names across all nodes.

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}
```

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 returns the selected attributes from the underlying declarative SQL mode view object to the integrated Excel worksheet.

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.

Typically the set of matching attributes does not change between invocation of the ADF Table component's `Download` and `Upload` actions. However, if previously downloaded attributes no longer exist in the tree binding when the ADF Table component invokes the `Upload` action, the integrated Excel workbook prompts the end user to determine if the end user wants to continue to upload data. For information about how to avoid the scenario just described (downloaded attributes no longer exist in the tree binding), see [Using an Integrated Excel Workbook Across Multiple Web Sessions](#).

Note:

The ADF Table component ignores the value of a column's `Visible` property when you configure a column to be dynamic. For information about ADF Table component column properties, see [Table A-11](#).

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.

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}"
```

How to Specify Styles for Dynamic Columns

If the same style can be applied for all expanded columns, specify the literal style name for the `CellStyleName` property of a dynamic column.

However, if different styles are needed for different expanded columns, an EL expression must be specified for the `CellStyleName` property of a dynamic column.

You can specify different styles for each attribute using a custom attribute property, for example, `adfdiCellStyle`. The following syntax would be used for the `CellStyleName` EL expression:

```
#{bindings.TreeID.[TreeNodeID].hints.*.adfdiCellStyle}
```

For information about custom attribute properties, see [Using ADF Desktop Integration EL-based Properties with Custom Attribute Properties](#).

Alternatively, you can specify different styles for each attribute using more complex EL expressions to compute the style name.

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")
```

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

Adding a ModelDrivenColumnComponent Subcomponent to Your ADF Table Component

The ModelDrivenColumnComponent is the default subcomponent when you insert an ADF Table component. The column subcomponent type is determined at runtime by the column's attribute Control Type hint specified 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 containing the model-driven list items.

Note:

- To use the (optional) date picker for a model-driven column with a date attribute, set the `Compatibility.TableComponents.ModelDrivenColumns.DatePickerEnabled` property to `True`. See [ADF Desktop Integration Compatibility Properties](#).
- 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 `combo_lov`, then the column uses an Input Text subcomponent.
- In a dependent list of values implementation, ADF Desktop Integration determines if each list subcomponent depends on another model-driven list when an ADF Table component uses multiple ModelDrivenColumnComponent subcomponents. To do this, it verifies that the bind variable specified for a list references an attribute bound to another list. See [Table 8-1](#).

For information about creating a model-driven list, see *How to Create a Model-Driven List* in *Developing Fusion Web Applications with Oracle Application Development Framework*.

Configuring an ADF Table Component to Resize Columns Based on Data at Runtime

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.

Resizing a column's width at runtime is a two-step process. First, you configure the table column with the desired width-related property values. Secondly, add the ADF Table component's `ResizeColumns` action to the desired action set. Typically, you add this action after the ADF Table component's `Download` action in the action set. The `EditCustomers-DT.xlsx` workbook in the Summit sample application, described in [Introduction to the ADF Desktop Integration Sample Application](#), demonstrates this implementation.

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.

Before you begin:

It may be helpful to have an understanding of configuring resizing behavior of ADF Table columns. See [Configuring an ADF Table Component to Resize Columns Based on Data at Runtime](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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 [Table 7-11](#):

Table 7-11 `ResizeColumnsMode` Values of the ADF Table Component

Value	Description
<code>UseColumnValue</code>	Default. All columns in the table will be resized (or not) based on their <code>Column.ResizeMode</code> property values. Columns with <code>InheritFromTable</code> will not be resized. Individual columns that have <code>Column.ResizeMode</code> properties set to a value other than <code>InheritFromTable</code> are resized accordingly.
<code>AutoFitAllWithHeader</code>	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. Individual columns that have <code>Column.ResizeMode</code> properties set to a value other than <code>InheritFromTable</code> are resized accordingly. Note that values in the column's cells above or below the table are not considered when finding the best fit.

Table 7-11 (Cont.) ResizeColumnsMode Values of the ADF Table Component

Value	Description
AutoFitAllWithoutHeader	<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>

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 [Table 7-12](#):

Table 7-12 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	<p>Default. The column is resized based on the table's <code>ResizeColumnsMode</code> setting.</p> <p>If <code>ResizeColumnsMode</code> is set to <code>UseColumnValue</code>, then no resizing occurs.</p>
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.

5. If the ADF Table component's `ResizeColumnsMode` property is set to `UseColumnValue` and a column's `ResizeMode` property is set to `SpecifiedWidth`, set the `Column.Width` property to the number of characters you want to display in the column.

A 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. An example EL expression for `width` that makes use of the UI Hint `displayWidth` for an attribute is:

```
#{bindings.Customers.hints.Name.displayWidth}
```

 **Note:**

- If the expression cannot be evaluated, or if the expression evaluates to less than 1 or greater than 255, the `ResizeMode` is considered to be `Manual` and the column is not resized.
- Use a decimal point regardless of the environment's Region and Language settings if you want to specify a fractional value. A 'decimal comma' (such as is seen in French locales) is not supported.

6. Click **OK**.

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 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. See [Configuring an ADF Table Component to Resize Columns Based on Data at Runtime](#) and [Using Action Sets](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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, as described in [How to Define a Workbook Ribbon Command for the Runtime Ribbon Tab](#).

 **Note:**

Instead of adding a ribbon command, you can configure a worksheet event to invoke the action set at runtime.

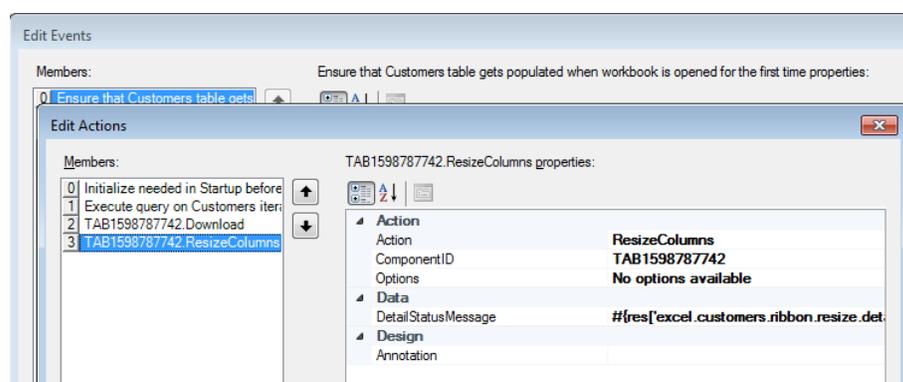
3. Open the Edit Action dialog to configure an action set. For information about invoking action sets, see [Using Action Sets](#).
4. Add the ADF Table component `ResizeColumns` action to the list of actions that the action set invokes at runtime. Note that `ResizeColumns` is a component action.
5. Click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

Figure 7-15 shows the `ResizeColumns` action at design-time that is configured in the worksheet `Events` property of the `EditCustomers-DT.xlsx` workbook.

 **Note:**

If you configure an action set that is invoked by the worksheet `Startup` event and this action set invokes the ADF Table component's `ResizeColumns` action after the `Download` action, make sure that the action set invokes the ADF Table component's `Initialize` action before the `Download` action. Figure 7-15 demonstrates this configuration.

Figure 7-15 `ResizeColumns` Action



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 [How to Configure an Action Set to Resize Columns of an ADF Table Component at Runtime](#)).

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 resize columns at runtime, all ADF Table components attempt to resize their columns independently. However, the ADF Table component'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

 **Note:**

- 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 7-15](#) for an example.
- Resizing the columns for a large table may take a significant amount of time. The end user may perceive a download to be slower due to this extra work. Be sure to test your workbook with typical data loads to determine whether resizing is worth the delay for your use case.
- 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, explicitly setting the row height of table cells at design time, or removing the **Wrap Text** setting from the style may produce better results.
- It may help to make Excel columns wider at design time if you use one of the `AutoFit` resizing modes and want to avoid text wrapping at runtime. This is due to the way that Excel's `AutoFit` resizing modes work.

Grouping Columns Together in an ADF Table Component

You can render group headers for columns that render in an ADF Table component to, for example, provide your end users with a more intuitive interface by using descriptive labels for groups of columns.

[Figure 7-22](#) shows an example where the `EditAllInventory-DT.xlsx` workbook from the Summit sample application for ADF Desktop Integration groups the **Product** to **Restock Date** columns into an **Inventory Details** group header while the **Warehouse** to **Country** columns have been grouped into a **Warehouse Details** group of columns.

Figure 7-16 Providing a Grouping Header for Columns in an ADF Table Component

The screenshot shows a Microsoft Excel spreadsheet with an ADF Table component. The table has two main sections: 'Inventory Details' and 'Warehouse Details'. The 'Inventory Details' section has columns for 'Out of Stock Explanation', 'Max in Stock', 'Reorder Point', and 'Restock Date'. The 'Warehouse Details' section has columns for 'Warehouse', 'Manager First Name', 'Manager Last Name', 'Address', and 'City'. The table data is as follows:

Inventory Details				Warehouse Details				
Out of Stock Explanation	Max in Stock	Reorder Point	Restock Date	Warehouse	Manager First Name	Manager Last Name	Address	City
590	2600	1500		101	Molly	Urguhart	283 King Street	Seattle
350	600	350		101	Molly	Urguhart	283 King Street	Seattle
400	700	400		101	Molly	Urguhart	283 King Street	Seattle
501	750	450		101	Molly	Urguhart	283 King Street	Seattle
471	750	450		101	Molly	Urguhart	283 King Street	Seattle
250	437	250		101	Molly	Urauhart	283 King Street	Seattle

ADF Desktop Integration implements the functionality shown in [Figure 7-16](#) by rendering an extra table header row above the ADF Table component's regular table header row at runtime. It renders this extra table header row if you configure the `GroupHeader` properties in one of the column definitions of the ADF Table component. You can also implement this functionality for dynamic columns. If you want to implement this functionality for dynamic columns, you must define custom attributes, as described in [How to Group Columns that Render in a Dynamic Column](#). For information about how to configure the `GroupHeader` properties for static and dynamic columns, see [How to Group Columns in an ADF Table Component](#).

Leave the row above the ADF Table component empty of data and styles if you want to render an extra table header row to group columns because, at runtime, existing data and styles will be overwritten to render the extra table header row.

How to Group Columns in an ADF Table Component

You group columns in an ADF Table component by configuring the `GroupHeader` properties for the start and end columns in the group of columns.

Before you begin:

It may be helpful to have an understanding of how you can group columns in an ADF Table component. See [Grouping Columns Together in an ADF Table Component](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality of Table-Type Components](#).

Define custom attribute properties, as described in [How to Group Columns that Render in a Dynamic Column](#), if you want to group header columns that render in a dynamic column. This step is not required if you want to group header columns in static columns.

To group columns 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.

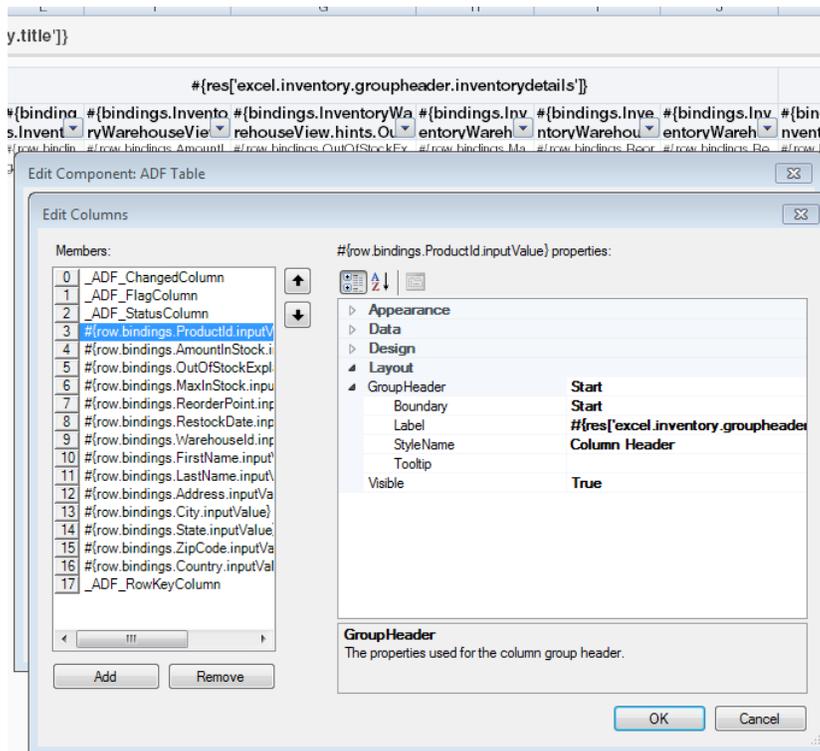
4. In the Edit Columns dialog, select the column that you want to start the group of columns.
5. In the right pane of the Edit Columns dialog, expand the **GroupHeader** property under the **Layout** field and configure the properties, as described in [Table 7-13](#).

Table 7-13 GroupHeader Properties of an ADF Table Component's Group Start Column

Set this property to ...	This value
Boundary	Set to <code>start</code> or an EL expression that evaluates to <code>start</code> . This defines the column as the start of a group of columns. If you want to define a dynamic column as the start of a group of columns, we recommend that you define a custom attribute property with a value of <code>start</code> , as described in How to Group Columns that Render in a Dynamic Column . Write an EL expression that retrieves the value of this custom attribute property.
Label	Set this property to a string or to an EL expression that evaluates to a label in the column group header at runtime. The evaluated value renders in the cell of the start column. The start column in the column group header requires a value for this property. No column group header forms at runtime if you do not specify a value. For information about labels, see Using Labels in an Integrated Excel Workbook . You can also edit this property by editing the Excel cell where the label text appears. Editing the text in the cell directly only affects this property. There is no effect on the <code>Boundary</code> property.
StyleName	Set this property to a style defined in the workbook or to an EL expression that evaluates to a style name. The named style is applied to the column group's header cell at runtime. For information about styles, see Configuring the Appearance of Your Integrated Excel Workbook .
Tooltip	(Optional) Specify a tooltip. The tooltip that you specify renders from the extra table header cell of the end column in the group. For information, see What Happens at Runtime: How an ADF Table Component Groups Columns . For information about tooltips, see How to Add a Tool Tip to an ADF Table Component .

[Figure 7-17](#) shows the values configured for the `GroupHeader` properties to start the **Inventory Details** group of columns in `EditAllInventory-DT.xlsx` workbook shown in [Figure 7-16](#).

Figure 7-17 Configuring the Start Column of a Group of Columns



6. In the Edit Columns dialog, select the column that you want to end the group of columns.

 **Note:**

If you do not configure a column to end the group of columns, the column that you configured as the start column in Step 5 renders as a single-column group.

7. In the right pane of the Edit Columns dialog, expand the **GroupHeader** property under the **Layout** category and configure the following property:
 - **Boundary:** Set to end or an EL expression that evaluates to end.
This defines the column as the end of a group of columns.
If you want to define a dynamic column as the end of a group of columns, define a custom attribute property with a value of end, as described in [How to Group Columns in an ADF Table Component](#) and write an EL expression that retrieves the value of this custom attribute property.
 - Do not set values for the remaining properties in **GroupHeader**. The values that you set for the start column in Step 5 determine the label, style and tooltip that render in the group of columns' header at runtime.

 **Note:**

You do not need to set properties for the columns between the start and end columns.

If, at runtime, the integrated Excel workbook does not find a start column to the left of the column that you configure as the end column, the value that you specify for the end column is ignored.

8. Click **OK**.

 **Note:**

Remember to leave the row above the ADF Table component empty of data and styles if you want to group columns because, at runtime, existing data and styles will be overwritten to render the extra table header row that will appear above the ADF Table component.

How to Group Columns that Render in a Dynamic Column

A dynamic column can expand to more than a single worksheet column at runtime. At runtime, the integrated Excel workbook evaluates EL expressions defined for the `GroupHeader` properties after the dynamic column expands. Depending on the results of evaluating the `GroupHeader` properties, column groups form and the integrated Excel workbook renders grouped headers for the dynamic column.

To configure `GroupHeader` properties for columns that render in a dynamic column, you first define custom attribute properties on the view object attributes that render in the dynamic column's columns at runtime.

Define custom attribute properties for the attributes that you want to render the start and end boundaries of the grouped header in the dynamic column at runtime. Configure custom attribute properties for the start attribute that the `GroupHeader` `Boundary`, `Label`, `StyleName`, and `Tooltip` properties reference at runtime using EL expressions. Configure a custom attribute property for the end attribute that the `GroupHeader` `Boundary` property references at runtime using an EL expression.

For information about defining custom attribute properties, see [Using ADF Desktop Integration EL-based Properties with Custom Attribute Properties](#).

Figure 7-18 shows an attribute (Address) that defines custom attribute properties to start a grouped header.

Figure 7-18 Custom Attribute Property to Start a Grouped Header in a Dynamic Column

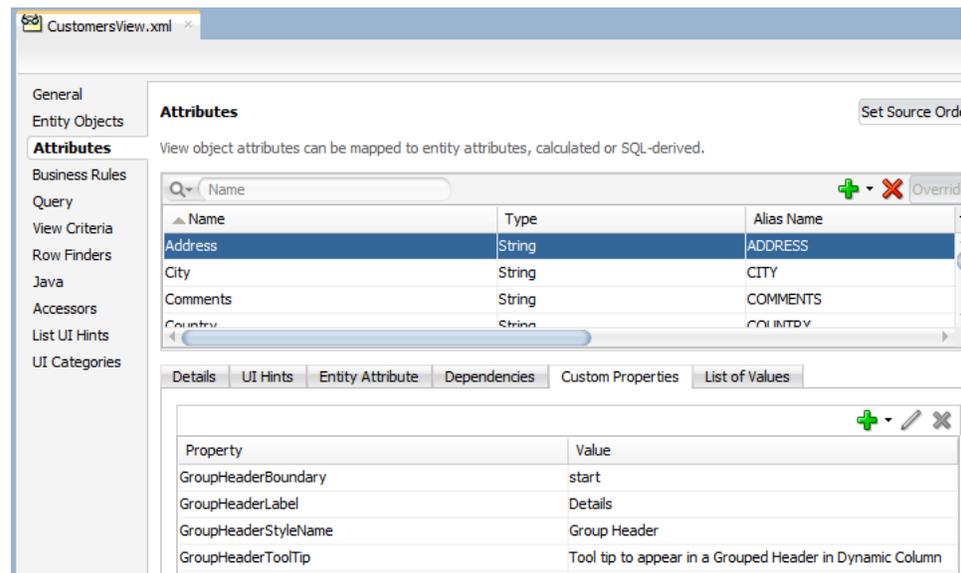
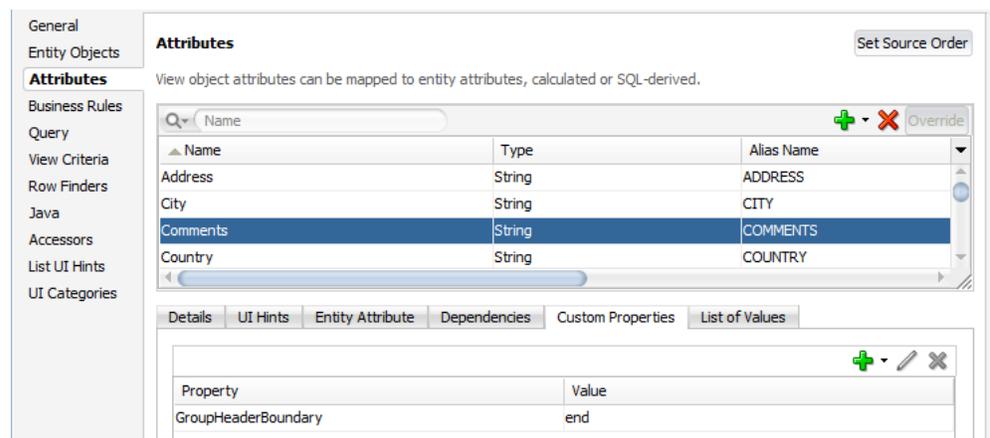


Figure 7-19 shows an attribute (Comment) that defines custom attribute property to end a grouped header.

Figure 7-19 Custom Attribute Property to End a Grouped Header in a Dynamic Column



At runtime, a dynamic column expands to render columns for the view object's attributes. For this example the expanded columns are Address, City, and Comments that are configured to render a grouped header. Figure 7-20 shows the `GroupHeader` properties that you configure in the dynamic column at design time. At runtime, the EL expressions retrieve and evaluate the values of the configured custom attribute properties shown in Figure 7-18 and Figure 7-19. A grouped header labeled **Details** forms for the Address, City, and Comments columns.

For information about how to configure the `GroupHeader` properties, see [How to Group Columns in an ADF Table Component](#).

Figure 7-20 Starting a Grouped Header in a Dynamic Column

Layout	
GroupHeader	<code>#{bindings.Customers.[oracle.summitdi.model.queries.CustomersView].hints.*.GroupHeaderBoundary}</code>
Boundary	<code>#{bindings.Customers.[oracle.summitdi.model.queries.CustomersView].hints.*.GroupHeaderBoundary}</code>
Label	<code>#{bindings.Customers.[oracle.summitdi.model.queries.CustomersView].hints.*.GroupHeaderLabel}</code>
StyleName	Group Header
Tooltip	<code>#{bindings.Customers.[oracle.summitdi.model.queries.CustomersView].hints.*.GroupHeaderTool Tip}</code>
Visible	True

What Happens at Runtime: How an ADF Table Component Groups Columns

An extra table header row renders above the ADF Table component's regular table header row at runtime if you configure values for the `GroupHeader` properties. The values you specify for the `Label`, `StyleName` and `Tooltip` properties of the start column in each group determines the label, style and tooltip of the group header. Any values that you configure for those `GroupHeader` properties of other columns in the column group are ignored.

The style specified by the `GroupHeader.StyleName` property in the column that you configure as the start column is applied on the extra table header cells of all columns in the group. The horizontal alignment of the group header label centers across the extra table header cells of all columns in the group.

The tooltip defined by the `GroupHeader.Tooltip` property in the start column renders on the extra table header cell of the end column in the group, as shown in [Figure 7-21](#).

Figure 7-21 Grouped Columns Rendering Styles and Displaying a Tooltip

Inventory Details					warehouse Details		
Product	Amount in Stock	Out of Stock Explanation	Max in Stock	Restock Date	Warehouse	Manager First Name	Manager Last Name
50169	2530		2600		101	Molly	Urguhart

Make sure that columns you configure as start and end columns render at runtime. If a column that you configure as a start column does not render at runtime, no column group forms. For example, if you configure Column 1 as a start column and Column 3 as an end column and Column 1 does not render at runtime because its `Visible` property returns false, no column group forms. Similarly, if Column 3 does not render, no column group forms that spans Column 1, Column 2, and Column 3. Instead, Column 1 renders as a single-column group.

Configuring an ADF Table Component to be Read-only

Describes how to configure an ADF Table component to be read-only.

The ADF Table component offers multiple features that are not available in the ADF Read-Only Table component, described in [Creating an ADF Read-Only Table Component](#). Examples of features available in the ADF Table component include dynamic columns, the ability to group columns together and the ability to resize columns. For this reason, you may want to create an ADF Table component and

configure it to be read-only rather than creating an ADF Read-only Table component. The `CustomerSearch-DT.xlsx` workbook in the Summit sample application contains an ADF Table component that is configured to be read-only.

How to Configure an ADF Table Component to be Read-only

You make an ADF Table component read-only by setting the `RowActions.UpdateRowEnabled` and `InsertRowEnabled` properties to `False` and deleting the `_ADF_ChangedColumn`, `_ADF_FlagColumn`, and `_ADF_StatusColumn` columns from the ADF Table component.

Before you begin:

It may be helpful to have an understanding of ADF Table component. See [Configuring an ADF Table Component to be Read-only](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality of Table-Type Components](#).

Insert an ADF Table component in your integrated Excel workbook, as described in [Inserting an ADF Table Component into an Excel Worksheet](#).

To configure an ADF Table component to be read-only:

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, set the `RowActions.UpdateRowEnabled` property of the ADF Table component to `False`.
4. In the Edit Component: ADF Table dialog, click the browse (...) icon of the Columns property.

The Edit Column dialog appears, listing all the columns of the ADF Table component.

5. Delete the `_ADF_ChangedColumn`, `_ADF_FlagColumn`, and `_ADF_StatusColumn` columns by selecting these columns and clicking **Remove** in the Edit Columns dialog.

Note:

Do not remove the **Key** column. For information about the **Key** column, see [Configuring ADF Table Component Key Column](#).

6. (Optional) Consider changing each column's `UpdateComponent` component type value from `ModelDrivenColumnComponent` to `OutputText`.

This is not necessary if the view object is configured to be read-only. This is because the `ReadOnly` property of each `UpdateComponent` is bound to the corresponding `readOnly` attribute hint by default if you create an ADF Table component from a tree binding. The `CustomerSearch-DT.xlsx` workbook demonstrates this implementation.

7. (Optional) Consider changing each column's `CellStyleName` property to `Read-only Cell` to visually distinguish read-only cells from editable cells.
See [Working with Styles](#).
8. Click **OK**.

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.

Note:

The ADF Table component offers multiple features that are not available in the ADF Read-Only Table component. For this reason, you may want to consider creating an ADF Table component and configure it to be read-only rather than creating an ADF Read-Only Table component. See [Configuring an ADF Table Component to be Read-only](#).

The ADF Read-only Table 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 [ADF Read-only Table Component Properties and Actions](#).

Note:

- 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.

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. See [Creating an ADF Read-Only Table Component](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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, as described in [Removing ADF Desktop Integration Components](#).

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.

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. See [Limiting the Number of Rows Your Table-Type Component Downloads](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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.

See [Using Action Sets](#).

3. Configure properties for the `RowLimit` group of properties, as described in [Table 7-14](#). For information about these properties, see [Frequently Used Properties in the ADF Desktop Integration](#).

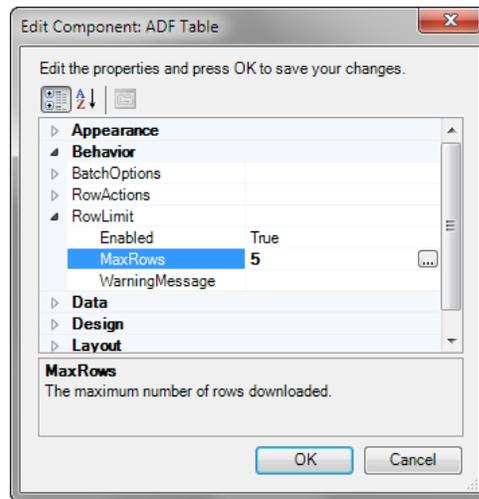
Table 7-14 RowLimit Group of Properties

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 Download 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 7-22](#) shows the Edit Component dialog in the `EditCustomers-DT.xlsx` workbook where the row limit of an ADF Table component is configured.

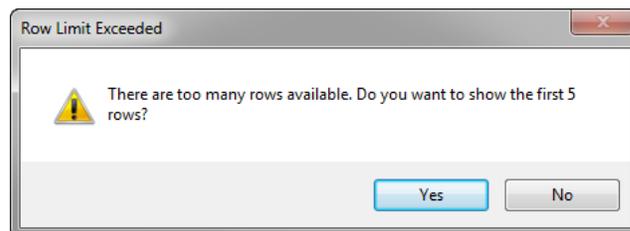
Figure 7-22 Limiting Number of Rows of an ADF Table Component



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 7-23](#) appears if you specify an EL expression for `RowLimit.MaxRows` or do not modify its default value.

Figure 7-23 Row Limit Exceeded Warning Message



Tracking Changes in an ADF Table Component

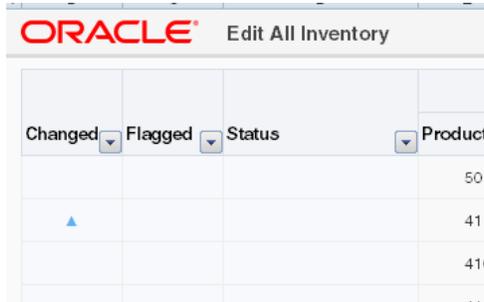
End users can create or modify data in the cells of an integrated Excel workbook that hosts an ADF Table component.

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 7-24](#) shows an example.

Figure 7-24 Changed Column in an ADF Table Component



Changed	Flagged	Status	Product
			501
▲			411
			410
			...

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`. Various subcomponents, such as the `ModelDrivenColumnComponent`, have a `ReadOnly` property. You can write an EL expression or a literal 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 information about `ReadOnly` EL expressions and change tracking, see [Evaluating EL Expressions for ReadOnly Properties](#).

Evaluating EL Expressions for ReadOnly Properties

If a table column's `ReadOnly` property 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`, `UploadAllOrNothing`), and change tracking

What Happens at Runtime: Evaluating EL Expression While Downloading Data

During `Download`, the EL expression is evaluated with the current binding value as expected.

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`.

 **Note:**

During change tracking, column component `Value` properties are not evaluated. So, for example, cell values will be blank for newly inserted rows regardless of the configured `Value` EL expression.

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 information about worksheet protection, see [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.

Displaying Error Messages in an Error Message Column

You can configure your ADF Table component to display a message column where row-level error messages appear at runtime.

At runtime, ADF Desktop Integration populates the cells in the error message column with the row-level error messages captured during invocation of the `Upload`, `UploadAllOrNothing`, and `DeleteFlagged` actions. End users of the integrated Excel workbook can use Excel's Filter, Sort, and Find capabilities to locate errors that display the same error message to efficiently resolve errors before resubmitting corrected rows.

Excel's Filter and Sort capabilities can be found by default in Excel's Home ribbon tab, in the Editing ribbon group. Filter and Sort can also be invoked by selecting a cell in the error message column and right-clicking to display the context menu where you choose **Filter** followed by the appropriate sub-menu option. When Excel worksheet protection is enabled, access to the Filter and Sort capabilities are disabled. This is expected behavior.

The cell in the error message column displays one error message row. If an action reported multiple errors, the end user needs to resolve the displayed error and replay the action before they can see the next error message in the cell. In this scenario, your end user should refer to the Status Viewer where all errors for a row appear simultaneously, in addition to details such as cause and actions to take to resolve the error.

For action sets that contain both `Upload` and `DeleteFlagged` actions, the error message column shows the message from the failed action. Where both `Upload` and `DeleteFlagged` actions produce row-level errors for the same row, the error message column shows the error from the second action in the action set.

How to Display Error Messages in an Error Message Column

You add a new column to your ADF Table component and set the ID of the new column to `_ADF_ErrorColumn`.

Before you begin:

It may be helpful to have an understanding of how you can display an error column in an ADF Table component. See [Displaying Error Messages in an Error Message Column](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality of Table-Type Components](#).

To display error messages in an error message 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 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. Set the value of the ID property to `_ADF_ErrorColumn`.
6. Leave the following properties using their default values:
 - `DynamicColumn`
 - `InsertComponent`
 - `InsertUsesUpdate`
 - `UpdateComponent`

- `TriggersRowRefresh`
7. Configure other properties for the new column, such as `HeaderLabel`, `HeaderStyleName`, and so on as necessary.

 **Tip:**

For best results, set the `TableComponents.SmartRowFailureReportingEnabled` ADF Desktop Integration compatibility property to `True`.

8. Click **OK**.

8

Working with Lists of Values

Describes how to create dropdown lists of values (including dependent lists of values) in integrated Excel workbooks, in tables within workbooks, and how to display Search and Select list picker dialogs from Fusion web applications that users can invoke from workbooks.

This chapter includes the following sections:

- [About List of Values in an Integrated Excel Workbook](#)
- [Creating a List of Values in an Excel Worksheet](#)
- [Creating a List of Values in an ADF Table Component Column](#)
- [Adding a Model-Driven List Picker to an ADF Table Component](#)
- [Creating Dependent Lists of Values in an Integrated Excel Workbook](#)

About List of Values in an Integrated Excel Workbook

Consider implementing list of values in your integrated Excel workbooks for scenarios where you want to offer end users the ability to choose from a range of values or you want to constrain the values that end users can enter in the integrated Excel workbook.

ADF Desktop Integration provides a number of ways to address these use cases. You can, for example, configure:

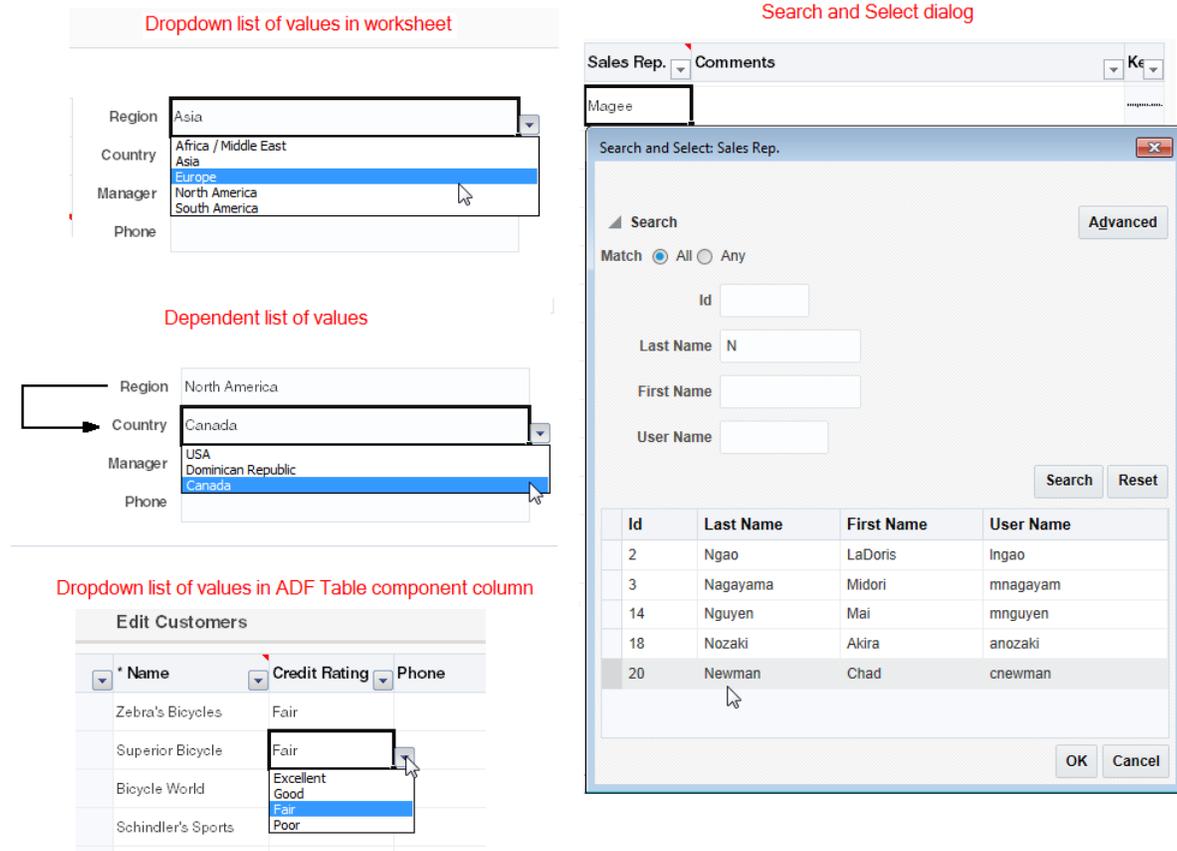
- A dropdown list of values in an Excel worksheet's cell
- A dependent list of values where the values displayed in one list (the child list of values) depends on the selected value in another list (the parent list of values)
- Configure the display of a Search and Select list picker dialog that provides advanced functionality for selecting values from lists

Consider configuring lists of values in the integrated Excel workbook in scenarios where the full set of values that end users can choose is relatively small (for example, numbers less than 30 values). The use of a Search and Select list picker dialog in a page from the Fusion web application may offer a better user experience when you have lists of values with more than 30 values.

Adding Lists of Values to Integrated Excel Workbooks Use Cases and Examples

Using the ADF List of Values component and other subcomponents from ADF Desktop Integration, you can create a variety of interfaces that present end users with data to view and select. [Figure 8-1](#) shows a number of examples from the Summit sample application workbooks that subsequent sections in this chapter discuss in more detail.

Figure 8-1 List of Values Implementations in Summit Sample Application Workbooks



Additional Functionality for Adding List of Values to an Integrated Excel Workbook

After you have added lists of values to your integrated Excel workbook, you may find that you need to add additional functionality to configure your workbook. The following sections describe other functionality that you can use:

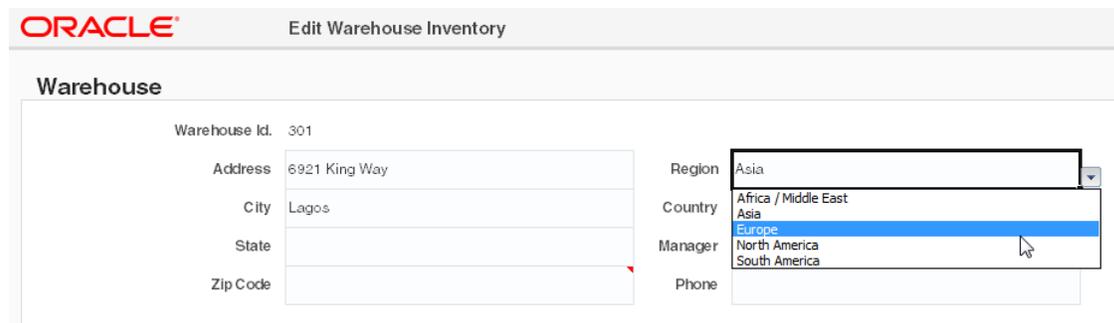
- **Styles:** You can configure the display of your form-type components using several predefined Excel styles. See [Working with Styles](#).
- **EL Expressions:** You can use EL expressions with form-type components. See [ADF Desktop Integration EL Expressions](#).
- **Tooltips:** You can configure tooltips to display additional information or instructional text to your end users. See [Displaying Tooltips in ADF Desktop Integration Components](#).
- **Action sets:** You can configure ordered lists of one or more actions to add interactivity to your integrated Excel workbook, as described in [Adding Interactivity to Your Integrated Excel Workbook](#).

Creating a List of Values in an Excel Worksheet

Use the ADF List of Values component when you want to create a dropdown list of values in an Excel worksheet cell at runtime.

The ADF List of Values component is intended for a short choice list, for example 20 or 30 items at most, but can display a maximum of 250 values at runtime. You can insert the ADF List of Values component into a cell in the Excel worksheet. [Figure 8-2](#) shows an implementation from the Summit sample application's `EditWarehouseInventory-DT.xlsx` where the user is constrained to picking one of the valid values for a list of regions.

Figure 8-2 Runtime List of Values in an Excel Worksheet



The screenshot shows the Oracle ADF application interface for 'Edit Warehouse Inventory'. The form is titled 'Warehouse' and contains several input fields. The 'Region' field is currently displaying 'Asia' and has a dropdown menu open, showing a list of regions: 'Africa / Middle East', 'Asia', 'Europe', 'North America', and 'South America'. The 'Europe' option is highlighted in blue. Other fields include 'Warehouse Id.' (301), 'Address' (6921 King Way), 'City' (Lagos), 'State', 'Zip Code', 'Country', 'Manager', and 'Phone'.

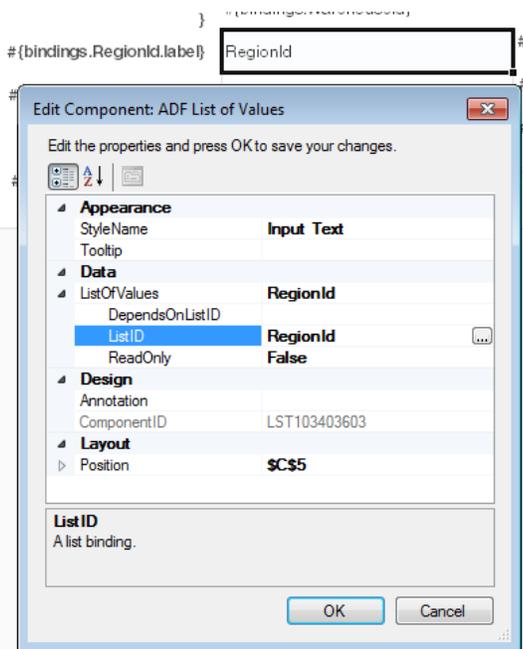
You must specify a value for the `ListID` property. The `ListID` property references the list binding which populates the dropdown menu with a list of values at runtime after you invoke the worksheet `DownSync` action.

[Figure 8-3](#) 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.

Note:

- You can display a dropdown menu in an ADF Table component's column. See [Creating a List of Values in an ADF Table Component Column](#).
- ADF List of Values components are intended for use with textual items; using date and numeric values is not supported.
- ADF List of Values does not support configurations where multiple display attributes are selected at design-time.

Figure 8-3 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 [ADF List of Values Component Properties](#).
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. 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, as described in [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.

Creating a List of Values in an ADF Table Component Column

Use the `ModelDrivenColumnComponent` subcomponent when you want to render a dropdown list of values in an ADF Table component column.

The list of values component is intended for a short choice list, for example 20 or 30 items at most, but can display a maximum of 250 values at runtime. Unlike other ADF Desktop Integration components, the `ModelDrivenColumnComponent` subcomponent does not appear in the components palette described in [Using the Components Palette](#). Instead, you select it as a subcomponent when you specify values for the `UpdateComponent` properties of an ADF Table component column. For information about the properties of an ADF Table component column, see [ADF Table Component Column Properties](#). For information about creating a model-driven list, see [Adding a ModelDrivenColumnComponent Subcomponent to Your ADF Table Component](#).

After you specify the `ModelDrivenColumnComponent` subcomponent, you must specify a tree binding attribute associated with a model-driven list as a value for the `ModelDrivenColumnComponent` subcomponent's `Value` property. The model-driven list of the tree binding attribute populates the dropdown menu in the ADF Table component's column with a list of values at runtime. For information about creating a model-driven list, see *How to Create a Model-Driven List of Developing Fusion Web Applications with Oracle Application Development Framework*.

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 `ModelDrivenColumnComponent` as the subcomponent. You then specify a tree binding attribute as the value for the `ModelDrivenColumnComponent` subcomponent's `Value` property. A model-driven list must be associated with the tree binding attribute that you specify.

 **Note:**

- The `ModelDrivenColumnComponent` subcomponent does not support a model-driven list whose control type is `combo_lov`.
- Tree attributes with a control type of `input_text_lov` will not render as ADF List of Values components. Instead, they expose model-driven list picker functionality, as described in [Adding a Model-Driven List Picker to an ADF Table Component](#).
- ADF List of Values components using date values are not supported.
- The `ModelDrivenColumnComponent` 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. See [Creating a List of Values in an ADF Table Component Column](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Adding List of Values to an Integrated Excel Workbook](#).

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 **UpdateComponent** to configure the runtime list of values for update and download operations.
 - (Optional) Click the browse (...) icon beside the input field for **InsertComponent** to configure the runtime list of values for insert operations. This is rare.

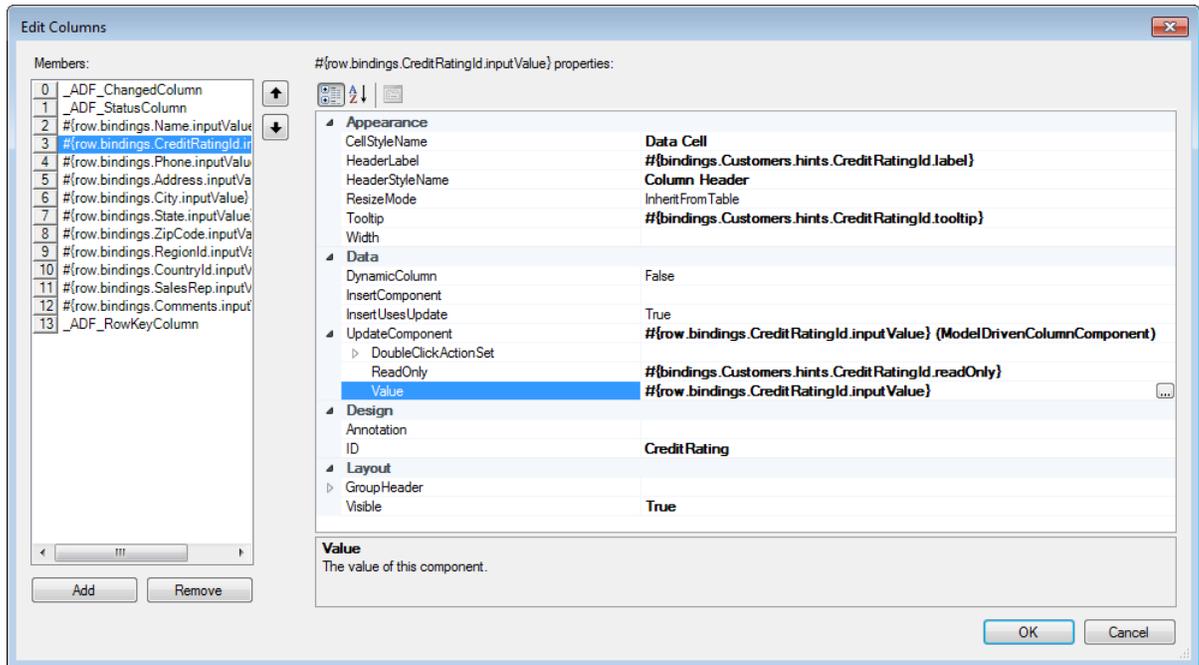
In both options, the Select subcomponent to create dialog appears.

6. Select **ModelDrivenColumnComponent** and click **OK**.
7. Expand the property that you selected in Step 5 and select a binding attribute associated with a model-driven list for the `Value` property.

Set the `ReadOnly` property to `False` if you do want users to edit the values in the column, set to `True` otherwise.

[Figure 8-4](#) shows the property inspector for the **Credit Rating** column that renders in the Summit sample application's `EditCustomers-DT.xlsx` workbook.

Figure 8-4 ADF Table Component Column Configured to Display a List of Values

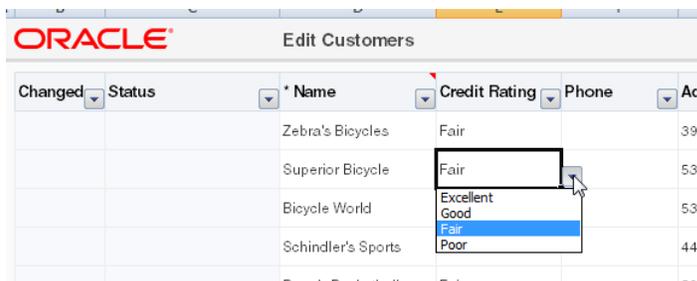


8. Click **OK**.

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 8-5 shows an example from `EditCustomers-DT.xlsx` workbook in the Summit sample application, where **Credit Rating** is the column configured to display a list of values.

Figure 8-5 Runtime View of an ADF Table Component Column Displaying a List of Values



Adding a Model-Driven List Picker to an ADF Table Component

You can configure an ADF Table component and use the existing model-layer metadata of your Fusion web application to provide a Search and Select list picker dialog in the integrated Excel workbook.

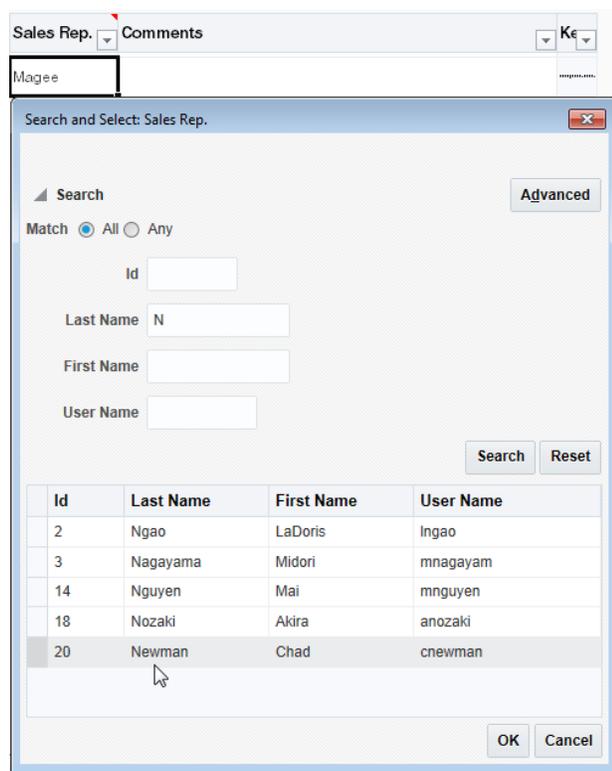
The Search and Select list picker dialog is similar to that seen when you click the search icon or button to open the Search and Select popup of the `af:inputListOfValues` component on an ADF Faces page. [Figure 8-6](#) shows an example from the `EditCustomers-DT.xlsx` workbook where an end user double-clicks the cell where they want to input a new data value. They search and select the new value in the popup that appears.



Tip:

Consider adding a column header tooltip that instructs users to double-click column cells in order to pick a value.

Figure 8-6 Model-Driven List Picker Invoked from Table Column Cell



To add a model-driven list picker to an ADF Table component:

1. Open your Fusion web application in JDeveloper.
2. Configure your view object in the same way as you would to use an `af:inputListOfValues` component.
 - a. Add a view accessor.

For information about creating a view accessor, see *How to Create a View Accessor for an Entity Object or View Object in Developing Fusion Web Applications with Oracle Application Development Framework*.
 - b. Create a List-of-Values (LOV) for the attribute.

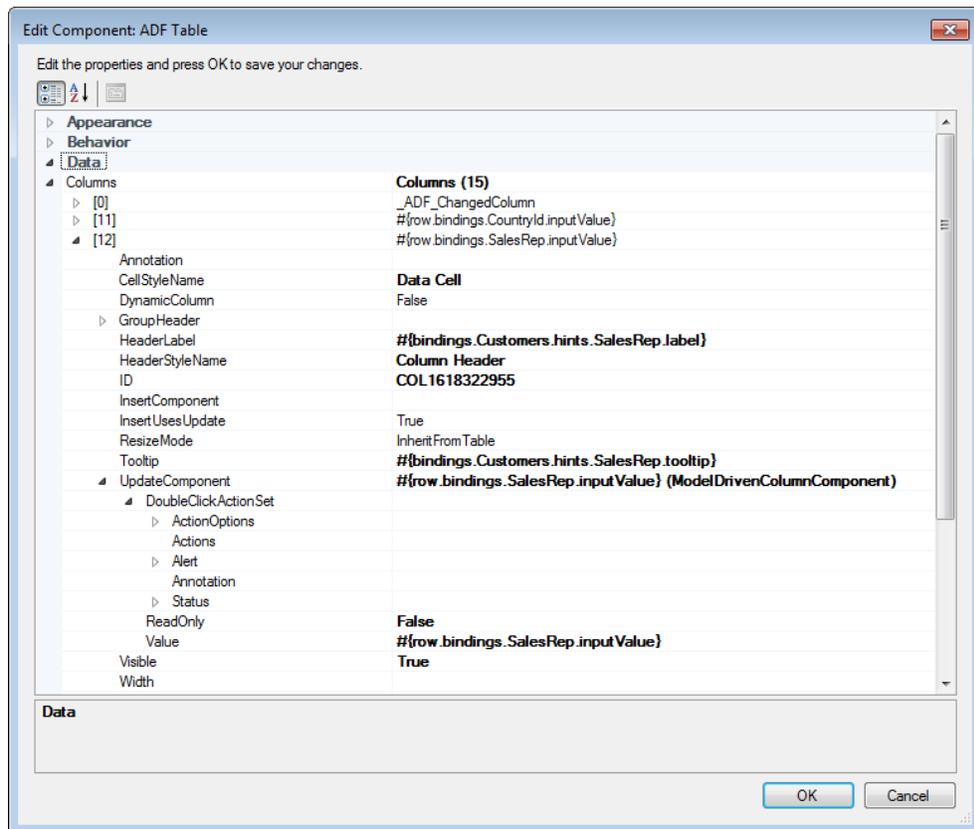
For information about creating a List of Values component, see *Creating List of Values (LOV) in Developing Fusion Web Applications with Oracle Application Development Framework*.
 - c. Set the UI Hints for the LOV. Ensure that **Default List Type** is set to `Input Text with List of Values`.

For information about setting UI Hints, see *How to Set User Interface Hints on a View Object LOV-Enabled Attribute in Developing Fusion Web Applications with Oracle Application Development Framework*.
 - d. Expose the view object as a tree binding in the page definition used by your worksheet.
3. Verify that your application's `web.xml` file configures the filter for ADF Library Web Application Support (`<filter-name>ADFLibraryFilter</filter-name>`).

For information, see [Configuring the ADF Library Filter for ADF Desktop Integration](#).
4. Open the integrated Excel workbook.
5. For any table columns bound to LOV-backed attributes, be sure to use the `ModelDrivenColumnComponent` component type in the column configuration. Note that for table columns configured in this way, the `DoubleClickActionSet` property of the `UpdateComponent` and `InsertComponent` will be ignored at runtime.

[Figure 8-7](#) shows the type of component of the `UpdateComponent` property set to `ModelDrivenColumnComponent`.

Figure 8-7 UpdateComponent Property of the ADF Table Component



6. If not set already, set the `Workbook.Compatibility.TableComponents.ModelDrivenColumns.InputListOfValuesPickerEnabled` property to `True`.

For information about the `InputListOfValuesPickerEnabled` property, see [ADF Desktop Integration Compatibility Properties](#).

7. (Optional) Configure `RowData.CachedAttributes` for the ADF Table component when a different attribute on the underlying iterator should be set by the action of the model-driven picker. For example, in the `EditCustomers-DT.xlsx` workbook, the **Sales Rep.** column exposes a `ModelDrivenColumnComponent` subcomponent, but also defines `SalesRepId` as a value for `RowData.CachedAttribute`.

After configuring the Fusion web application, integrated Excel workbook, and table columns, run the workbook and double-click the table columns that expose LOV-backed attributes to open the model-driven list picker dialog.

What You May Need to Know About Model-Driven List Pickers in ADF Table Components

By default, all columns in a table are configured to use the `ModelDrivenColumnComponent` subcomponent when you create an ADF Table component by double-clicking a tree binding in the bindings palette. Any tree attributes bound to model-driven lists with a control type of `input_text_lov` automatically

support the rendering of a Search and Select list picker dialog at runtime. That is, no special configuration is needed.

If the tree attributes are not bound to model-driven lists or if you need a custom picker user-interface, see [Displaying Web Pages from a Fusion Web Application](#).

 **Note:**

End users may initially see an empty model-driven list picker dialog if the ADF Table component's ModelDrivenColumnComponent subcomponent exposes a view object attribute that has multiple list of values (LOV) defined on it through an LOV switcher. If an empty model-driven list picker dialog appears, end users should cancel and re-invoke the dialog to render the expected content in the dialog. See *Creating List of Values (LOV) in Developing Fusion Web Applications with Oracle Application Development Framework*.

Ensure that a row exists in the tree node, especially on child tree nodes, for any attribute configured to show a model-driven list picker dialog.

Creating Dependent Lists of Values in an Integrated Excel Workbook

You can create a dependent list of values in an integrated Excel workbook using the list of values components that ADF Desktop Integration provides and by configuring the server-side list bindings appropriately.

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.
- **ModelDrivenColumnComponent subcomponent**
You configure properties for this subcomponent 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 8-8](#) shows an example with two illustrations from the `EditWarehouseInventory-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 8-8 List of Values and Dependent List of Values

The figure consists of two screenshots of the 'Edit Warehouse Inventory' form. Both screenshots show the 'Warehouse' section with fields for Warehouse Id. (301), Address (6921 King Way), City (Toronto/Tokyo), State, and Zip Code. To the right, there are fields for Region, Country, Manager, and Phone. In the top screenshot, Region is 'North America' and Country is 'Canada'. A dropdown menu for Country is open, showing 'USA', 'Dominican Republic', and 'Canada'. In the bottom screenshot, Region is 'Asia' and Country is 'Japan'. A dropdown menu for Country is open, showing 'Japan', 'India', and 'Hong Kong'.

Table 8-1 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 8-1 require model-driven lists. For information about creating a model-driven list, see *How to Create a Model-Driven List in Developing Fusion Web Applications with Oracle Application Development Framework*.

Table 8-1 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>See How to Create Dependent Lists of Values in Excel Worksheets.</p>

Table 8-1 (Cont.) Dependent List of Values Configuration Options

Configuration	Requirements
Render both the parent and child list of values in ADF Table component columns using <code>ModelDrivenColumnComponent</code> subcomponents.	<p>Both the parent and child list of values (<code>ModelDrivenColumnComponent</code> subcomponents) must reference tree binding attributes associated with model-driven lists.</p> <p>Both columns (parent and child list of values) must use the same value for the <code>InsertUsesUpdate</code> column property.</p> <p>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:</p> <ul style="list-style-type: none"> • 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 <p>See How to Create Dependent Lists of Values in ADF Table Component Columns.</p>

Note the following points if you plan to create a dependent list of values:

- If the selection in the parent list of values changes, the child list of values is reset without warning the 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 [Caching Lists of Values for Use Across Multiple Web Sessions](#).
- 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.

How to Create Dependent Lists of Values in Excel Worksheets

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 information about ADF List of Values, see [ADF List of Values Component Properties](#).

Before you begin:

It may be helpful to have an understanding of dependent list of values. See [Creating Dependent Lists of Values in an Integrated Excel Workbook](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Adding List of Values 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 information about adding bindings to page definition files, see [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 [Creating a List of Values in an Excel Worksheet](#).
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`

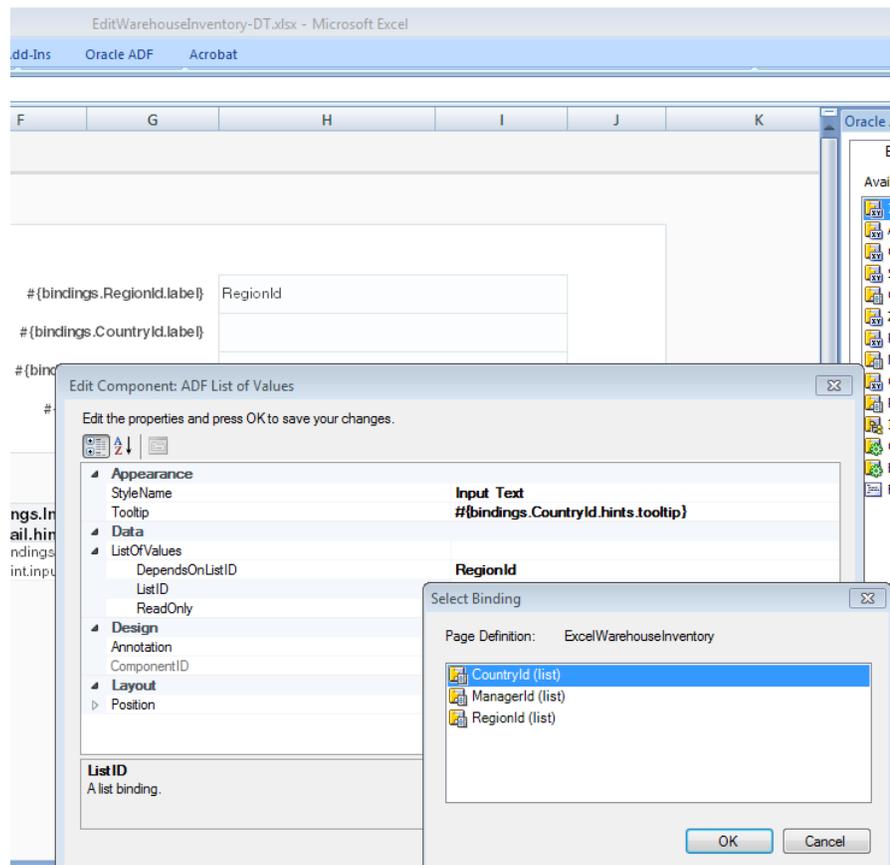
Specify the list binding that is the child in the dependent list of values.

- `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 8-9](#) shows the property inspector for the child ADF List of Values where the `RegionId` list binding is specified as the parent list of values (`DependsOnListID` property) and `CountryId` list is the dependent list of values (`ListID` property).

Figure 8-9 Design Time Dependent List of Values in an Excel Worksheet



6. Click **OK**.

What Happens at Runtime: How an 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 8-10 shows an example where **Country**, a dependent list value, displays only the states from the selected **Region** list value.

Figure 8-10 Runtime Dependent List of Values in an Excel Worksheet

The screenshot shows an Oracle ADF application window titled 'Edit Warehouse Inventory'. The 'Warehouse' section contains several input fields and dropdown menus:

- Warehouse Id.**: 301
- Address**: 6921 King Way
- City**: Toronto
- State**: (empty)
- Zip Code**: (empty)
- Region**: North America
- Country**: A dropdown menu with 'Canada' selected.
- Manager**: A dropdown menu with 'USA', 'Dominican Republic', and 'Canada' (highlighted) as options.
- Phone**: (empty)

How to Create Dependent Lists of Values in ADF Table Component Columns

Use instances of the `ModelDrivenColumnComponent` 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 for the parent `ModelDrivenColumnComponent` subcomponent's `value` property. Also specify a tree binding attribute for the child `ModelDrivenColumnComponent` subcomponent's `value` property. Ensure that both tree binding attributes are associated with model-driven lists before you add the tree binding to your page definition file. Ensure also that the dependency between the parent and child model-driven lists is configured on the server.

The **Region** and **Country** columns in the Summit sample application's `EditCustomers-DT.xlsx` workbook demonstrate an implementation of a dependent list of values in an ADF Table component.

The following links provide information about:

- Creating a model-driven list, see [How to Create a Model-Driven List in *Developing Fusion Web Applications with Oracle Application Development Framework*](#).
- Defining a dependent model-driven list, see [How to Define Cascading Lists for LOV-Enabled View Object Attributes in *Developing Fusion Web Applications with Oracle Application Development Framework*](#).
- Adding a tree binding to your page definition file, see [Working with Page Definition Files for an Integrated Excel Workbook](#).
- For information about the `ModelDrivenColumnComponent` subcomponent, see [ModelDrivenColumnComponent Subcomponent Properties](#).

Before you begin:

It may be helpful to have an understanding of dependent list of values. See [Creating Dependent Lists of Values in an Integrated Excel Workbook](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Adding List of Values 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, as described in [Inserting an ADF Table Component into an 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 `ModelDrivenColumnComponent` as the column's subcomponent type. For more information about creating a list of values, see [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 `ModelDrivenColumnComponent` as the column's subcomponent type and specify the same value for the `InsertUsesUpdate` column property as the column in the parent list of values. Both columns (parent and child list of values) must use the same value for the `InsertUsesUpdate` column property. For more information about creating a list of values, see [Creating a List of Values in an ADF Table Component Column](#).
6. Click **OK**.

What Happens at Runtime: ADF Table Component Column Renders a Dependent List of Values

At runtime, the ADF Table component renders both instances of the `ModelDrivenColumnComponent` 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 8-11](#) shows an example from the Summit sample application's `EditCustomers-DT.xlsx` workbook where the value that the end user selects in the **Region** column list of values (Europe) results in the corresponding values for sub-category appearing in the **Country** column list of values (France, Germany, and so on).

Figure 8-11 Runtime Dependent List of Values in an ADF Table Component's Columns

Changed	Status	* Name	Credit Rating	Address	City	State	Zip Code	Region	Country	Sal
		Zebra's Bicycles	Fair	3910 Colley Ave	Norfolk	VA	23508	North America	USA	Mac
▲		Superior Bicycle	Fair	538 Superior Ave E	Cleveland	OH	44114	Europe	France	Mac
		Bicycle World	Fair	5300 SW 17th St	Topeka	KS	66604	North America	France Germany Czech Republic Russian Federation	Mac
		Schindler's Sports	Fair	4479 Forest Park Ave	St Louis	MO	63108	North America		Mac
		Barry's Basketball	Fair	56 E Superior St	Chicago	IL	60611	North America	USA	Mac
		Gavin Sporting Goods	Fair	1935 SE Hawthorne Blvd	Portland	OR	97214	North America	USA	Mac

 **Note:**

When the end user changes the parent list selection, the child list items are changed for the current row only.

9

Adding Interactivity to Your Integrated Excel Workbook

Add interactivity to your integrated Excel workbooks by configuring action sets that invoke actions such as Upload and Download, exposing commands on the ribbon tab, and using EL expressions in Excel formulas.

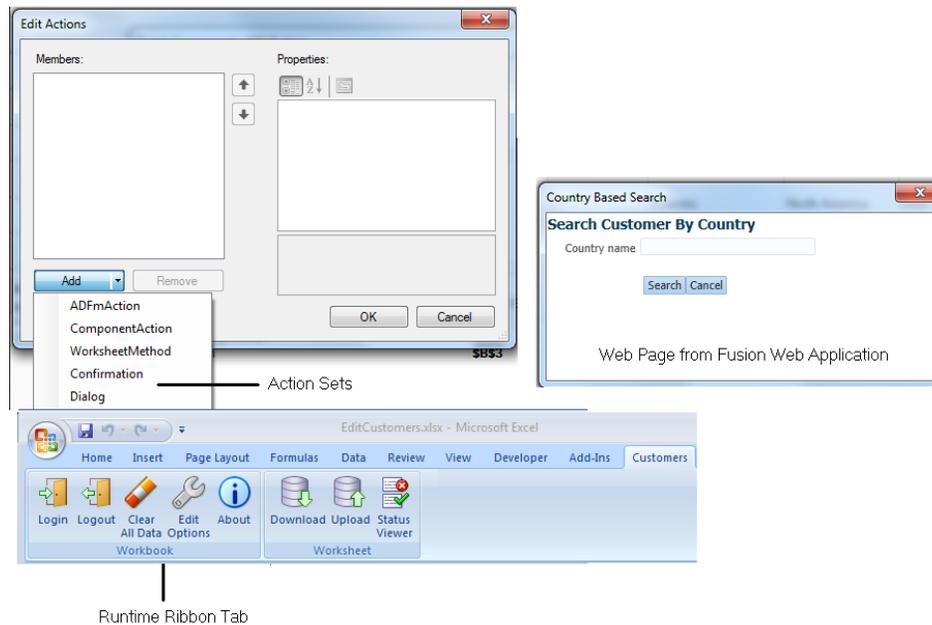
This chapter includes the following sections:

- [About Adding Interactivity to an Integrated Excel Workbook](#)
- [Using Action Sets](#)
- [Configuring the Runtime Ribbon Tab](#)
- [Displaying Web Pages from a Fusion Web Application](#)
- [Using Row-Level Action Sets in a Table Column](#)
- [Using EL Expression to Generate an Excel Formula](#)
- [Using Calculated Cells in an Integrated Excel Workbook](#)
- [Configuring Automatic Row Refresh in an ADF Table Component](#)
- [Using Macros in an Integrated Excel Workbook](#)

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 9-1](#) shows some of the interactive features.

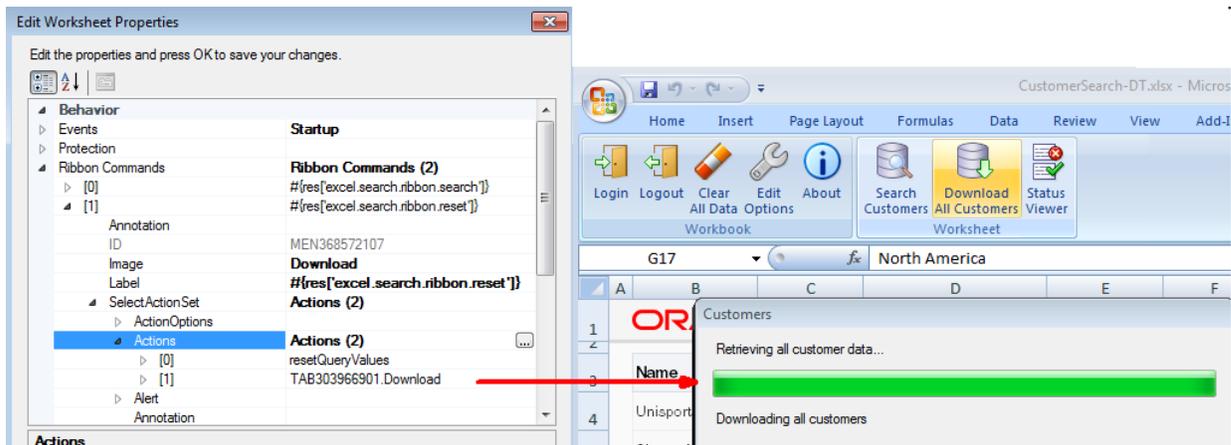
Figure 9-1 Interactivity Features in an Integrated Excel Workbook

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 commands that end users use at runtime to invoke action sets.

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 9-2](#), the **Download All Customers** ribbon command in `CustomerSearch-DT.xlsx` uses an action set with two actions to reset the query values associated with the worksheet. [Figure 9-2](#) also shows a ribbon command (**Search Customers**) where end users can invoke search functionality.

Figure 9-2 Action Sets of Download All Customers Ribbon Command



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. The following sections describe other functionality that you can use:

- **Display Web Pages:** You can display pages from the Fusion web application with which you integrate your Excel workbook. See [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. See [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. See [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. See [Using Calculated Cells in an Integrated Excel Workbook](#) and [Using Macros in an Integrated Excel Workbook](#).

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

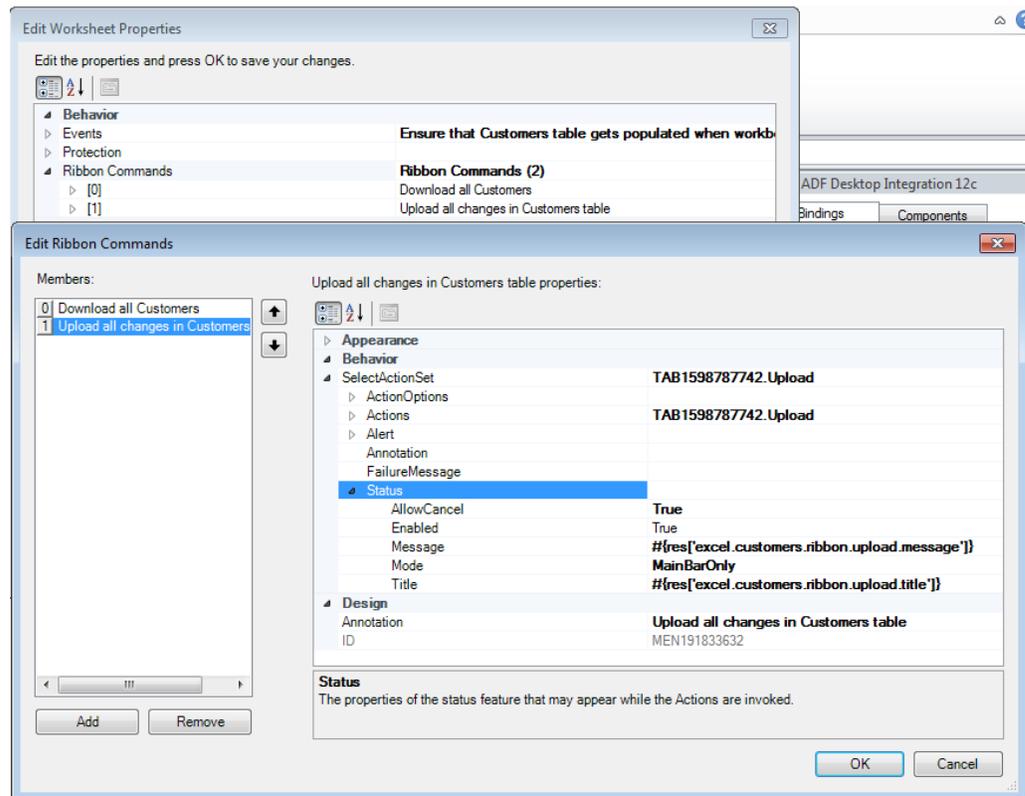
An action set can be invoked by an end-user's gesture (for example, clicking a ribbon command) or an Excel worksheet event. Where an end-user gesture invokes an action set, the name of the action set property in the ADF component's property inspector is prefaced by the name of the gesture required. The following list describes the property names that ADF Desktop Integration displays in property inspectors, and what user gesture can invoke an action set:

- `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 command, as the end user selects a ribbon command 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 command, 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 progress bar to end users while an action set runs the actions you define. For information about opening the Edit Action dialog, see [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 9-3](#) which shows the Edit Action dialog in design mode.

By default, an end user cannot open another integrated Excel workbook while an action set runs. If you know an action set will be long running, make it non-blocking, so your end users can do other work while they wait for the long running action set to complete.

Figure 9-3 Action Set for Upload Ribbon Command in 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 can help you manage multiple items more effectively.

Note:

ADF Desktop Integration invokes the actions in an action set in the order that you specify in the **Members** list view.

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 information about page definition files and action bindings in an integrated Excel workbook, see [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. See [Using Action Sets](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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:

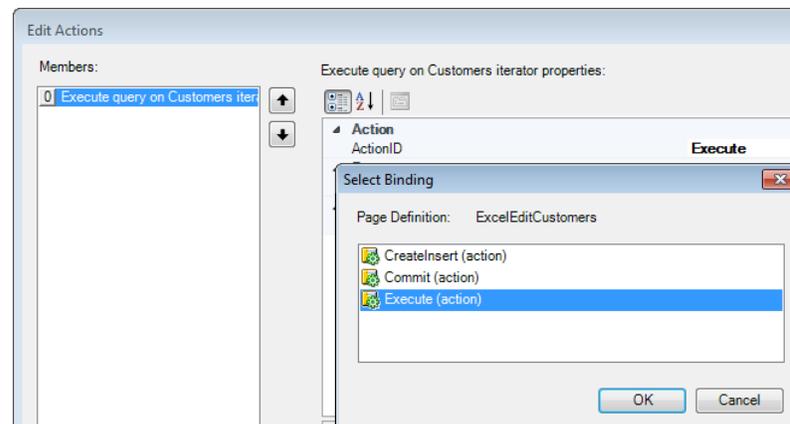
- ActionID

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. [Figure 9-4](#), for example, shows the `Execute` action binding that is the first action the `Download` action set in the Summit sample application's `EditCustomers-DT.xlsx` workbook 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.

Figure 9-4 Execute Action Binding



4. Click **OK**.

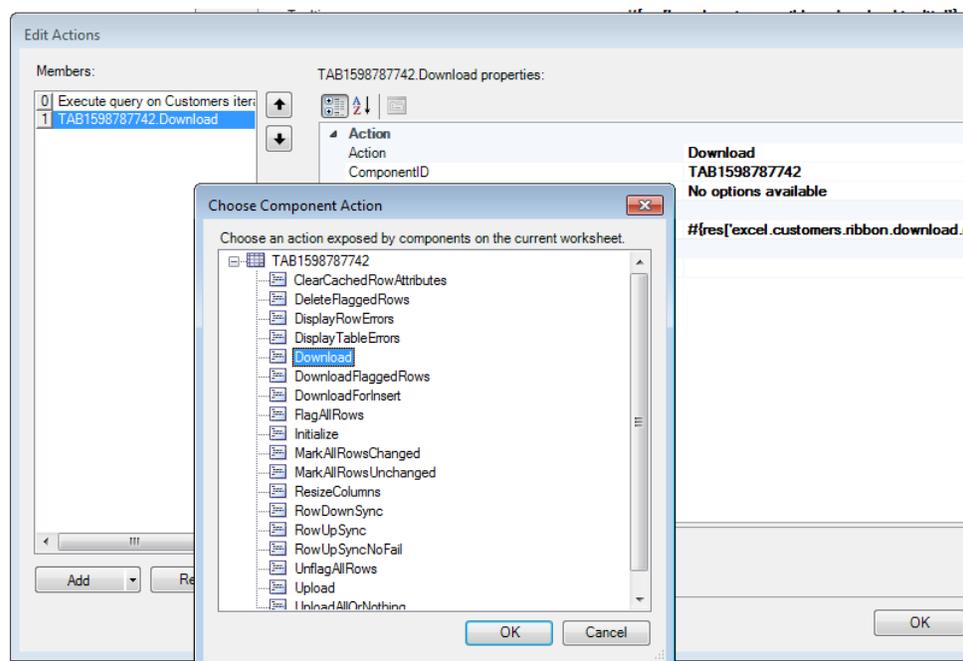
How to Invoke Component Actions in an Action Set

Some components, such as the ADF Table component, expose actions that can be used to manage the transfer of data between Excel worksheets that you integrate with a Fusion web application. More information about the actions available for ADF

Desktop Integration components can be found in [ADF Desktop Integration Component Properties and Actions](#).

You configure action sets to invoke one or more component actions by adding component actions to the array of actions in the action set. For example, [Figure 9-5](#) shows the Choose Component Action dialog where the `Download` action exposed by the ADF Table component present in the Summit sample application's `EditCustomers-DT.xlsx` workbook can be selected for invocation by that workbook's **Download** ribbon command's `SelectActionSet` action set.

Figure 9-5 Choose Component Method Dialog



 **Note:**

The Choose Component Action dialog appears empty if the current worksheet does not include any components that expose component actions.

Before you begin:

It may be helpful to have an understanding of action sets. See [Using Action Sets](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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 `Action` 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 [How to Display a Progress Bar while an Action Set Executes](#)).
4. Click **OK**.

What You May Need to Know About an Action Set Invoking a Component Action

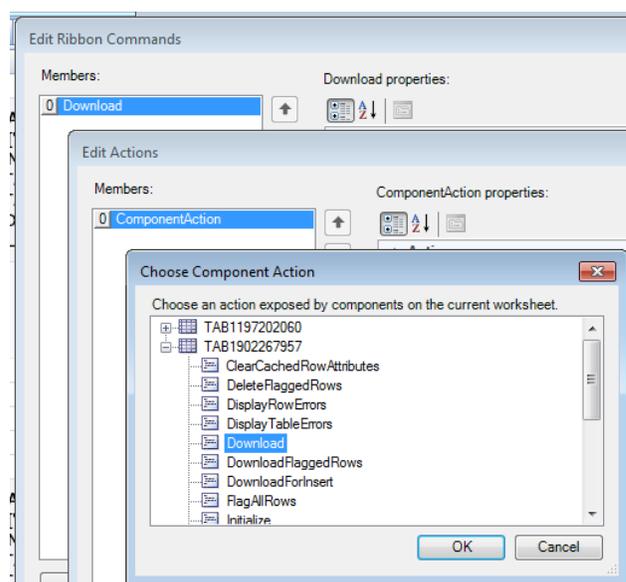
Note the following pieces of information about the behavior of action sets in integrated Excel workbooks.

Invoking Action Sets Before Logging In

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 [Authenticating the Excel Workbook User](#).

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 a component. [Figure 9-6](#) shows the Choose Component Action dialog displaying two instances of the ADF 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 9-6 Choose Component Action Dialog

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 information about the `Events` worksheet property and the worksheet events that can invoke an action set, see [Table A-20](#). See [Table A-15](#) for 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. See [Using Action Sets](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Adding Interactivity to an Integrated Excel Workbook](#).

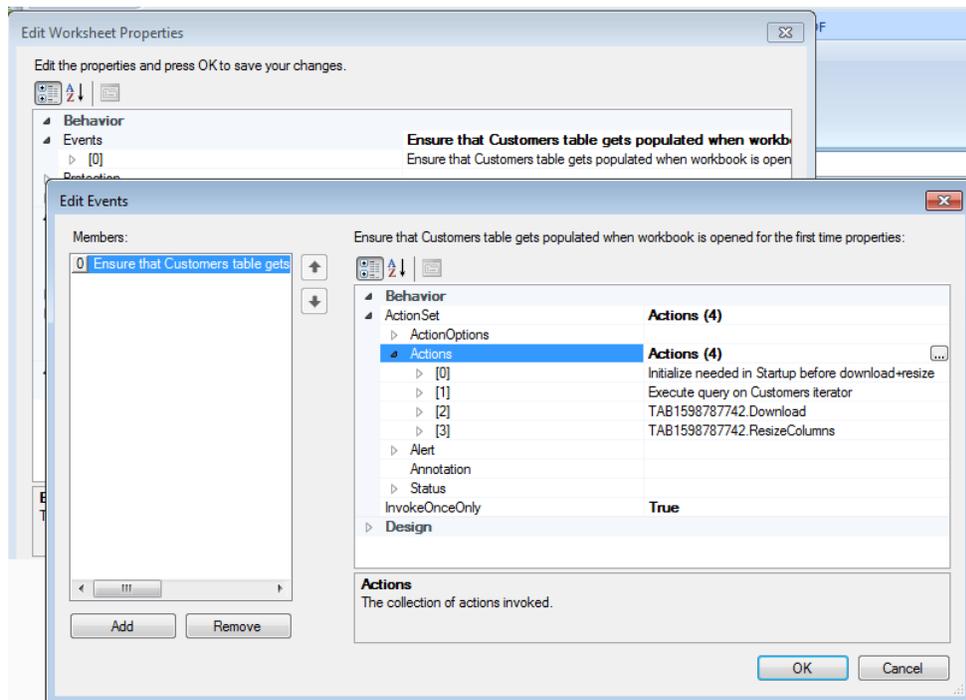
To invoke an action set from a worksheet event:

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 9-7 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 9-7 Worksheet Startup Event Invokes an Action Set



5. Click **OK**.

How to Display a Progress Bar while an Action Set Executes

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 a 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 property inspector for the action set where you want to configure the `Status` properties in the action set. Use, for example, the Edit Ribbons Command dialog if you want to configure `Status` properties in the `SelectActionSet` that a ribbon command invokes at runtime.

Before you begin:

It may be helpful to have an understanding of action sets. See [Using Action Sets](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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 Actions dialog of, for example, the ribbon command that invokes the action set.
3. Set values for the properties in the `Status` group of properties as described in [Table 9-1](#).

Table 9-1 Status Group of Properties

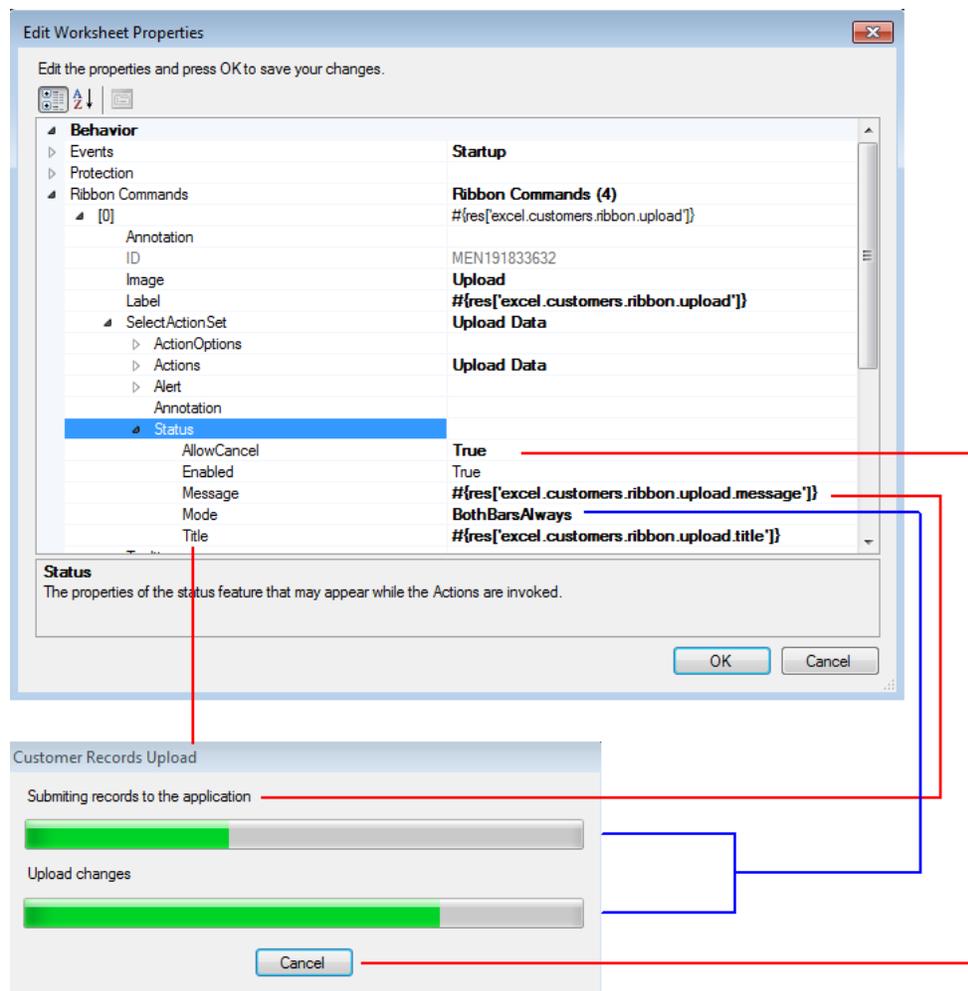
For this property...	Enter or select this value...
AllowCancel	True to display the Cancel button in the status dialog box. It indicates whether the action set execution can be canceled by the end user. For information about the Cancel button, see What You May Need to Know About Canceling an Action .
Enabled	True to display a status message. True is the default value.
Message	An optional EL expression or literal value that resolves to the status message to display at runtime. For example, the Upload ribbon command in the <code>EditCustomers-DT.xlsx</code> file has the following EL expression configured for the Message property: <code>{res['excel.customers.ribbon.upload.message']}</code>
Mode	Choose the visual appearance of progress bars. <ul style="list-style-type: none"> • Automatic: ADF Desktop Integration analyzes the action set to determine which progress bars to display. • BothBarsAlways: Shows both main and detail progress bars. • MainBarOnly: Shows one progress bar only. The bar displays progress through the list of actions. • DetailBarOnly: Shows one progress bar only. The bar displays progress of the current action. • MainMessageOnly: None of the progress bars are shown.
Title	An optional EL expression or literal value that resolves to the title of the status message to display at runtime. For example, the Upload ribbon command in the <code>EditCustomers-DT.xlsx</code> file has the following EL expression configured for the Title property: <code>{res['excel.customers.ribbon.upload.title']}</code>

 **Note:**

ADF Desktop Integration renders generic text at runtime if you do not specify values for the `Message` and `Title` properties described in [Table 9-1](#). For this reason, we recommend that you provide values for these properties that are specific to the functional context of your action set.

[Figure 9-8](#) shows the property values, along with their corresponding visual elements, configured for the `Status` group of properties of an ADF Table component's `Upload` action.

Figure 9-8 Status Message Properties in an Action Set



For information about the `Status` group of properties, see the entry for `Status` in [Table A-15](#).

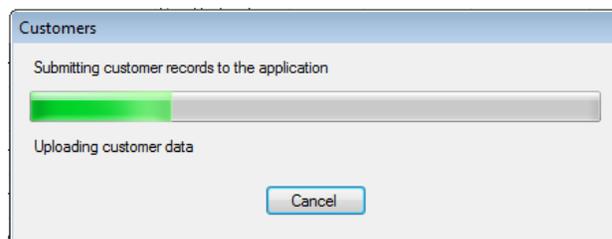
You can also use the optional `DetailStatusMessage` property to provide additional information to the user. For information about the `DetailStatusMessage` property, see [How to Invoke Component Actions in an Action Set](#).

4. Click **OK**.

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 9-9](#) 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 9-9 Runtime View of Status Message



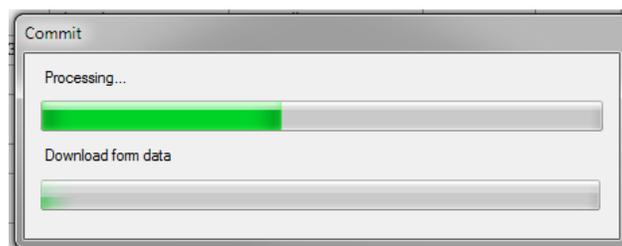
At runtime, if the value of the `Message` property is empty, ADF Desktop Integration provides a default localized value. If the `Title` property is empty, the label from the action set container (such as a ribbon command) is used. If the label of the container is also empty, then the default value provided by ADF Desktop Integration is used.

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 9-10](#) 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 9-10 Progress Bar for ADFmAction Type



- In the `Automatic` mode, if the action set has fewer 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 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, `Worksheet.DownSync`) or action sets that only display a dialog, the best practice is to disable the status message.

How to Allow End Users to Continue Working in Excel While an ActionSet Executes

You can configure your integrated Excel workbook so a long running action set does not prevent end users from using other integrated Excel workbooks or worksheets.

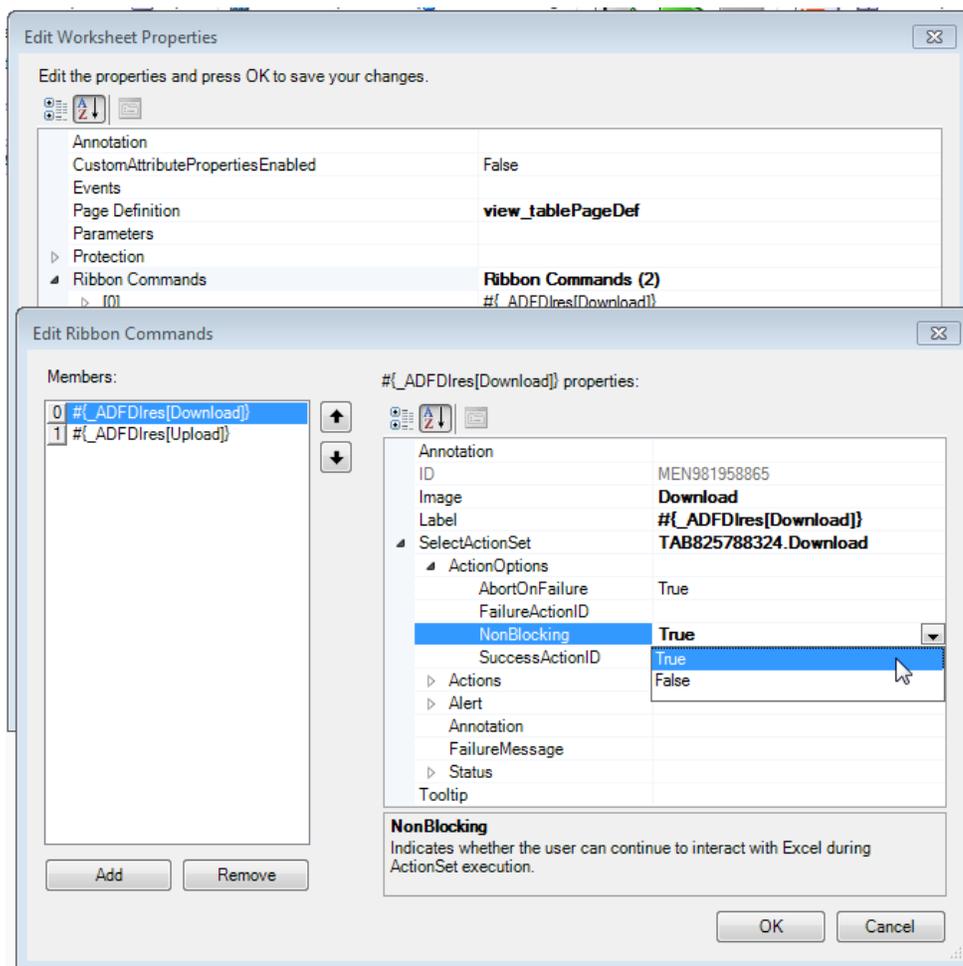
Integrated Excel workbooks that execute long running action sets (for example, an action set that includes a `Table.Download` action for 100,000 rows) block end users from using Microsoft Excel to access other integrated Excel workbooks, worksheets and non-integrated Excel workbooks and worksheets. You can configure action sets that you know contain long running actions so that end users can continue to use Excel to access other workbooks and worksheets while they wait for the action set to complete. To do this, you set the `ActionOptions.NonBlocking` property for the `ActionSet` to `True`. The default value is `False`.

If you set the `ActionOptions.NonBlocking` property to `True`, also consider displaying a progress dialog with a `Cancel` button to allow end user to cancel the action set. See [How to Display a Progress Bar while an Action Set Executes](#).

To allow end users to continue using Microsoft Excel while an `ActionSet` executes:

1. Open the integrated Excel workbook.
2. Open the property inspector of the component that, at runtime, invokes the long running `ActionSet` and navigate to the `ActionOptions` group of properties.
3. Set the `NonBlocking` property to `True`.

For example, [Figure 9-11](#) shows the property inspector where the `NonBlocking` property for a `Table.Download` action that a ribbon command invokes has been set to `True`.

Figure 9-11 NonBlocking Property to Allow an End User Use Excel While an ActionSet Executes

4. Click **OK**.

What Happens at Runtime: How End Users Continue Working While an ActionSet Executes

At runtime, ADF Desktop Integration starts a background operation when an integrated Excel worksheet invokes an `ActionSet` for which you have set the `NonBlocking` property to `True`.

While the background operation processes the non-blocking `ActionSet`, an end user can perform other operations, such as

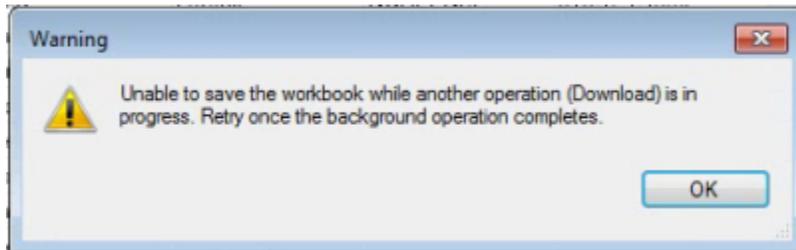
- Switch workbooks or worksheets
- Edit cells in other worksheets, including integrated Excel worksheets

The end user cannot edit the integrated Excel worksheet that contains the non-blocking `ActionSet` until the background operation completes.

If an end user performs another operation that requires communication with the ADF Desktop Integration-enabled Fusion web application, ADF Desktop Integrations sends

a notification that a background operation is in progress. For example, the following notification appears to an end user who attempts to save a worksheet while a background operation is in progress.

Figure 9-12 Notification Message that an Operation Cannot Proceed Until Background Operation Completes



Once the background operation completes, the end user can once again edit the integrated Excel worksheet that invoked the background operation. If the end user's currently active worksheet is different than the non-blocking `ActionSet` worksheet when the background operation completes, he or she receives a notification message that the background operation has completed.

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`
- `ADFmAction`
- Table actions: `RowUpSync`, `RowUpSyncNoFail`, `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 message content for the worksheet in the Status Viewer (if open) does not change. See [Using the Status Viewer to Report Error Messages to End Users](#).

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. The Status Viewer reflects the status of the rows processed before cancelation.

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`

The appearance of a Cancel button that allows end users cancel an action set requires you to set the `AllowCancel` property set to `True`, as described in [How to Display a Progress Bar while an Action Set Executes](#). 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.

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. Consider using an alert message for action sets that execute very quickly but have no interactive actions. In these cases, you may want to disable the `ActionSet.Status` group of properties and enable the `ActionSet.Alert` properties.

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 information about displaying web pages, see [Displaying Web Pages from a Fusion Web Application](#).

Before you begin:

It may be helpful to have an understanding of action sets. See [Using Action Sets](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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. In the Workbook group of the Oracle ADF tab, click **Worksheet Properties**.
3. In the Edit Worksheet Properties dialog, expand the **Ribbon Commands** node and select the ribbon command that contains the `SelectActionSet` for which you want to display an alert.
4. Expand the **Alert** group of properties for the action set and set values as described in [Table 9-2](#).

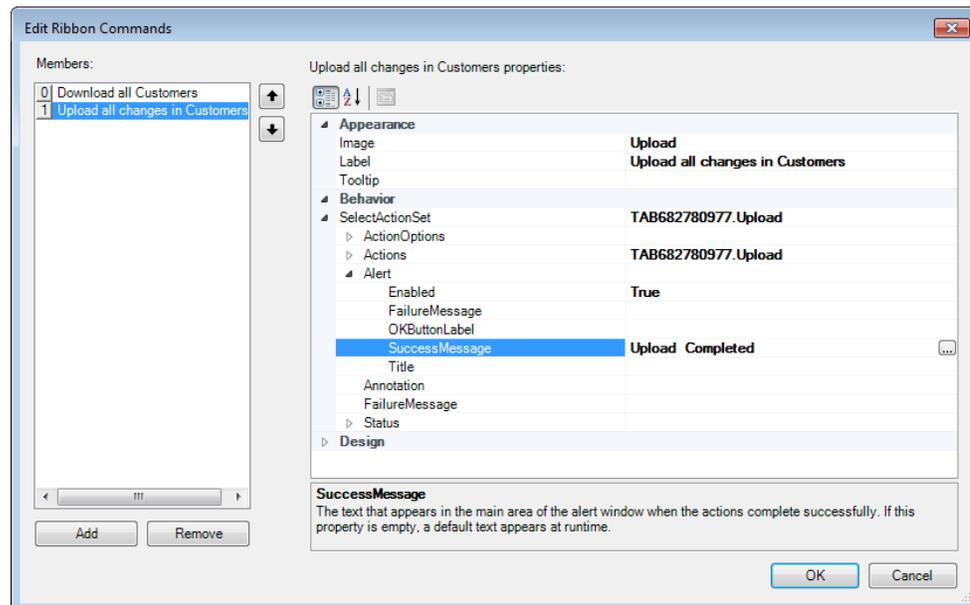
For example, [Figure 9-13](#) shows values configured for the `SuccessMessage` property in the `Alert` group of properties.

Table 9-2 ActionSet.Alert Group of Properties

For this property...	Enter or select this value...
Enabled	Select <code>True</code> from the dropdown list to display an alert message once the action set completes. The default value is <code>False</code> .

Table 9-2 (Cont.) ActionSet.Alert Group of Properties

For this property...	Enter or select this value...
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 information about error handling, see Using the Status Viewer to Report Error Messages to End Users .
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.

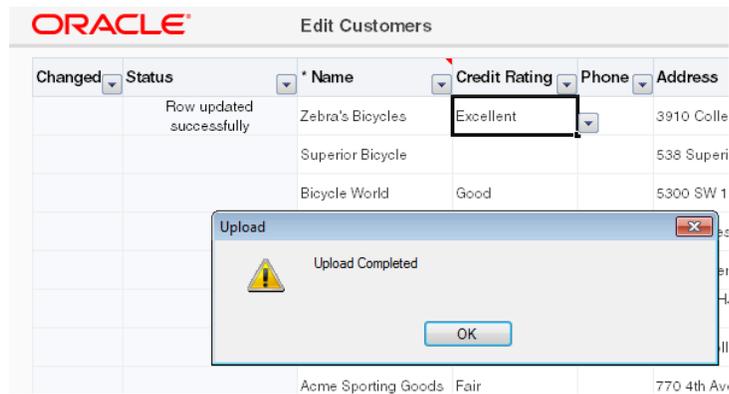
Figure 9-13 Alert Message Properties in an Action Set

5. Click **OK**.

What Happens at Runtime: How the Action Set Provides an Alert

Figure 9-14 shows an alert message configured for the `SuccessMessage` property in the `Alert` group of properties that appears at runtime when the action set successfully completes execution.

Figure 9-14 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.

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 [Getting Started with the Development Tools](#). More information about action set properties can be found in [Action Set Properties](#).

Before you begin:

It may be helpful to have an understanding of action sets. See [Using Action Sets](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Adding Interactivity to an Integrated Excel Workbook](#).

Integrated Excel workbooks report status information and errors that occur at runtime to the end user in the Status Viewer. See [Using the Status Viewer to Report Error Messages to End Users](#).

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 9-3](#).

Table 9-3 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 False, the action set runs all actions regardless of the success or failure of previous actions.
FailureActionID	(Optional) 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 FailureActionID methods is not supported. The Rollback method also should not be specified as the FailureActionID in an action set.
SuccessActionID	(Optional) 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 SuccessActionID methods is not supported.

- Optionally, write an EL expression for the action set's FailureMessage property that evaluates to a message to appear to the end user at runtime if the action set fails. This message appears in the worksheet area of the Status Viewer described in [Using the Status Viewer to Report Error Messages to End Users](#).
- Click **OK**.

How to Prompt the User for Confirmation 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 set proceeds 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. See [Using Action Sets](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Adding Interactivity to an Integrated Excel Workbook](#).

To invoke a Confirmation action from a component

- 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.



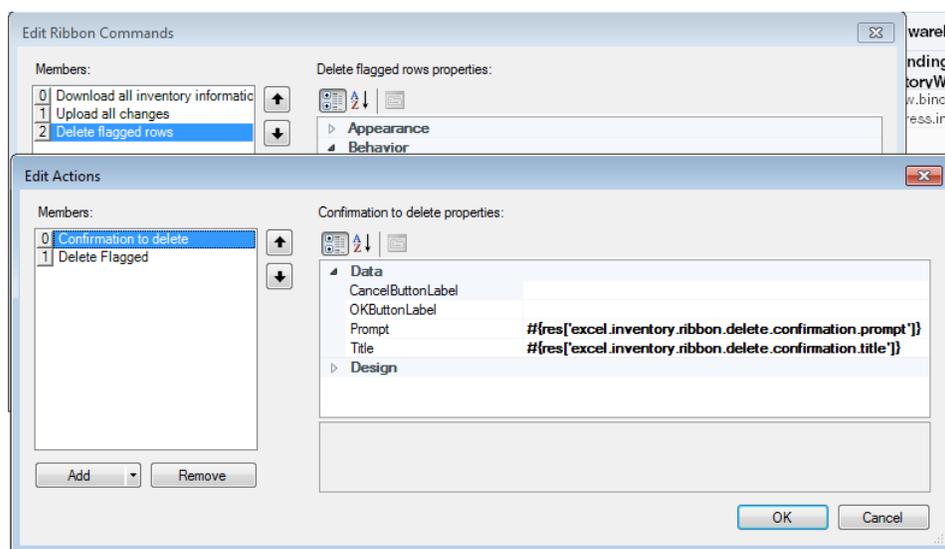
3. Select **Confirmation** and configure its Data properties as described in the following list:
 - `CancelButtonLabel`
Specify an optional EL expression or literal value that evaluates to a message to appear in the **Cancel** button of the dialog.
 - `OKButtonLabel`
Specify an optional EL expression 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**.

 **Note:**

We recommend that you provide values for the `Title` and `Prompt` properties that are specific to your business use case.

Figure 9-15 shows the Edit Action dialog with default attribute values for the **Delete flagged rows** ribbon command in the Summit sample application's `EditAllInventory-DT.xlsx` workbook.

Figure 9-15 Confirmation Action Attributes



What Happens at Runtime: How the Action Set Prompts the User for Confirmation

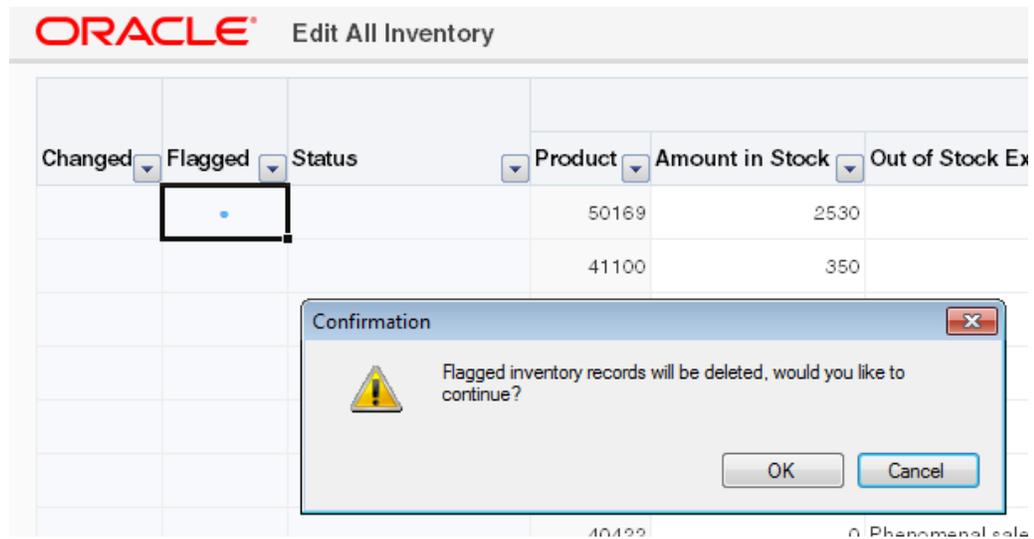
Once the action set is invoked, the user is prompted with a confirmation dialog. If the user clicks **OK**, the next action operation is performed; and if the user clicks **Cancel**, the Action Set execution terminates without an error.

Note:

If the user cancels a Confirmation action, the `FailureActionID` binding does not run.

Figure 9-16 shows the Confirmation dialog that appears when you click the **Delete flagged rows** ribbon command in the Summit sample application's `EditAllInventory-DT.xlsx` workbook.

Figure 9-16 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 a ribbon command) is used. If the label of the container is also empty, then the default value provided by ADF Desktop Integration is used.

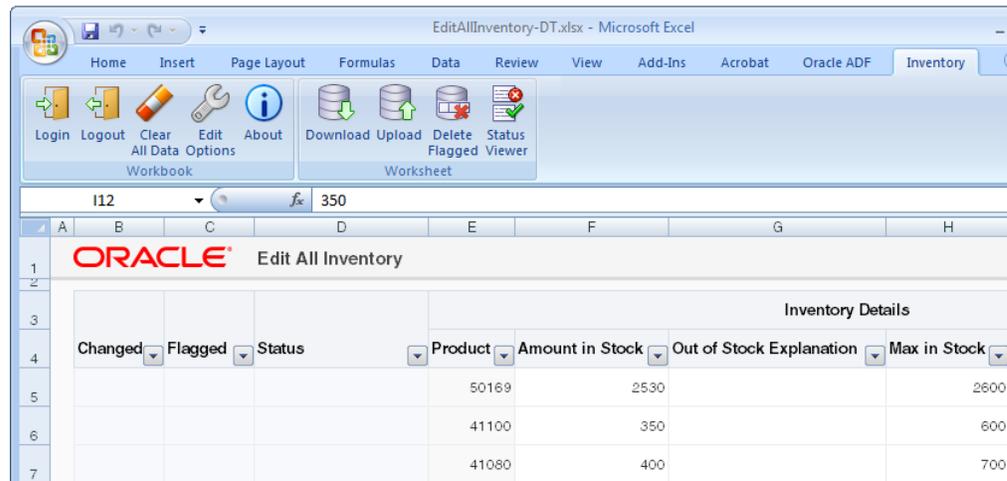
Configuring the Runtime Ribbon Tab

You can add a runtime ribbon tab to the Excel Ribbon in your integrated Excel workbook with ribbon commands to invoke Oracle ADF functionality.

The runtime ribbon tab groups these items into two groups: workbook and worksheet. You configure the workbook group to display ribbon commands to invoke the workbook actions described in [Workbook Actions and Properties](#), while you configure the worksheet group to invoke a range of actions on the active worksheet.

Figure 9-17 shows the **Inventory** runtime ribbon tab in the `EditAllInventory-DT.xlsx` workbook that configures ribbon commands in both the workbook and worksheet groups. The workbook group exposes ribbon commands to invoke the standard default workbook actions while the worksheet group exposes ribbon commands that invoke a number of component actions exposed by an ADF Table component that renders in the worksheet (`Upload`, `DeleteFlaggedRows`, and so on).

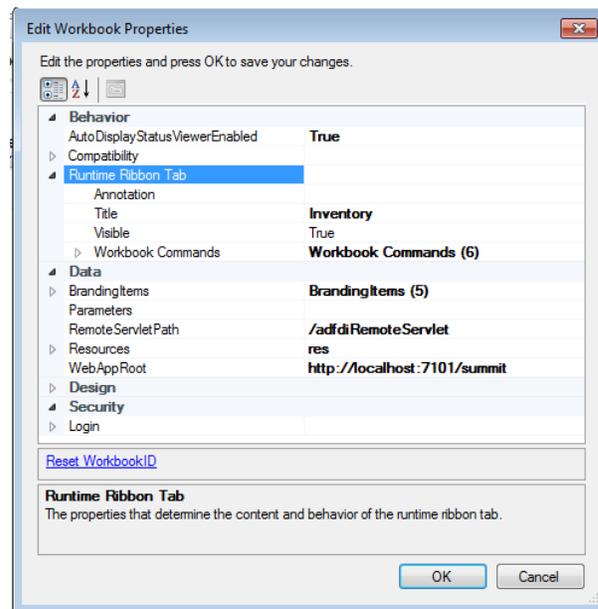
Figure 9-17 Runtime View of Ribbon Tab in EditAllInventory-DT.xlsx



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 the integrated Excel workbook or worksheet is active.

You set the `Visible` workbook property to `True` to make the ribbon tab appear in the Excel Ribbon at runtime. The value you specify for the `Title` property determines the title of the tab that the end user sees at runtime, as illustrated in [Figure 9-18](#).

Figure 9-18 Workbook Properties for Runtime Ribbon Tab





Tip:

You can use the & character in the Title property value to render a key tip in Excel for the runtime ribbon tab. The key tip appears when a user types the Alt key to view available key tips. If, for example, the Title property evaluates to I&nventory, Excel displays an **Inventory** label for the runtime ribbon tab and attempts to assign n as the key tip. In this example, the successful display of n as the key tip requires that no other runtime ribbon tab at the same scope (level) tries to use n as its key tip.

For information about how you define a workbook ribbon command, see [How to Define a Workbook Ribbon Command for the Runtime Ribbon Tab](#). For information about how you configure a worksheet ribbon command, see [How to Configure a Worksheet Ribbon Command for the Runtime Ribbon Tab](#).

How to Define a Workbook Ribbon Command for the Runtime Ribbon Tab

You configure the `Runtime Ribbon Tab` group of workbook properties to define a workbook ribbon command.

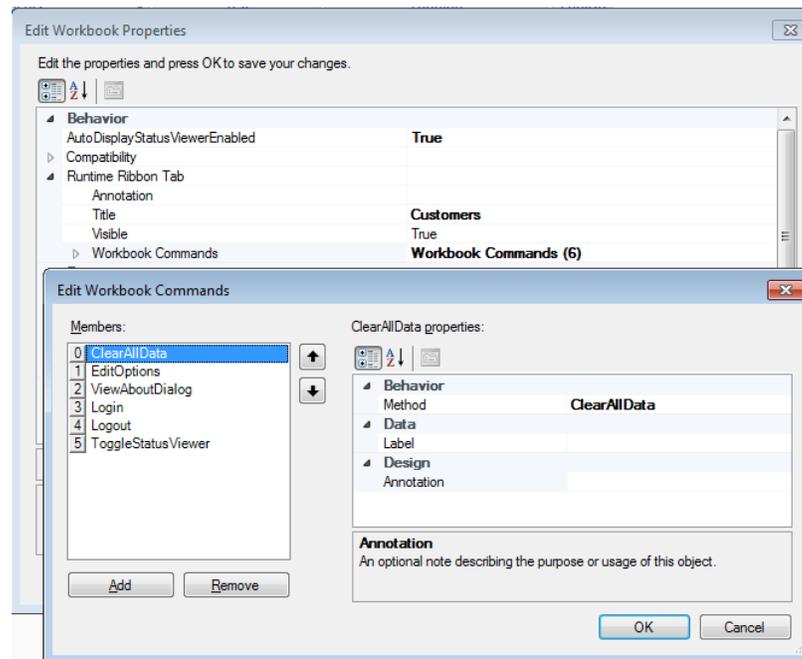
Before you begin:

It may be helpful to have an understanding of the runtime ribbon tab in Excel. See [Configuring the Runtime Ribbon Tab](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Adding Interactivity to an Integrated Excel Workbook](#).

To define a workbook ribbon command:

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 9-19](#).

Figure 9-19 Edit Workbook Commands Dialog

4. Click **Add** and specify values for the properties of the workbook ribbon commands as follows:

- Method

Specify the workbook action that you want the workbook ribbon command to invoke. For the list of available workbook actions, see [Workbook Actions and Properties](#).

- Label

If no label is specified, ADF Desktop Integration uses a default label at runtime.

(Optionally) 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 information about using resource bundles, see [Using Resource Bundles in an Integrated Excel Workbook](#).

5. Click **OK**.

How to Configure a Worksheet Ribbon Command for the Runtime Ribbon Tab

You configure the Ribbon Command group of worksheet properties to define a worksheet ribbon command. By default, no ribbon commands are defined for the worksheet group in the worksheet properties.

Before you begin:

It may be helpful to have an understanding of the runtime ribbon tab in Excel. See [Configuring the Runtime Ribbon Tab](#).

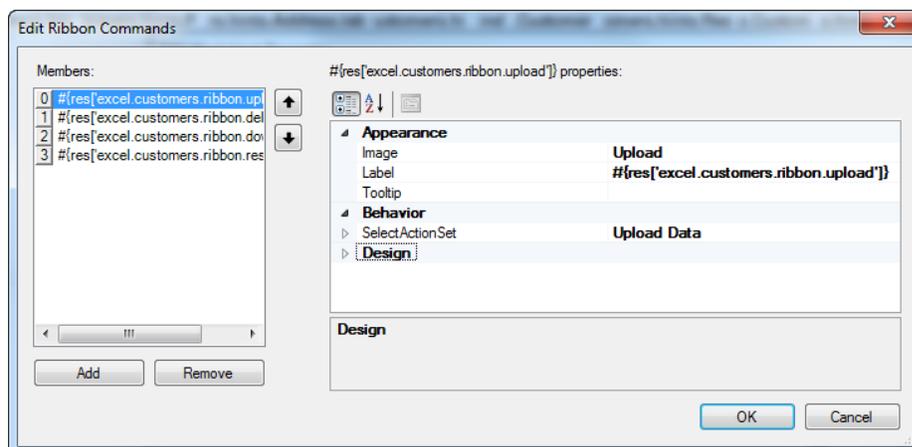
You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Adding Interactivity to an Integrated Excel Workbook](#).

Make sure to set the `Runtime Ribbon Tab.Visible` workbook property to `True`. If the `Runtime Ribbon Tab.Visible` is set to `False`, no runtime ribbon tab appears for this workbook. For information about workbook properties, see [Table A-19](#).

To define a worksheet ribbon command:

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 9-20](#).

Figure 9-20 Edit Ribbon Commands Dialog



4. Click **Add** to add a new ribbon command in the **Members** list of the collection editor.
5. Configure the ribbon command properties as described in [Table 9-4](#).

Table 9-4 Worksheet Ribbon Command Properties

Set this property to...	This value...
SelectActionSet	Specify the type of action(s) that the ribbon command invokes. See Using Action Sets .

Table 9-4 (Cont.) Worksheet Ribbon Command Properties

Set this property to...	This value...
Image	Select an appropriate image for the ribbon command from the dropdown list. For example, if the ribbon command's action set invokes an ADF Table component's <code>Download</code> action, select Download . Choose Generic if the other options do not correspond to the action that the ribbon command invokes. ADF Desktop Integration provides the images that you can use.
Label	Specify text to appear as a label or an EL expression that evaluates to a label at runtime. For information about EL expressions in ADF Desktop Integration, see ADF Desktop Integration EL Expressions .
Tooltip	Specify text to appear as a tooltip or an EL expression that evaluates to a tooltip at runtime. 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.

 **Tip:**

You can use the `&` character in the Label property value to render a key tip in Excel for the ribbon command. The key tip appears when a user types the Alt key to view available key tips. If, for example, the Label property evaluates to `U&pload`, Excel displays an **Upload** label for the ribbon command and attempts to assign `p` as the key tip. In this example, the successful display of `p` as the key tip requires that no other ribbon command at the same scope (level) tries to use `p` as its key tip.

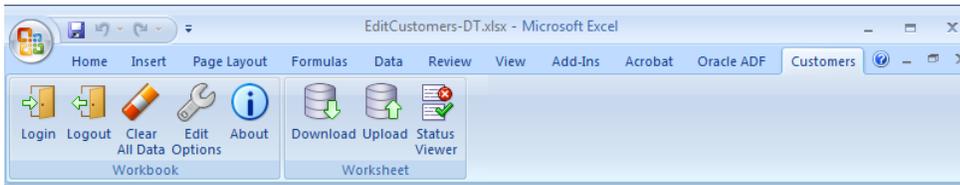
6. Click **OK**.

What Happens at Runtime: Ribbon Commands in the Ribbon Tab

[Figure 9-21](#) shows the **Customers** ribbon tab from the Summit sample application's `EditCustomers-DT.xlsx` workbook. The order and grouping of the workbook-level ribbon commands is always the same at runtime. The worksheet commands appear in the same order as you define them in the Edit Ribbon Commands dialog.

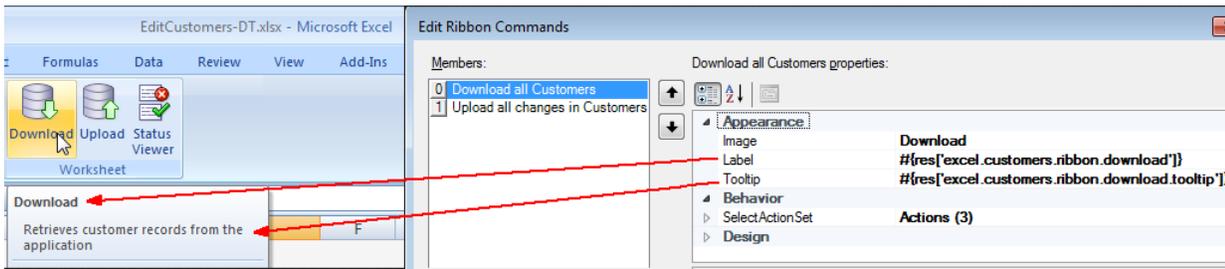
Although the Status Viewer is configured once per workbook and appears in the workbook commands at design time, it appears in the worksheet group at runtime. This is because the Status Viewer is worksheet-specific and displays information about the worksheet that is in focus. If your end users navigate to a non-integrated worksheet and click the Status Viewer ribbon command, a message appears that tells the end user the Status Viewer cannot be used in that worksheet.

Figure 9-21 Ribbon Commands in the Ribbon Tab



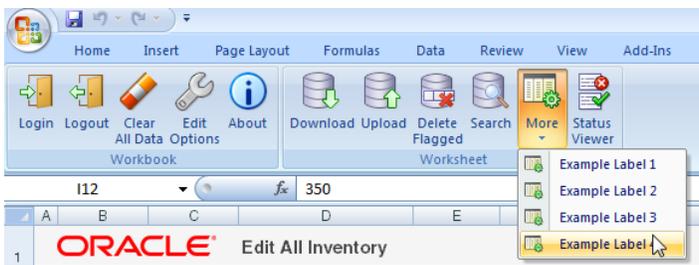
When the user hovers the mouse over the ribbon command with the tooltip, a multi-part tooltip appears. The ribbon command label appears first in bold followed by the text from the `Tooltip` property. Below this text, the add-in name appears. Figure 9-22 shows the tooltip that appears when you hover over the **Download** worksheet ribbon command in the Summit sample application's `EditCustomers-DT.xlsx` workbook.

Figure 9-22 Ribbon Command Displaying a Tooltip



If you define 5 or fewer worksheet-level ribbon commands, each appears in the worksheet group with a large icon. If you define 6 or more worksheet-level ribbon commands, the first 4 ribbon commands appear with a large icon. The remaining ribbon commands appear in a menu labelled **More**, as shown in Figure 9-23.

Figure 9-23 Worksheet's More Ribbon Command Displaying Dropdown List



 **Note:**

The ribbon controls of the toolbar are shared among all open integrated workbooks. If you open two, or more, workbooks using different ribbon commands 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.

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 that interact with your Fusion web application.

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 `CustomerSearch-DT.xlsx` workbook specifies the following relative URL as a value for the page to invoke when a user clicks the **Search 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
```

 **Tip:**

If you want to add a model-driven list picker to a table column, see [Adding a Model-Driven List Picker to an ADF Table Component](#).

 **Note:**

The `Dialog` action does not support ADF task flows.

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 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.

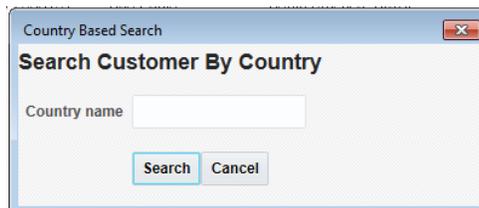
The web page that the action set invokes must contain a reserved HTML `` element that has a case-sensitive ID attribute set to `ADFdi_CloseWindow`.

The following example shows how you can automatically set the value of the `span` element using the `rendered` property of the `f:verbatim` tag.

```
<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 9-24 shows the `searchForm.jspx` page hosted by the `CustomerSearch-DT.xlsx` workbook's browser control.

Figure 9-24 Search Popup Dialog



In scenarios where you cannot use the `rendered` property of the `f:verbatim` tag, you may need to:

1. Create a backing bean that exposes the `Dialog` action's result value as a property.

The following example illustrates what the backing bean code might look like:

```
public class PickDialogBackingBean {
    private String dialogResult = null;

    public PickDialogBackingBean () {
    }

    public void pickerOkListener(ActionEvent actionEvent) {
        setDialogResult("Continue");
        return;
    }

    public String getDialogResult() {
        return dialogResult;
    }

    public void setDialogResult(String result) {
        dialogResult = result;
    }
}
```

```
    }
}
```

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.

The following example illustrates how you use an EL expression to set the value `ADFdi_CloseWindow` to the bean property value.

```
<af:outputText id="otx1" value="<span
id="ADFdi_CloseWindow">#{backing_PickDialog.dialogResult}</span>"
escape="false"/>
```

The following example illustrates how the `actionListener` attribute for a button component in your dialog's JSF page invokes the backing bean:

```
<af:button actionListener="#{backing_PickDialog.pickerOkListener}"
text="OK"/>
```

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.
- `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 custom web page in a modal popup dialog using a .NET 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 within the browser control until they close the browser control, or ADF Desktop Integration closes it.

You may want to add additional actions after the `Dialog` action to take advantage of user choices in your custom page. For example, a user is expected to type a country name in a country-based search. In this scenario, the next logical actions to invoke are `Execute` (a query with the country name the user entered) and the `Download` action for the ADF Table component.

 **Note:**

- 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.

- You should avoid configuring the web page that appears in a popup dialog so that it allows the end user to download an integrated Excel workbook. In that case, the Oracle ADF functionality becomes disabled when the end user opens a workbook downloaded from a popup dialog.
- If you use the HTML `<select>` components, such as list box or dropdown list, note that `<select>` components do not follow `z-order` configuration when the page displays 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.

How to Display a Web Page Search Form in a Popup Dialog

You can use a 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 ribbon command to invoke the `Download` action for the ADF 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 *Creating ADF Databound Search Forms* in *Developing Fusion Web Applications with Oracle Application Development Framework*.

 **Note:**

ADF Desktop Integration does not support usage of the `FindMode` attribute in page definition files. For information about the `FindMode` attribute, see *pageNamePageDef.xml* section of *Developing Fusion Web Applications with Oracle Application Development Framework*.

Before you begin:

It may be helpful to have an understanding of how web pages render in an integrated Excel workbook. See [Displaying Web Pages from a Fusion Web Application](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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 the ribbon command in the Excel worksheet, as described in [How to Configure a Worksheet Ribbon Command for the Runtime Ribbon Tab](#).
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 9-5](#) describes the actions to invoke in sequence.

Table 9-5 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 information about displaying pages from a Fusion web application, see How to Display a Web Page in a Popup Dialog .
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 9-25](#) shows an example from the `CustomerSearch-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 9-25 Ribbon Command Configured to open a Web Page

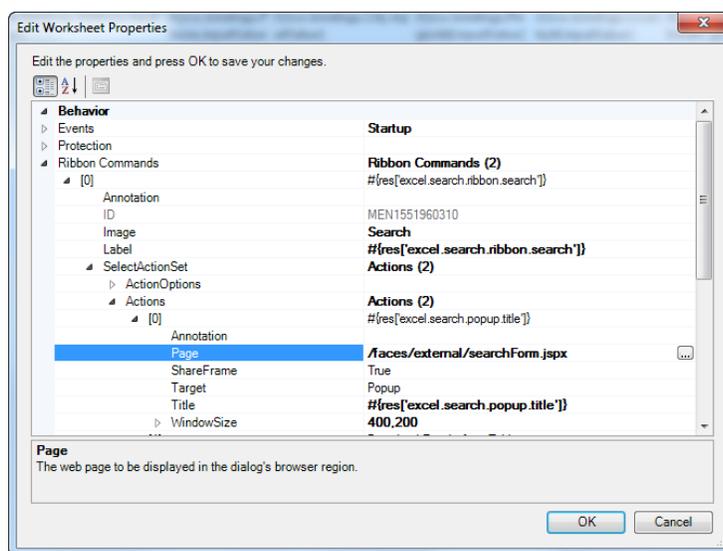
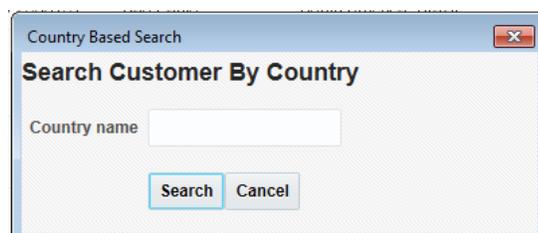


Figure 9-26 shows the web page search form at runtime.

Figure 9-26 Web Page Search Form



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.

 **Note:**

- 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).

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.

 **Note:**

- 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**.
See About Popup Dialogs, Menus, and Windows in *Developing Web User Interfaces with Oracle ADF Faces*.
- The `Dialog.Page` property does not accept EL expressions.

Sending Data Between an Integrated Excel Worksheet and a Fusion Web Application Page

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` or `RowUpSyncNoFail` 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.
- One way to capture data state from the web page (if necessary) is for logic in the web page's backing bean to retrieve data from its data bindings and to transfer that data into the bindings for the integrated Excel worksheet.
- 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 a `DoubleClickActionSet`, the server-side model must be in the same state after executing the action set as it was before executing the action set. To achieve this, make sure the ADF Table component supports row-level action set model management, as described in [How to Enable Row-Level Action Set Model Management](#).

For information about synchronizing data between an integrated Excel workbook and a Fusion web application, see [Using an Integrated Excel Workbook Across Multiple Web Sessions](#). For information about worksheet actions and ADF Table component actions, see [ADF Desktop Integration Component Properties and Actions](#).

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 information about data control frames in a Fusion web application, see [Sharing Data Controls Between Task Flows in *Developing Fusion Web Applications with Oracle Application Development Framework*](#).

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 supports 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.

Localizing Fusion Web Application Pages

The localization of content in the Fusion web application pages that appear in the `Dialog` action of an integrated Excel workbook is governed by the same mechanism that localizes pages elsewhere in the Fusion web application. For information about how to localize the UI strings that the ADF Faces components of Fusion web applications render, see *Internationalizing and Localizing Pages in Developing Web User Interfaces with Oracle ADF Faces*. For information about localization in ADF Desktop Integration, see [Localization in ADF Desktop Integration](#).

Using Row-Level Action Sets in a Table Column

You may want to configure an action set that executes in the context of the current table row whenever the end user double-clicks a column.

For example, you might configure an ADF Table component column `DoubleClickActionSet` to launch a custom dialog that enables the end user to select server-side row attribute values for the current table row, as described in [How to Add a Custom Popup Picker Dialog to an ADF Table Column](#).

Row-Level Action Set Model Management

You can automate the management of the server-side model state when table-based row-level action sets that may alter the model state are invoked. ADF Desktop Integration creates a save point before invoking the actions in the action set and restores to the save point after the action set runs. This ensures that the model state after the action set was invoked remains the same if the action set is aborted or cancelled and reverts back to the same state as it was before the action set was invoked.

For insert worksheet rows, ADF Desktop Integration automatically creates a temporary server-side row that can be used during the action set. For both insert and update worksheet rows, ADF Desktop Integration automatically reverts any model changes that occur during the action set (including the temporary row in the insert case).

This is useful if you have integrated Excel workbooks with ADF Table components configured with row action sets that modify the server-side model. For example, a column component double-click action set that launches a custom dialog to select

server-side row attribute values for the current worksheet row, as described in [How to Add a Custom Popup Picker Dialog to an ADF Table Column](#).

How to Enable Row-Level Action Set Model Management

To manage the server-side model state with a row-level action set, set the following workbook property to `True`:

```
Compatibility.TableComponents.RowActionSetModelMgmtEnabled
```

Before you set the `RowActionSetModelMgmtEnabled` property to `True`, note that ADF Desktop Integration creates a `DataControl` savepoint to capture and restore the model state. So, make sure that the `DataControl` providers of your Fusion web application support savepoints.

To enable row-level action set model management:

1. Open the integrated Excel workbook.
2. In the Workbook group of the Oracle ADF tab, click **Workbook Properties**.
3. If the ADF Table component supports row inserts (`InsertRowEnabled` row-level action), set the `InsertBeforeRowActionID` action to create a temporary server-side row during a row-level action set.

If your use case requires a separate action to create a temporary row for row-level action sets, configure the `InsertTempRowActionID` property.

Note:

If the `InsertBeforeRowActionID` action is sufficient for creating a temporary server-side row during a row-level action set, `InsertTempRowActionID` should be left blank.

4. In the Edit Workbook Properties dialog, if not set already, set the `Workbook.Compatibility.TableComponents.RowActionSetModelMgmtEnabled` property to `True`.

Note that the `RowActionSetModelMgmtEnabled` property is set to `False` in integrated Excel workbooks created with versions of ADF Desktop Integration that did not include this feature.

5. Click **OK** to close the Edit Workbook Properties dialog.

Note:

For integrated Excel workbooks created with older versions of ADF Desktop Integration, set the `RowActionSetModelMgmtEnabled` property to `True` and remove any custom configuration or code that manages model state during row-level action sets.

What Happens at Runtime: RowActionSetModelMgmtEnabled is Set to True

If `RowActionSetModelMgmtEnabled` property is set to `True`, ADF Desktop Integration automatically manages the model state while the row-level action set runs.

For an insert worksheet row, a temporary server-side row is automatically created when the action set runs and is automatically removed after a successful upload. When the `InsertTempRowActionID` action is configured, it gets invoked to create the temporary server-side row. Otherwise, the `InsertBeforeRowActionID` action is invoked to create the temporary server-side row instead.

If neither the `InsertTempRowActionID` nor `InsertBeforeRowActionID` actions are configured, no action is invoked for insert rows. The `InsertTempRowActionID` action is ignored if `InsertRowEnabled` is set to `False`.

When the end-user invokes a row-level action set configured in an ADF Table component and the row-level action set contains one or more actions that may alter the model state, ADF Desktop Integration does the following:

1. Positions the server-side row (for update worksheet rows only)
2. Creates a data control save point
3. Invokes the `InsertTempRowActionID` or `InsertBeforeRowActionID` action to create a temporary server-side row (for insert worksheet rows only)
4. Invokes the actions in the action set
5. Restores to the previously created save point after the action set invocation is completed, regardless of how it terminates including:
 - Upload successful
 - Upload failure
 - End user clicks the Cancel button

 **Note:**

The following actions (or action types) may alter the model state:

- `Table.RowUpSync`
- `Table.RowDownSync`—only applies to insert rows
`RowDownSync` for an existing row does not alter the model state.
- `Table.RowUpSyncNoFail`
- `Worksheet.UpSync`
This action is also supported in row-level action sets.

- `ADFmAction`
- `Dialog`

The `Dialog` action may change the model state if `ShareFrame` is set to `True` and the web page is part of the same web application.

If the `RowActionSetModelMgmtEnabled` property is set to `False`, you must explicitly manage the creation and deletion of the temporary server-side row while the action set runs.

How to Synchronize Changes from ADF Table Component Using `RowUpSyncNoFail`

A row-level action set may contain `ADFmAction` or `Dialog` actions that depend on the current state of the model to complete successfully. The `Table.RowUpSync` action sends the current value of individual table rows from the worksheet to the model layer in the Fusion web application. The `Table.RowUpSync` action requires all cells in a table row to contain valid data for the action to complete successfully. For example, in a newly-inserted row, all required attributes must have valid values for a `Table.RowUpSync` action to complete. In contrast, the `Table.RowUpSyncNoFail` action synchronizes valid values from cells in a table row and ignores any validation failures for invalid values. Like `RowUpSync`, the `RowUpSyncNoFail` action is intended for use in the row-level action sets of table columns that supports `DoubleClickActionSet`.

Enable row-level action set model management when using `RowUpSyncNoFail`, as described in [How to Enable Row-Level Action Set Model Management](#).

To synchronize changes from ADF Table Component using `RowUpSyncNoFail`:

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.
3. Click the browse (...) icon of the `Columns` property.
4. In the Edit Columns dialog, select the column, and click the browse (...) icon of the `UpdateComponent` property.
5. Add the ADF Table component `RowUpSyncNoFail` action to the list of actions of the column's `DoubleClickActionSet`.

6. Click **OK**.

What Happens at Runtime: RowUpSyncNoFail Action is Invoked

When the `RowUpSyncNoFail` action is invoked, data values from the current table row are uploaded to the server and common failures, error reporting, and error handling are ignored. Fatal errors, such as the server being unavailable, will be reported.

The `RowUpSyncNoFail` action modifies the state of the model and the changes are not reverted on error. Consequently, it is possible that a call to `RowUpSyncNoFail` may leave the row in the model with values that would cause row validation to fail. This may in turn impact the behavior of subsequent calls to other methods, such as `Table.Upload`. For this reason, you should ensure that row-level action set model management is enabled.

How to Add 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 *Creating Databound Selection Lists and Shuttles* in *Developing Fusion Web Applications with Oracle Application Development Framework*.

Before you begin:

It may be helpful to have an understanding of using row-level action sets. See [Using Row-Level Action Sets in a Table Column](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Adding Interactivity to an Integrated Excel Workbook](#).

Make sure the ADF Table component supports row-level action set model management, as described in [How to Enable Row-Level Action Set Model Management](#), if you want the custom pick dialog to function correctly in an ADF Table component that supports an insert component. Without row-level action set model management enabled, no temporary insert rows will be created at runtime.

To invoke a custom pick dialog from an ADF Table component column:

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 9-6](#).

Table 9-6 DoubleClickActionSet Properties

Add this action...	To...
ComponentAction	Invoke the ADF Table component's <code>Table.RowUpSync</code> action to synchronize any pending changes in the current row of the ADF Table component to the Fusion web application.
Dialog	Configure the <code>Dialog</code> action to invoke the pick dialog page from the Fusion web application. Set the <code>Dialog</code> action's <code>ShareFrame</code> property to <code>True</code> . See Displaying Web Pages from a Fusion Web Application .
ComponentAction	Invoke the ADF Table component's <code>Table.RowDownSync</code> 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.

7. Click **OK**.

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 9-1](#) uses `HYPERLINK` function to navigate to `http://www.oracle.com/technetwork/developer-tools/adf/overview/index-085534.html` when end user clicks the component.

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.

Example 9-1 HYPERLINK Function

```
=T( "=HYPERLINK( \"http://www.oracle.com/technetwork/developer-tools/adf/overview/index-085534.html\", \"#{res['excel.workbook.powerby']}\" ) )
```

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. See [Using EL Expression to Generate an Excel Formula](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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 9-1](#), 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**.

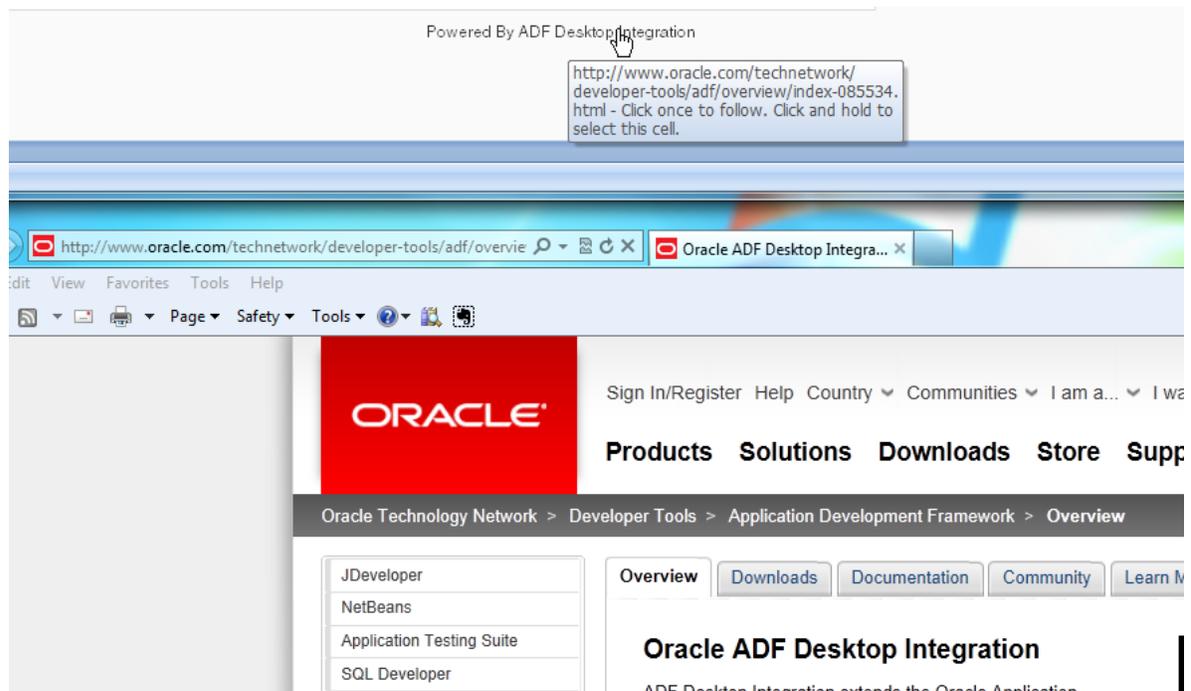
What Happens at Runtime: How a Cell Displays a Hyperlink using an EL Expression

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 9-27](#) shows the runtime view of the example configured in [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 9-27 ADF Output Text Component Configured to Display a Hyperlink



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 information about Excel functions, see the Function reference section in Excel's online help documentation.

How to Calculate the Sum of a Table-Type Component Column

The following task illustrates how you use the Excel functions `AVERAGE` and `OFFSET` to calculate the average of the column labeled **Salary** 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 `AVERAGE` function calculates the average value 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. See [Using Calculated Cells in an Integrated Excel Workbook](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Adding Interactivity to an Integrated Excel Workbook](#).

Make sure that the ADF Table component's `RowActions.AutoConvertNewRowsEnabled` property is set to `False`. For information about this property, see [ADF Table Component Properties](#).

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, J2.
2. Write the Excel formula that performs a calculation on a range of cells at runtime. For example:

```
=AVERAGE(OFFSET(J2,1,0):OFFSET(J4,-1,0))
```

where `AVERAGE` calculates the average value in the range of cells currently referenced by J2 and J4.

[Figure 9-28](#) shows the design time view of the Excel formula in the integrated Excel workbook.

Figure 9-28 Design Time View of Excel Formula in an Integrated Excel Workbook

B	C	D	E	F	G	H	I	J	K
#[_ADFDIres[COMPONENTS_TABLE_CHANGED_COL_LABEL]]	#[_ADFDIres[COMPONENTS_TABLE_CHANGED_FLAGGED_COL_LABEL]]	#[_ADFDIres[COMPONENTS_TABLE_CHANGED_STATUS_COL_LABEL]]	#{bindings.SEmpView1.hints.Id.label}	#{bindings.SEmpView1.hints.FirstName.label}	#{bindings.SEmpView1.hints.LastName.label}	#{bindings.SEmpView1.hints.Email.label}	#{bindings.SEmpView1.hints.StartDate.label}	#{bindings.SEmpView1.hints.Salary.label}	#[_ADFDIres[COMPONENTS_TABLE_ROWKEY_COL_LABEL]]
			#{row.bindings.Id.inputValue}	#{row.bindings.FirstName.inputValue}	#{row.bindings.LastName.inputValue}	#{row.bindings.Email.inputValue}	#{row.bindings.StartDate.inputValue}	#{row.bindings.Salary.inputValue}	
							Average Salary	#DIV/0!	

3. Save your changes and switch to runtime mode to test that the Excel formula you entered evaluates correctly.

What Happens at Runtime: How Excel Calculates the Sum of a Table-Type Component Column

Figure 9-29 shows the runtime view in the integrated Excel workbook when the Excel formula shown in Figure 9-28 is evaluated. The Excel formula calculates the average of the values in the range of cells that you specified in design mode.

Figure 9-29 Runtime View of Excel Formula in an Integrated Excel Workbook

=AVERAGE(OFFSET(J2,1,0):OFFSET(J16,-1,0))									
B	C	D	E	F	G	H	I	J	K
Changed	Flagged	Status	ID	First Name	Last Name	Email	Start Date	Salary	Key
			1	Carmen	Velasquez	cvelasqu@summit.com	03/03/2010 00:00	USD 2,500.00	
			2	LaDoris	Ngao	lngao@summit.com	08/03/2010 00:00	USD 1,450.00	
			3	Midori	Nagayama	mnagayam@summit.com	17/06/2010 00:00	USD 1,400.00	
			4	Mark	Quick-To-See	mquickto@summit.com	17/06/2010 00:00	USD 1,450.00	
			5	Audry	Ropeburn	aropebur@summit.com	17/06/2010 00:00	USD 1,550.00	
			6	Molly	Urguhart	murguhar@summit.com	18/01/2011 00:00	USD 1,200.00	
			7	Roberta	Menchu	rmenchu@summit.com	14/05/2010 00:00	USD 1,250.00	
			8	Ben	Biri	bbiri@summit.com	07/04/2011 00:00	USD 1,100.00	
			9	Antoinette	Catchpole	acatchpo@summit.com	09/02/2011 00:00	USD 1,300.00	
			10	Marta	Havel	mhavel@summit.com	27/02/2011 00:00	USD 1,307.00	
			11	Colin	Magee	cmagee@summit.com	14/05/2010 00:00	USD 1,400.00	
			12	Henry	Giljum	hgiljum@summit.com	18/02/2011 00:00	USD 1,490.00	
			13	Yasmin	Sedeghi	ysedeghi@summit.com	18/02/2011 00:00	USD 1,515.00	
							Average Salary	USD 1,454.77	

Configuring Automatic Row Refresh in an ADF Table Component

An ADF Table component can be configured to automatically invoke custom business logic each time an end user modifies values in a table row.

Assume that an ADF Table component contains various columns including Current Salary (read-only column), Raise Percentage, Raise Amount, and Proposed Salary. An end user edits a value in any of the latter three columns and the values in the other two writable columns are expected to recalculate automatically and immediately. Later, during invocation of the ADF Table component `Upload` action, all modified and recalculated values upload for each changed row. For example, if the end user changes Raise Amount, Raise Percentage is set to $(\text{Raise Amount} / \text{Current Salary})$ and Proposed Salary is set to $(\text{Current Salary} + \text{Raise Amount})$. Similar changes occur when the end user enters values in the other editable columns.

Implementing the functionality described so far requires you to write a custom method where you specify the business logic to invoke when ADF Desktop Integration triggers a row refresh. You also need to configure the `AutoRefreshRowEnabled` and `AutoRefreshRowActionID` properties in the ADF Table component's `RowActions` group of properties. Finally, you identify the columns in the ADF Table component where an end user editing a row value triggers an automatic row refresh. You do this by setting the column's `TriggersRowRefresh` property to `True`.

Be aware of the impact that implementing automatic row refresh can have on performance. See [What You May Need to Know About Automatic Row Refresh in an ADF Table Component](#).

How to Write Custom Methods for Use in Automatic Row Refresh

ADF Desktop Integration invokes the custom method that you write for every row involved in the automatic row refresh.

You can give the method an arbitrary name, but the method signature must include a `java.util.ArrayList` parameter that identifies the attributes (ID property of the ADF Table component's columns) to perform a refresh of row data, as shown by the following example:

```
public void refreshRow (ArrayList attrIds)
```

The custom method that you write should:

- Determine what to do with null values.
- Update other attributes in the same row to achieve the business goal.
- Handle exceptions properly:
 - You can throw exceptions in the method. The message that you associate with the exception you throw will be reported to the end user, so write a message that the end user understands.
 - Catch and handle unexpected exceptions in the custom method.

Do not write code in the custom method that changes the iterator status or that commits ADF Model layer changes.

ADF Desktop Integration handles the following changes for you, so you do not need to write code in your custom method.

- Positions the iterator on the target row.
- Transfers attribute values from the ADF Table component to the Fusion web application.
- Automatically reverts server-side changes.

Once you complete your custom method, expose it as a method action binding in the page definition file that is associated with your integrated Excel worksheet. Verify that the method action binding appears in the bindings palette of the ADF Desktop Integration task pane. You may need to reload the page definition file, as described in [How to Reload a Page Definition File in an Excel Workbook](#), to accomplish this.

How to Configure Automatic Row Refresh in an ADF Table Component

Configure the ADF Table component to enable the automatic row refresh of data, specify the method action to invoke when a refresh occurs, and the column(s) that trigger a refresh.

Before you begin:

Write a custom method that executes the business logic you want to invoke when an end user triggers an automatic row refresh. For information about this custom method

and automatic row refresh, see [How to Write Custom Methods for Use in Automatic Row Refresh](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Adding Interactivity to an Integrated Excel Workbook](#).

To configure automatic row refresh 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 following `RowActions` properties in the Edit Component: ADF Table dialog that appears.

Table 9-7 RowAction Properties to Enable Automatic Row Refresh

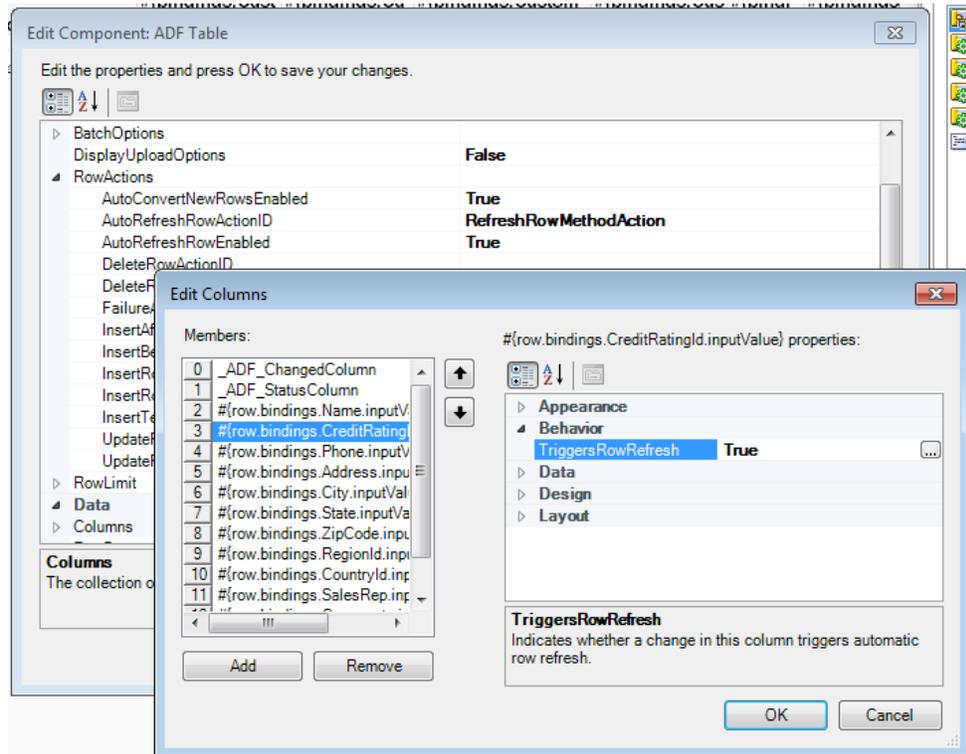
Set this property...	To...
<code>AutoRefreshRowActionID</code>	Specify the method action binding that invokes the custom method with the business logic that executes when an end user triggers an automatic row refresh. You do not need to specify a method action binding if you want to perform an automatic row refresh to trigger server-side validation or refresh the value of a transient attribute. For information about the custom method, see How to Write Custom Methods for Use in Automatic Row Refresh .
<code>AutoRefreshRowEnabled</code>	When <code>True</code> , ADF Desktop Integration evaluates the <code>TriggersRowRefresh</code> property of the columns in this table.

4. 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 ADF Table component.
5. Set the `TriggersRowRefresh` property to `True` or to an EL expression that evaluates to `True` for each column that contains rows where you want to trigger an automatic row refresh if an end user edits a value.

[Figure 9-30](#) shows the dialog where you set the `TriggersRowRefresh` property.

Figure 9-30 TriggersRowRefresh Property to Configure Automatic Row Refresh



6. Click **OK**.

What Happens at Runtime: Automatic Row Refresh in an ADF Table Component

ADF Desktop Integration performs automatic row refresh on ADF Table component cells in columns that you have configured to enable this behavior by setting the column's `TriggersRowRefresh` property to `true`.

ADF Desktop Integration inspects the `TriggersRowRefresh` property of each ADF Table component column involved in the user edit and terminates the automatic row refresh operation if it does not find any columns where the property is set to `True`. Otherwise, it adds the value of the `ID` property of those columns that have the `TriggersRowRefresh` property set to `True` to the `ArrayList` parameter of the custom method.

A progress bar appears if the estimated processing time is significant.

ADF Desktop Integration creates a save point before processing changes. It then proceeds to perform the following changes for each row that needs to be refreshed:

1. Positions the tree binding iterator to the correct row for pending update rows and creates a temporary row for pending insert rows.
2. Transfers all attributes in the row from the worksheet into the ADF Model layer's iterator row. Exceptions (if any) are recorded.

3. Invokes the custom method that you specified as the method action binding for the ADF Table component's `RowActions.AutoRefreshRowActionID` property.
4. Refreshes the row data and reports exceptions (if any).
 - a. If no exceptions occur in Steps 2 and 3, the Status column and the error details for the refreshed row are cleared.
 - b. If exceptions reported in Steps 2 and 3 are `oracle.jbo.attrValException` for columns with the `TriggersRowRefresh` property set to `False`, the following occurs:
 - i. The Status column for the row updates with an "Invalid" entry. Details appear in the Status Viewer.
 - ii. Row data refreshes only in columns that do not report errors.
 - c. Otherwise, the following occurs:
 - i. The Status column for the row updates with an "Invalid" entry and details appear in the Status Viewer.
 - ii. The row does not refresh.
5. ADF Desktop Integration restores the save point that it created before it began the automatic row refresh. It restores the save point irrespective of the result of the automatic row refresh operation (succeeds or fails). End users have to invoke the ADF Table component's `Upload` action to commit the changes to the server side that are visible in the ADF Table component after the automatic row refresh.

What You May Need to Know About Automatic Row Refresh in an ADF Table Component

Note the following points if you enable automatic row refresh in an ADF Table component:

- It can impact data entry performance in the ADF Table component because automatic row refresh makes a request to the Fusion web application and executes the business logic in the custom method. The degree to which it affects data entry performance depends on the computational complexity of the business logic and other factors, such as network latency. If, for example, network latency is high, end users may see a significant delay before the row refresh completes after data entry.
- ADF Desktop Integration sends all attributes in a row from the client to the server and then from the server to the client. This includes modified and unmodified attributes.
- Many end user gestures in an ADF Table component can alter the contents of multiple cells at the same time. When an end user performs a multiple cell edit, ADF Desktop Integration examines the cells in the modified range with the set of columns in the ADF Table component for which the `TriggersRowRefresh` property is set to `True`. An automatic row refresh is performed for each table row that contains a modified cell in a column with the `TriggersRowRefresh` property set to `True`.
- The errors reported by automatic row refresh do not block the ADF Table component's `Upload` action. The values used by the `Upload` action are those in the row after automatic row refresh completes, including the values that caused errors. For this reason, apply required constraints during invocation of the `Upload` action.

Using Macros in an Integrated Excel Workbook

You can define and run macros based on Excel events in an integrated Excel workbook. ADF Desktop Integration reacts to Excel events. An example of an Excel event is the change event that occurs when something in an Excel worksheet changes.

Excel events can occur when an end user or a macro perform an action (for example, insert a new row). ADF Desktop Integration reacts to the Excel event. While ADF Desktop Integration triggers code in response to the Excel event, all further Excel events are suppressed.

Assume, for example, that you write a macro in your integrated Excel workbook that the workbook triggers when a change event occurs in a particular cell. If an end user changes the cell, the Excel event occurs and the macro executes. However, if ADF Desktop Integration changes the cell, no Excel event occurs and the macro does not execute.

For information about Excel events, see Microsoft's documentation.

10

Configuring the Appearance of Your Integrated Excel Workbook

Configure the appearance of an integrated Excel workbook using predefined and custom styles in Excel, EL expressions to dynamically apply styles to Oracle ADF components in the workbook at runtime, and use labels to brand the Excel workbook. This chapter includes the following sections:

- [About Configuring the Appearance of an Integrated Excel Workbook](#)
- [Working with Styles](#)
- [Applying Styles Dynamically Using EL Expressions](#)
- [Using Labels in an Integrated Excel Workbook](#)
- [Branding Your Integrated Excel Workbook](#)
- [Displaying Tooltips in ADF Desktop Integration Components](#)
- [Using Worksheet Protection](#)
- [Using ADF Desktop Integration EL-based Properties with Custom Attribute Properties](#)

About Configuring the Appearance of an Integrated Excel Workbook

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.

Integrated Excel Workbook Configuration Use Cases and Examples

You can customize the appearance of ADF Desktop Integration components using styles. For example, [Figure 10-1](#) shows various styles applied to the columns of ADF Table in `EditCustomers-DT.xlsx`.

Figure 10-1 Styles Applied to Columns of ADF Table in `EditCustomers-DT.xlsx`

Changed	Status	* Name	Credit Rating	Address	City	Zip Code	Country	Sales Rep.	Comments	Key
		Zebra's Bicycles	Fair	3910 Collier Ave	Norfolk	23508	USA	Magee		
		Superior Bicycle	Fair	538 Superior Ave E	Cleveland	44114	USA	Magee		
		Bicycle World	Fair	5						
		Schindler's Sports	Fair	4						
		Barry's Basketball	Fair	56 E Superior St	Chicago	60611	USA	Magee		
		Gavin Sporting Goods	Fair	1935 SE Hawthorne Blvd	Portland	97214	USA	Magee		
		MoreAndMoreStuffz	Fair	3501 McKinney Ave	Dallas	75204	USA	Magee		
		BuyMyJunk	Fair	5610 E Mockingbird Ln	Dallas	75206	USA	Magee		
		Everything Under the	Fair	1028 F 6th St	Tucson	85719	USA	Magee		

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. The following sections describe other functionality 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. See [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. See [Internationalizing Your Integrated Excel Workbook](#).

Working with Styles

ADF Desktop Integration provides a mechanism to apply Excel-named styles to 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 component support the properties `HeaderStyleName` and `CellStyleName` that determine styles to apply at runtime.

Predefined Styles in ADF Desktop Integration

Many properties have default values that are drawn from a predefined list of ADF Desktop Integration styles. For example, the `HeaderStyleName` property's default value is `Column Header`, one of the predefined styles in ADF Desktop Integration. ADF Desktop Integration automatically adds these predefined styles to the Excel workbook when it is enabled for use with ADF Desktop Integration. The predefined styles that ADF Desktop Integration provides are consistent with the Oracle Alta UI, described in <http://www.oracle.com/webfolder/ux/middleware/alta/index.html>.

The following is the list of predefined styles:

- Styles for forms:
 - Form Header
 - Form SubHeader
 - Input Text
 - Label
 - Output Text
- Styles for tables:
 - Column Header
 - Data Cell
 - Indicator Cell
 - Key Cell
 - Read-only Cell
 - Status Cell
- Branding Area

Tip:

Microsoft Excel has a Merge Styles dialog (accessed from the Styles gallery in the **Home** runtime ribbon) that allows you to merge all the named styles from one workbook to another workbook.

You may create additional styles for use in your Excel workbook. For example, to add a date-specific formatting, you can duplicate `Data Cell`, call it `My Date Cell`, and add your date-specific formatting.

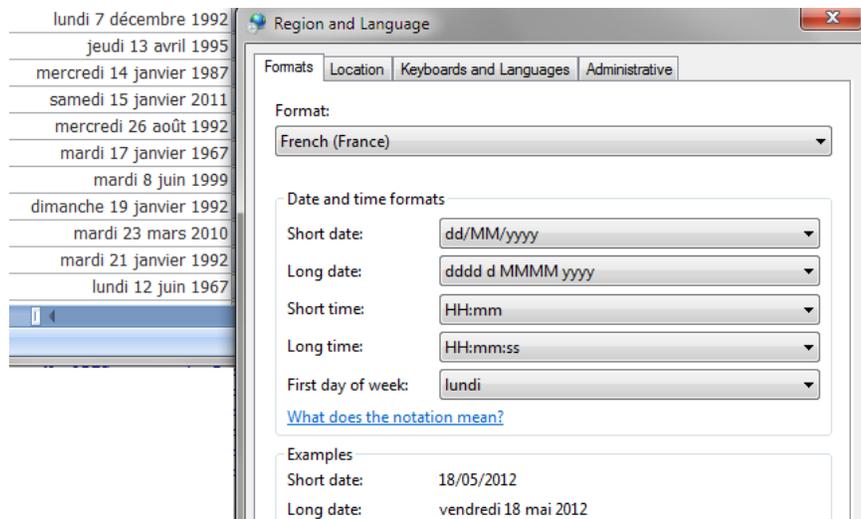
Once you have decided what styles to apply to the ADF Desktop Integration components at runtime, you can write EL expressions to associate a style with a component. The ADF Desktop Integration component properties that include `StyleName` in their name take an EL expression as a value. The ADF Label 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 information about creating new styles and merging styles into a workbook, see Excel's documentation.

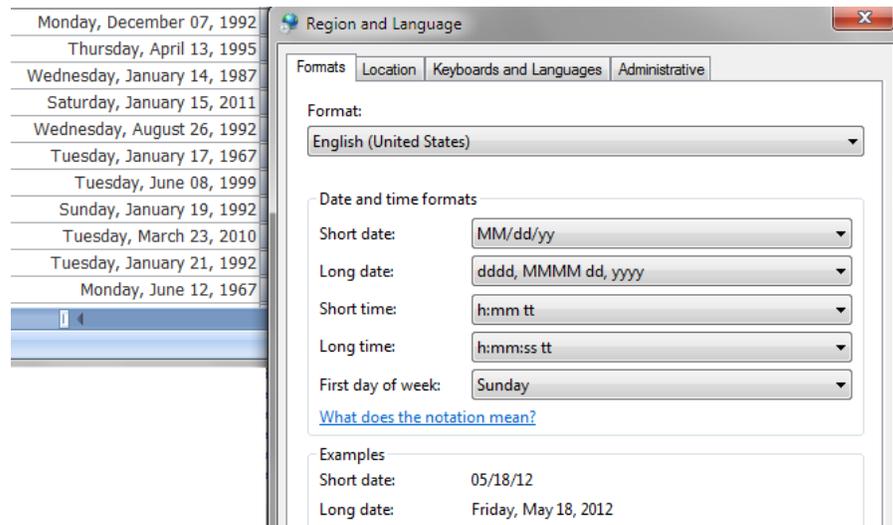
Excel's Date Formats and Microsoft Windows' Regional and Language Options

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 10-2](#) shows the Regional and Language Options dialog where an example of a Date type that formats dates in a cell using French (France) conventions appears.

Figure 10-2 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 10-3](#), the cells that are formatted with the style in [Figure 10-2](#) use the **English (United States)** conventions.

Figure 10-3 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.

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. See [Working with Styles](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Configuring the Appearance of an Integrated Excel Workbook](#).

To apply a style:

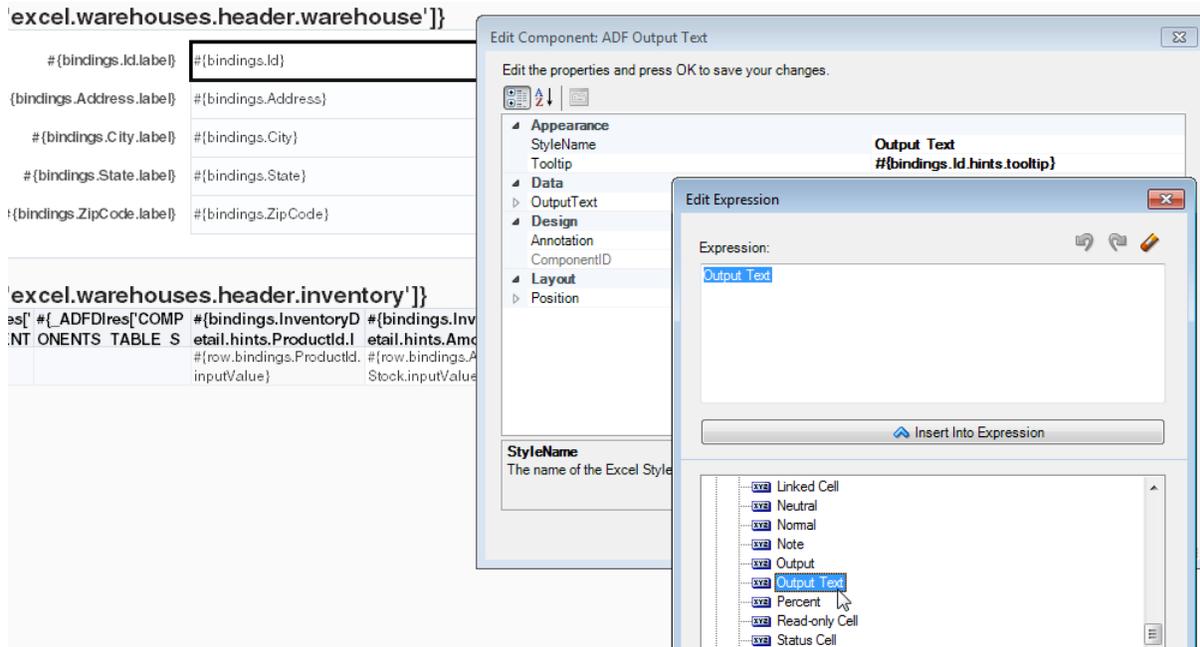
1. In the integrated Excel workbook, select the cell that references the Oracle ADF component you want to modify and then click the **Edit Properties** button in the **Oracle ADF** tab.
2. Select the **StyleName** property and click the browse (...) icon to display the Edit Expression dialog.
3. Expand the **Styles** node and select the style to apply to cell at runtime.

For example, apply an `Output Text` style to the Binding Warehouse ID output text field.

4. Click **Insert Into Expression** to insert the selected style into the Expression field.

Figure 10-4 shows the Edit Expression dialog where we define the style for the `OutputText` component that displays the **Warehouse ID** in the Summit sample application's `EditWarehouseInventory-DT.xlsx` workbook.

Figure 10-4 Edit Expression Dialog Applying a Style



5. Click **OK**.

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 style is applied to a cell that references an ADF component, the ADF component overwrites that style at runtime with any property values (font, alignment, and so on) defined by the style referenced by its `StyleName` property.

For example, Figure 10-5 shows the runtime appearance of the **Warehouse ID** field defined by the `Output Text` style in the Summit sample application's `EditWarehouseInventory-DT.xlsx` workbook.

Figure 10-5 Runtime Appearance of Component with Style Applied

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.

The following examples show different contexts where you can use EL expressions to determine the behavior and appearance of ADF components at runtime. [Example 10-1](#) 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 10-2](#) uses a mixture of Excel formulas and ADF binding expressions to handle errors and type conversion. [Example 10-3](#) demonstrates how to use a custom attribute property to specify the style. For information about custom attribute properties, see [Using ADF Desktop Integration EL-based Properties with Custom Attribute Properties](#).

Example 10-1 Applying a Style Dynamically During Download

```
=IF("#{bindings.Status}" = "Closed", "MyReadOnlyStyle", "MyReadWriteStyle")
```

Example 10-2 EL Expressions to Handle Errors and Type Conversion

```
=IF(ISERROR(VALUE("#{bindings.DealSize}")), "BlackStyle",  
IF(VALUE("#{bindings.DealSize}") > 300, "RedStyle", "BlackStyle"))
```

Example 10-3 Using a Custom Attribute Property to Specify the Style

```
#{bindings.EmpCompView1.hints.EmployeeId.diCellStyle}
```

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 10-1](#), ADF Desktop Integration first

determines that value of the binding expression, `{bindings.Status}`, in the following Excel formula:

```
=IF("#{bindings.Status}" = "Closed", "MyReadOnlyStyle", "MyReadWriteStyle")
```

It then replaces the binding expression with the runtime value, as in the following example, where the expression evaluated to `Closed`:

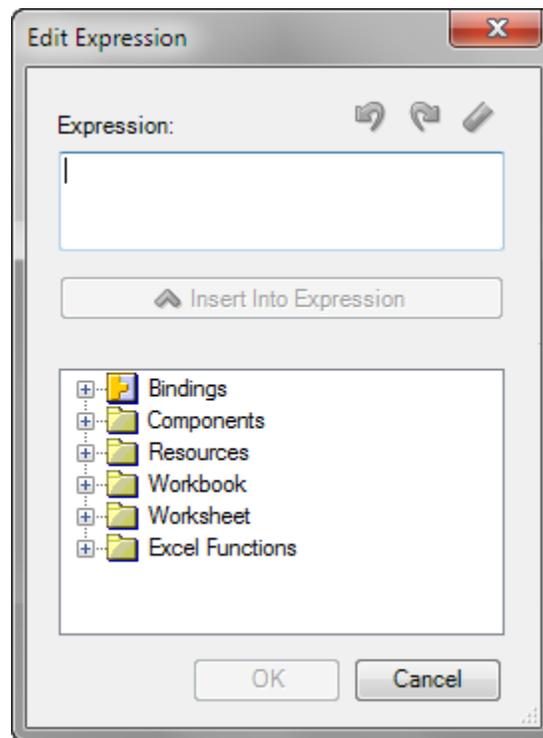
```
=IF("Closed" = "Closed", "MyReadOnlyStyle", "MyReadWriteStyle")
```

Excel evaluates the formula and, in this example, applies the `MyReadOnlyStyle` style.

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 10-6](#) displays an Edit Expression dialog launched from the property inspector window of a ribbon command.

Figure 10-6 Edit Expression Dialog



Before you begin:

It may be helpful to have an understanding of how to apply styles dynamically. See [Applying Styles Dynamically Using EL Expressions](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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 10-6](#), displays a hierarchical list of the Oracle ADF components, bindings, styles, resources, and Excel functions that you can reference in EL expressions. For information about the syntax of EL expressions that you enter in this dialog, see [ADF Desktop Integration EL Expressions](#).

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
 - An action set invokes a worksheet `DownSync` action
- EL expressions that evaluate to styles are not applied when:
 - A row-level action set invokes an ADF Table component `RowDownSync` action
 - The end user edits the format properties of a cell
 - An EL expression that evaluates to a style is not reevaluated when an 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 information about client-side logging, see [Generating Log Files for an Integrated Excel Workbook](#).

- When a user navigates between cells or during upload. ADF Desktop Integration does not evaluate or apply styles during these end user actions.

- 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 unused worksheet.

Using Labels in an Integrated Excel Workbook

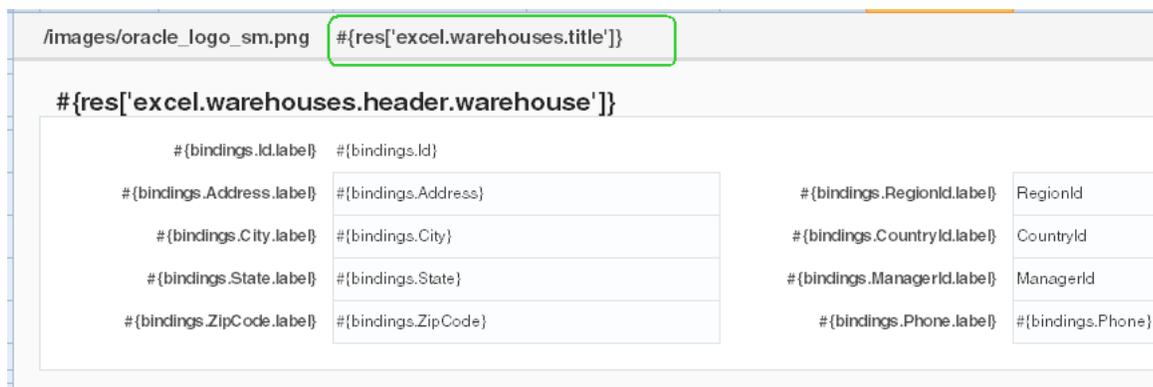
Use labels to provide end users with information about how they use the functionality in an integrated Excel workbook.

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.

Retrieving the Values of String Keys from a Resource Bundle

Figure 10-7 shows a portion of the design time view of the `EditWarehouseInventory-DT.xlsx` workbook in the Summit sample application for ADF Desktop Integration. It shows an ADF Label component that uses an EL expressions to retrieve the value of its `Label` property.

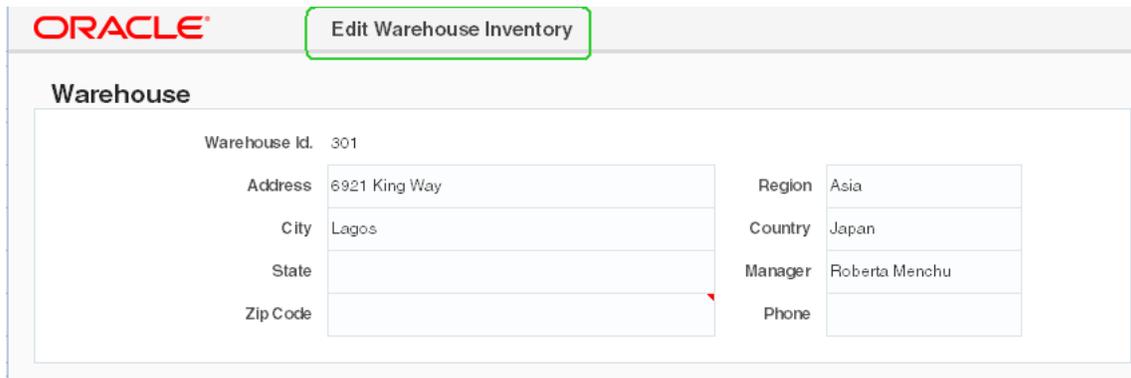
Figure 10-7 Design Time View of an ADF Label Component and an ADF Input Text Component with Label Property



At runtime, this EL expression resolves to a string key 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 [Using Resource Bundles in an Integrated Excel Workbook](#).

Figure 10-8 shows the corresponding runtime view of the ADF Label component illustrated in design mode in Figure 10-7.

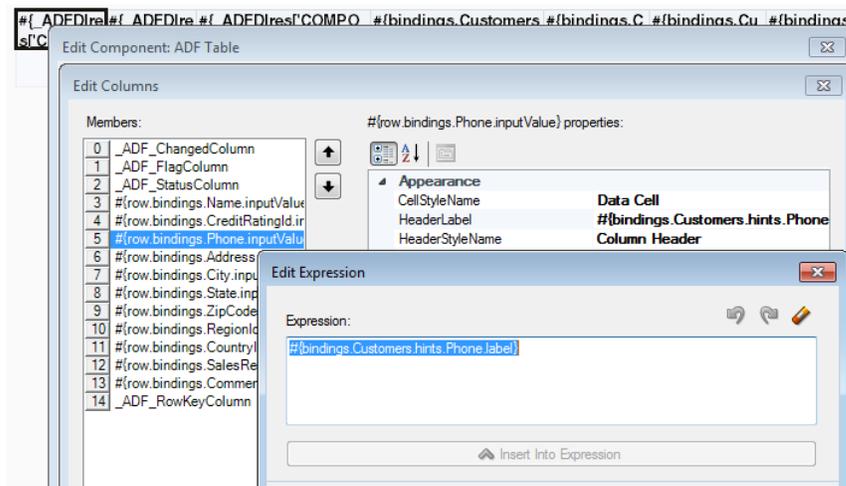
Figure 10-8 Runtime View of an ADF Label Component



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 10-9 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 10-9 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 *Defining Attribute Control Hints for Entity Objects* of *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 *Defining UI Hints for View Objects* of *Developing Fusion Web Applications with Oracle Application Development Framework*.

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 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-18](#).

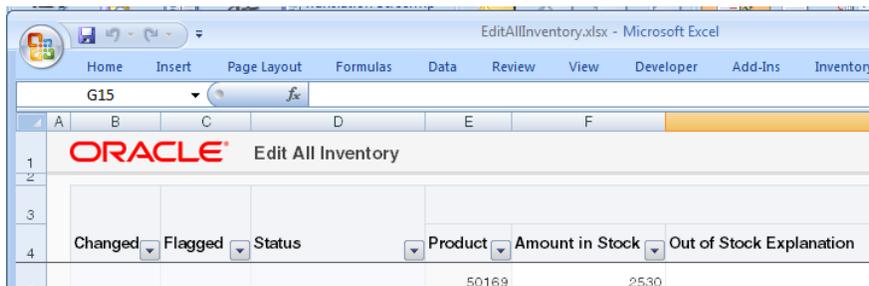
Branding Your Integrated Excel Workbook

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 [Configuring the Runtime Ribbon Tab](#) so that the end user can view this branding information by clicking a ribbon command that invokes the `ViewAboutDialog` workbook action at runtime. See [Table A-18](#).

ADF Desktop Integration also provides a style (`Branding Area`) to assist you in branding your integrated Excel workbooks. The ADF Desktop Integration sample application applies this style to the first row of each of its sample workbooks. Used with the ADF Image and ADF Output components, as demonstrated in [Figure 10-10](#), the style contributes to the consistent branding of the integrated Excel workbooks in the sample application.

Figure 10-10 Branding Area in Sample Workbook



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 [Using Resource Bundles in an Integrated Excel Workbook](#).

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 the brand of your integrated Excel workbook. See [Branding Your Integrated Excel Workbook](#).

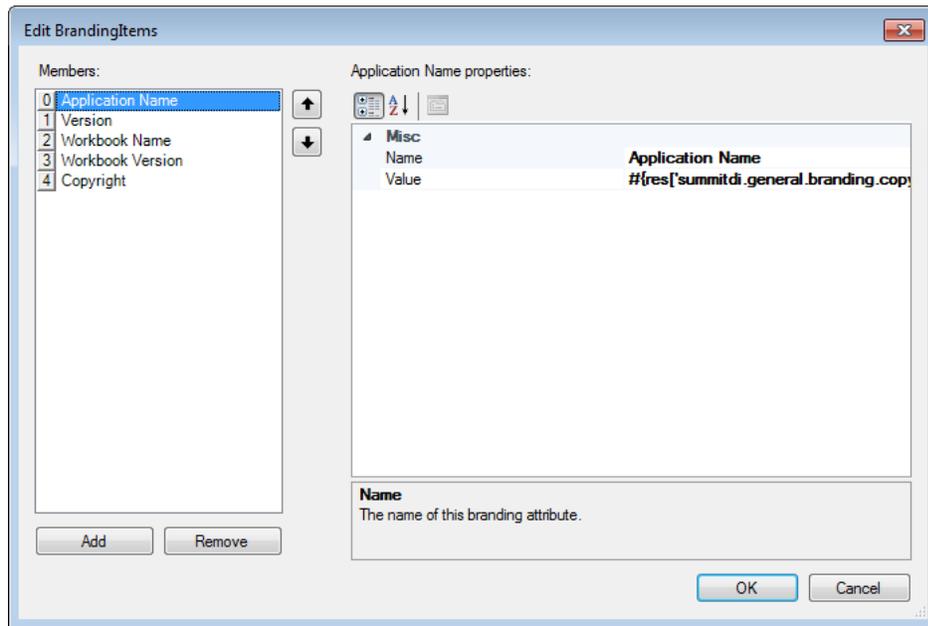
You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Configuring the Appearance of an Integrated Excel Workbook](#).

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 10-11](#) shows the design time view of branding items in the Summit sample application for ADF Desktop Integration.

Figure 10-11 Design Time View of Branding Items in the Summit Sample Application for ADF Desktop Integration



5. Click **OK**.



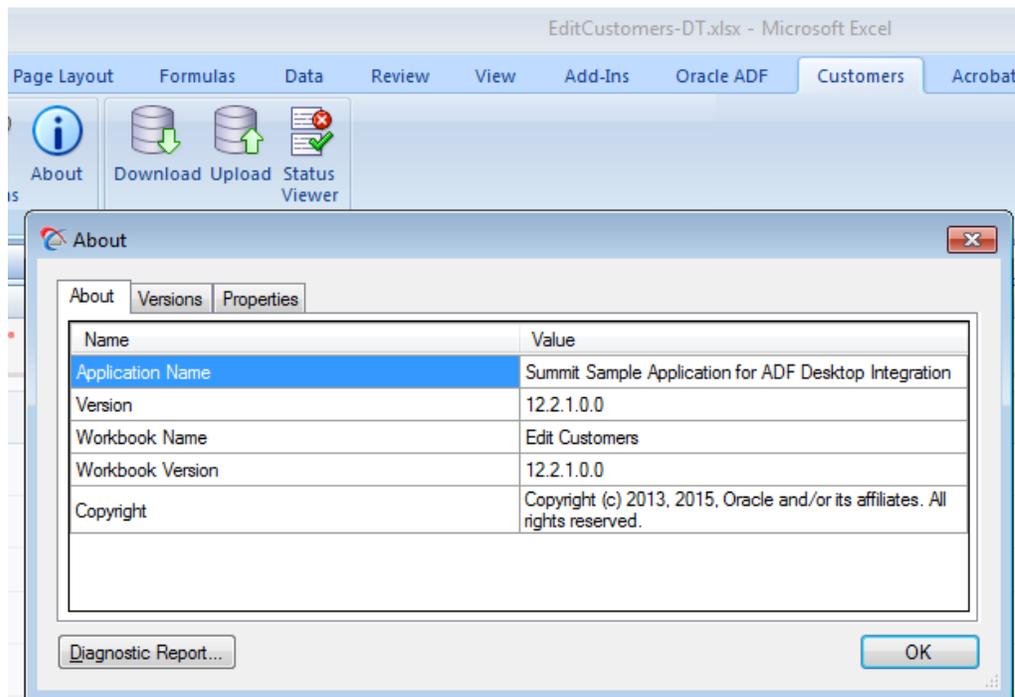
Tip:

You may also add your brand's image or logo to the integrated Excel spreadsheets. See [Inserting an ADF Image Component](#).

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 the About tab of the About dialog that the end user invokes using the **About** ribbon command of the runtime ribbon tab. You configure the runtime ribbon tab to appear, as described in [Configuring the Runtime Ribbon Tab](#). [Figure 10-12](#) shows the runtime view of branding items in an integrated Excel workbook.

Figure 10-12 Runtime View of Branding Items in the Summit Sample Application for ADF Desktop Integration



Note:

No About tab appears in the About dialog at runtime if you do not specify properties for the `BrandingItems` group of workbook properties.

Displaying Tooltips in ADF Desktop Integration Components

Use tooltips to display a hint or instruction text for ADF Desktop Integration components and table column headers. The tooltip appears in the Comment window of the cell that anchors the component or in the column header cell in the case of table column headers.

Tooltips can be defined as literal strings or EL expressions. You enter the literal string in the `Tooltip` property of the component or the column. You can also specify the EL expression (including a resource expression) as a value for the `Tooltip` property. At runtime, the EL expression resolves to the tooltip to display.

Note:

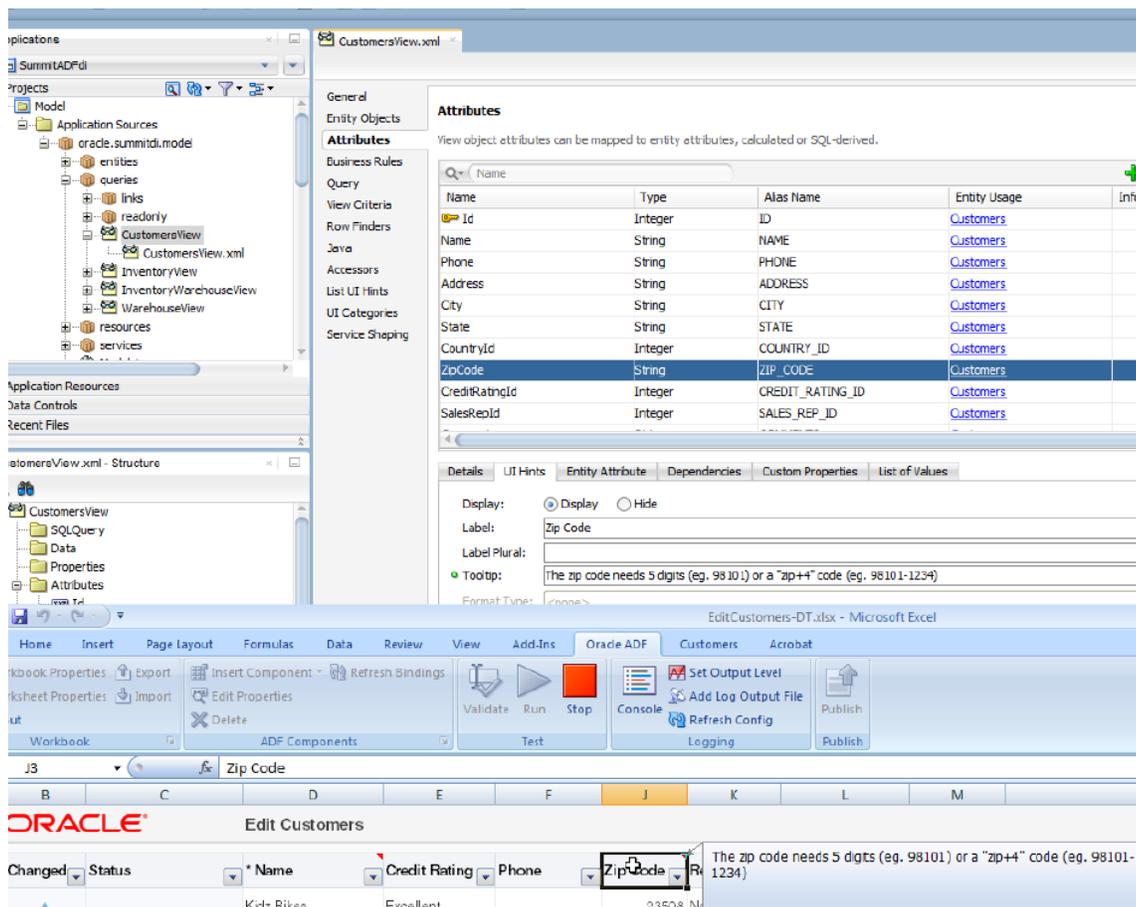
ADF Desktop Integration also supports `toolTip` attribute control hint in EL expressions. The support is similar to the `mandatory` control hint described in [Table B-3 of ADF Desktop Integration EL Expressions](#).

If you create a component from a binding element, the tooltip is automatically set to the model-driven tooltip, otherwise it is empty. Note that the tooltip is always initially empty for the ADF Label component. For table column headers, the default value that it renders is the value of the Fusion web application's attribute control `Tooltip` property, as shown in [Figure 10-13](#), if you created the ADF Table component from a tree binding. The Special columns (**Changed**, **Flagged**, **Status**) are an exception. By default, they do not render a tooltip.

Attribute control hints can be configured for view objects. Information about how to define a UI hint for a view object can be found in *Defining UI Hints for View Objects of Developing Fusion Web Applications with Oracle Application Development Framework*. For information about how to retrieve the value of an attribute control hint in an integrated Excel workbook, see [Retrieving the Values of Attribute Control Hints](#).

For information, see [How to Add a Tool Tip to an ADF Table Component](#) and [How to Add a Tool Tip to a Form-Type Component](#).

Figure 10-13 Attribute Control Hint Tooltip that Renders Tooltip in ADF Table Column Header



 **Note:**

In [Figure 10-13](#), notice the small red arrow at the top-right of the **Zip Code** column header cell in the `EditCustomers-DT.xlsx` workbook. It indicates that the header cell has a comment. Hover your mouse pointer over the cell to see the tooltip message.

How to Add a Tool Tip to an ADF Table Component

You configure the `Tooltip` property of the column in the ADF Table component that you want to render a tooltip.

Before you begin:

It may be helpful to have an understanding of tooltips. See [Displaying Tooltips in ADF Desktop Integration Components](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Configuring the Appearance of an Integrated Excel Workbook](#).

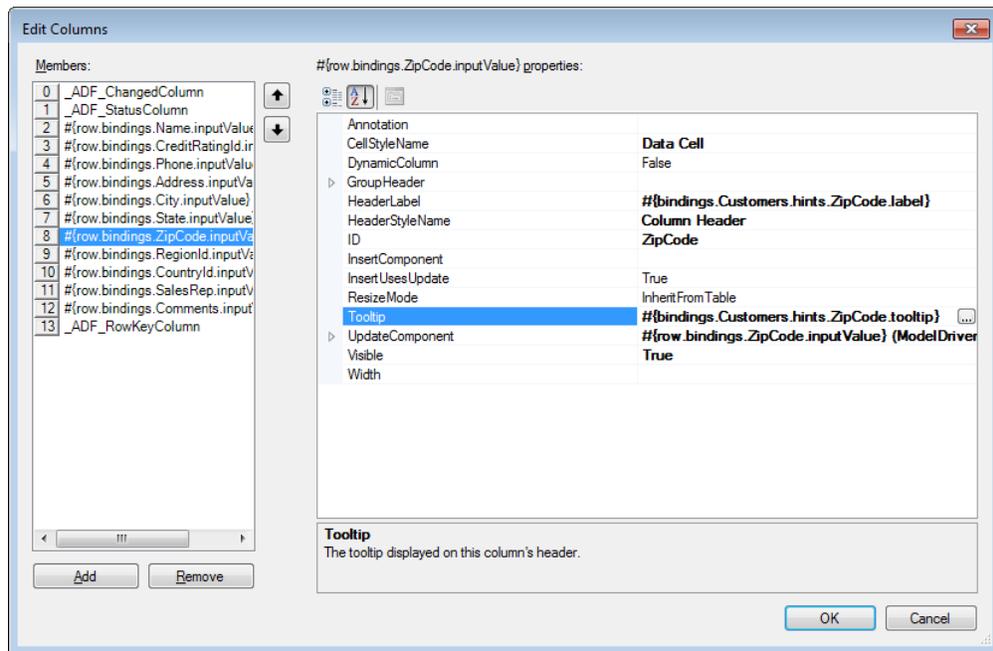
To add a tooltip to a table column header:

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. You can enter a literal string or an EL expression.

[Figure 10-14](#) shows the tooltip EL expression for the ADF Table column header in the Summit sample application's `EditCustomers-DT.xlsx` workbook that renders the runtime tooltip shown in [Figure 10-13](#).

Figure 10-14 Tooltip for ADF Table Column Header at Design-time



4. Click **OK**.

How to Add a Tool Tip to a Form-Type Component

You configure the `Tooltip` property of component that you want to render a tooltip.

Before you begin:

It may be helpful to have an understanding of tooltips. See [Displaying Tooltips in ADF Desktop Integration Components](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Configuring the Appearance of an Integrated Excel Workbook](#).

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.

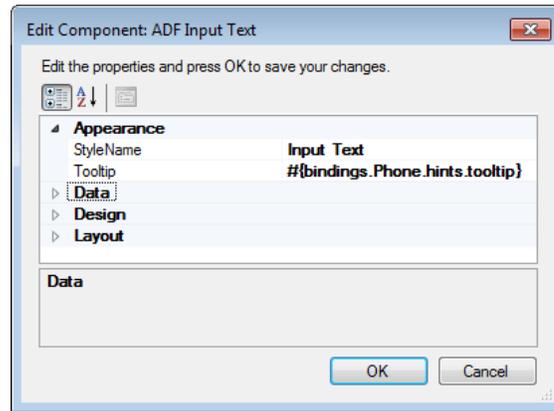
To insert a component to the worksheet, select the cell where you want to anchor the component. In the components palette or the bindings palette, select the form-type component or the binding, and click **Insert Component** or **Insert Binding**.

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 10-15](#) shows the `Tooltip` property of an Input Text component.

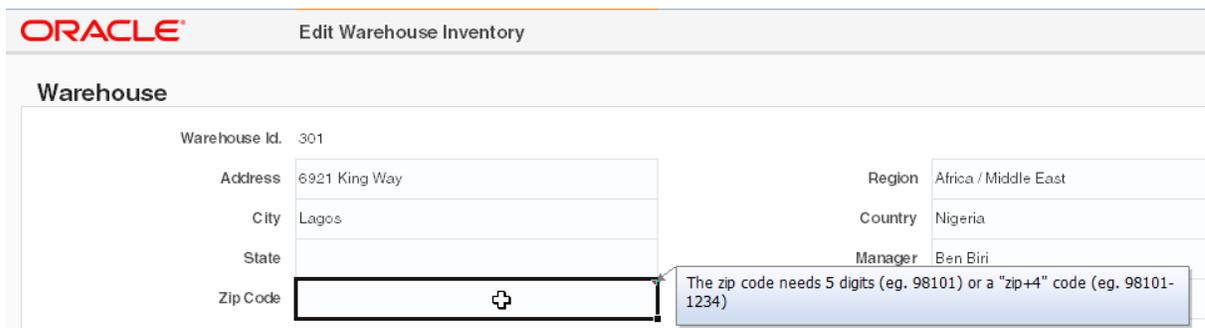
Figure 10-15 Tooltip Property of Input Text Component



4. Click **OK**.

Figure 10-16 shows the tooltip message 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 10-16 Tooltip Message of Input Text Component at Runtime



What You May Need to Know About Tooltips for Form-Type Components

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

Any Excel comment added manually at design time to a cell (or merged area) containing an ADF component is removed at runtime and replaced by the ADF component's tooltip. Similarly, any Excel comment added manually to an ADF component's cell during test mode is removed when the integrated Excel workbook returns to design-time mode. Excel comments added to cells with no ADF components, or to ADF components that do not support tooltips remain unchanged.

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 [What You May Need to Know About Tooltips for Table Columns](#)) and Worksheet Ribbon commands (see [How to Configure a Worksheet Ribbon Command for the Runtime Ribbon Tab](#)).

 **Note:**

- 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.

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.

If the `Tooltip` property of a column is set to a non-empty EL expression, the text that the EL expression evaluates to 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 information about tooltips, see [What You May Need to Know About Tooltips for Form-Type Components](#). You can also add tooltips to the headers of special columns of the table components (see [Special Columns in the ADF Table Component](#)) and the dynamic columns (see [Adding a Dynamic Column to Your ADF Table Component](#))

Using Worksheet Protection

By default, end users can edit the values of locked cells and ADF Desktop Integration components that have implied read-only behavior, such as ADF Label, ADF Output Text and ADF Table component's header rows, at runtime.

While uploading data, ADF Desktop Integration ignores these changes and overwrites them when it next refreshes the component.

Various ADF Desktop Integration components, (for example, ADF InputText component) and subcomponents (for example, `ModelDrivenColumnComponent` subcomponent) include a `ReadOnly` property.

To prevent editing of locked cells at runtime, enable ADF Desktop Integration worksheet protection. Optionally, you can also provide a password to prevent the end user from turning off 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.

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. See [Using Worksheet Protection](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Configuring the Appearance of an Integrated Excel Workbook](#).

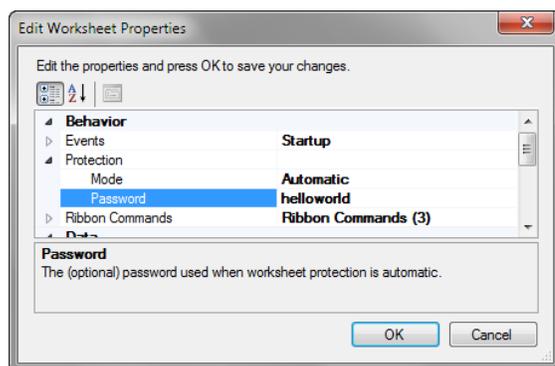
To enable Worksheet Protection:

1. Open the integrated Excel workbook.
2. In the Workbook group of the Oracle ADF tab, click **Worksheet 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 10-17](#) shows the design time view of worksheet protection in the Summit sample application for ADF Desktop Integration.

Figure 10-17 Design Time View of Worksheet Protection in the Summit Sample Application for ADF Desktop Integration

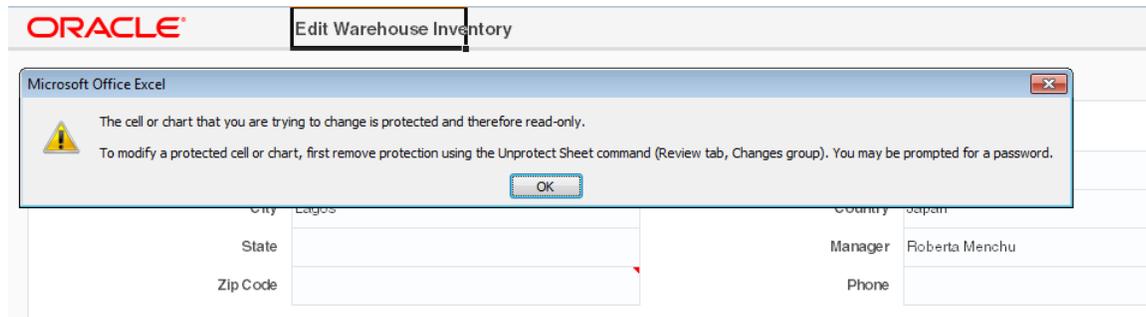


4. Click **OK**.

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 10-18](#).

Figure 10-18 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. ADF Desktop Integration does not alter the **Locked** property of cells outside the bounds of ADF Desktop Integration components.

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 may or may not be read-only depending on their configuration. At runtime, each component's read-only behavior is evaluated and the corresponding cell's **Locked** property is set to the appropriate value.

What You May Need to Know About Worksheet Protection

Worksheet protection is not enabled by default. You 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.

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.

Using ADF Desktop Integration EL-based Properties with Custom Attribute Properties

You can use custom attribute properties defined in view objects on the server with ADF Desktop Integration EL-based properties of the integrated Excel workbook. By default, ADF Desktop Integration EL evaluation does not support custom attribute properties defined on the server.

To enable the support, you must set the `Worksheet.CustomAttributePropertiesEnabled` property to `True`.

After enabling the support, you can reference custom attribute property names in EL-based property values.

How to Enable Custom Attribute Properties in Integrated Excel Workbook

Before you enable custom attribute properties, configure one (or more) custom attribute properties in your Fusion web application. For information about how to define a UI hint for a view object, see *Defining UI Hints for View Objects in Developing Fusion Web Applications with Oracle Application Development Framework*.

To enable custom attribute properties in integrated Excel workbook:

1. Open the integrated Excel Workbook.
2. In the Oracle ADF tab, click **Worksheet Properties**.
3. Set `CustomAttributePropertiesEnabled` to `True`.
4. Click **OK**.

After setting `CustomAttributePropertiesEnabled` to `True`, you can reference custom attribute properties within EL expressions using one of the following formats:

- For attribute hint, use this format: `"#{bindings.{attr id}.hints.{custom property}}"`
- For tree attribute hint, use this format: `"#{bindings.{tree id}.hints.{attr id}.{custom property}}"`
- For dynamic column hint, use this format: `"#{bindings.{tree id}.[{node id}].hints.*.{custom property}}"`

In the following examples, `diCellStyle` is a custom attribute property that the developer added to the relevant model attribute:

- static column example:
`#{bindings.EmpCompView1.hints.EmployeeId.diCellStyle}`
- dynamic column example: `#{bindings.EmpCompDeclSqlView1.[model.EmpCompDeclSqlView].hints.*.diCellStyle}`

What Happens at Runtime: CustomAttributePropertiesEnabled is Set to True

When a worksheet's `CustomAttributePropertiesEnabled` is set to `True`, ADF Desktop Integration EL-based properties start evaluating custom attribute property values returned from the server.



Tip:

For best performance, whenever possible, ensure that the custom property value should be a literal value (for example: an Excel style name).

What You May Need to Know About the CustomAttributePropertiesEnabled Property

Note the following points about the `CustomAttributePropertiesEnabled` property and its behavior:

- Custom property names are case-sensitive.
- If the custom property value is itself an EL expression (rather than a literal value), the returned property value gets re-evaluated as EL.



Note:

If the custom property value is an EL expression evaluated for a column's `ReadOnly` property, see [Evaluating EL Expressions for ReadOnly Properties](#) for information about evaluating this EL expression.

- EL re-evaluation does not apply to standard attribute hint values.
- When `CustomAttributePropertiesEnabled` is `True`, configuration validation does not report a validation error for custom property names in EL.

When `CustomAttributePropertiesEnabled` is `False`, configuration validation does report a validation error for custom property names in EL.

- If a custom property name matches a reserved hint name (for example, `label`), the custom property is ignored.

11

Internationalizing Your Integrated Excel Workbook

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:

- [About Internationalizing Your Integrated Excel Workbook](#)
- [Using Resource Bundles in an Integrated Excel Workbook](#)
- [Localization in ADF Desktop Integration](#)

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 runtime.

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. See [Working with Styles](#).

For information about internationalizing Fusion web applications, see Internationalizing and Localizing Pages in *Developing Web User Interfaces with Oracle ADF Faces*.

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, you can configure your integrated Excel workbook for a specific local language or culture by providing translations of the user-visible strings that appear to end users at runtime. See [Localization in ADF Desktop Integration](#).

Figure 11-1 shows an example of an integrated Excel workbook configured for the Japanese language.

Figure 11-1 Integrated Excel Workbook in Japanese

	A	B	C	D	E	F	G	H	I	L	M
1											
2		変更済	フラグ付き	ステータス	従業員番号	従業員名	仕事	マネージャー	日付を雇う	部署番号	キー
3					7369	スミス	書記	7902	29572	20	
4					7499	艾伦	推銷員	7698	29637	30	
5					7521	歓迎	推銷員	7698	29639	30	
6					7566	ばかりば	良い家	7839	29769	20	
7					7654	ばかりば	推銷員	7698	29857	30	

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. The following sections describe other functionality 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. See [Securing Your Integrated Excel Workbook](#).
- **Validating integrated Excel workbook:** You can configure server-side and client-side data entry validation for the Fusion web application and the integrated Excel workbook. See [Adding Validation to 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. See [Deploying Your Integrated Excel Workbook](#).

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. You can use JDeveloper to create and manage resource bundles in your Fusion web application.

You can register multiple resource bundles with an integrated Excel workbook. At runtime, ADF Desktop Integration downloads only those string key values that the integrated Excel workbook uses from registered resource bundles during workbook initialization. Assume, for example, that you register resource bundle A with an integrated Excel workbook that references three string key values from resource bundle A. During workbook initialization, ADF Desktop Integration downloads the three string key values that the workbook references from resource bundle A.

 **Note:**

If you register more than 20 resource bundles with an integrated Excel workbook, ADF Desktop Integration logs a warning message. For information about client-side logging, see [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 [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. ADF Desktop Integration uses the value `_ADFDirres` to uniquely identify this resource bundle. Many EL expressions reference string values in this resource bundle.

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:

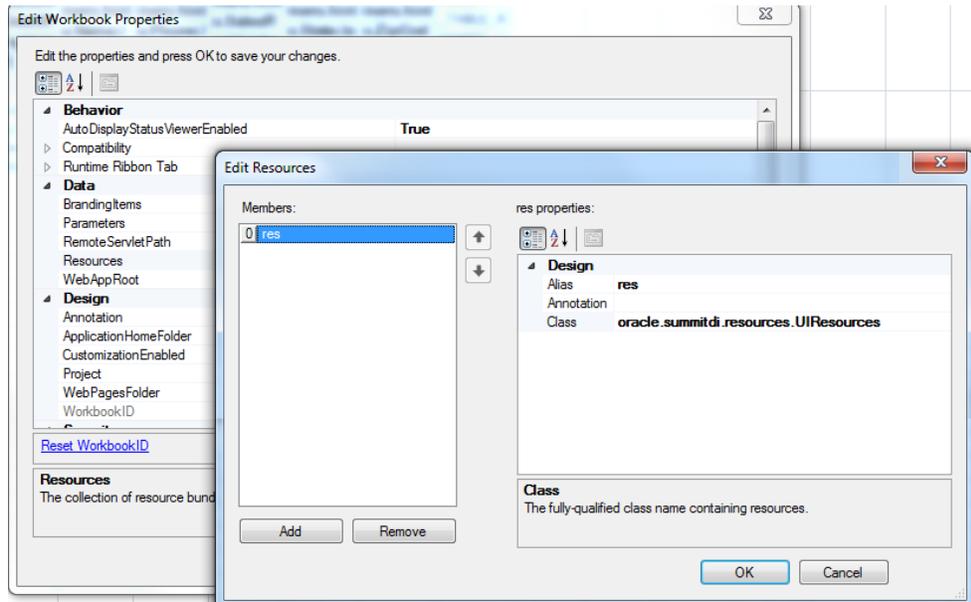
It may be helpful to have an understanding of how to use resource bundles. See [Using Resource Bundles in an Integrated Excel Workbook](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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 11-2](#).

Figure 11-2 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-19](#).



Tip:

While registering a resource bundle class, do not include the file extension.

What You May Need to Know About Resource Bundles

See the following sections for additional information about resource bundles in an integrated Excel workbook.

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 information about resource bundles, see *Manually Defining Resource Bundles and Locales* in *Developing Web User Interfaces with Oracle ADF Faces*.

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 information about the `ClearAllData` workbook action, see [Table A-18](#).

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}
```

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 11-1](#) presents a summary of elements involved and their role in translation:

Table 11-1 Summary of Localization

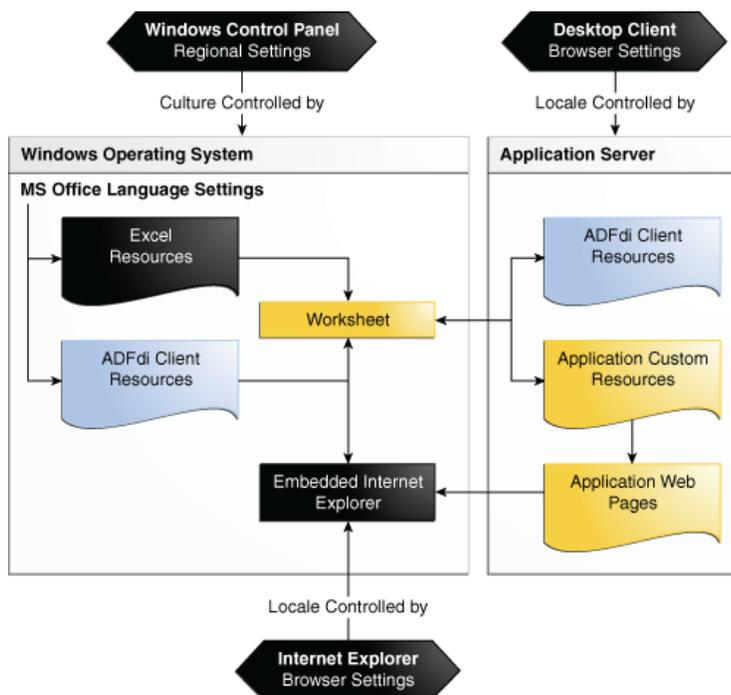
Area subject to localization	Determination of language to use
Microsoft operating system	Operating system language settings. You can choose the language through Regional Settings on Control Panel.
Microsoft Office	Microsoft Office language settings
Web pages displayed in ADF Desktop Integration Dialog 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

Table 11-1 (Cont.) Summary of Localization

Area subject to localization	Determination of language to use
ADF Desktop Integration custom resource bundles	Microsoft Internet Explorer language preferences

Figure 11-3 illustrates how various elements involved in a Fusion web application play their role in translation.

Figure 11-3 Localization in ADF Desktop Integration



For 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>

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.

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 11-1](#) shows a sample implementation of a user preference handler.

Note:

The handler class must have a constructor with no arguments, or uses the default Java constructor.

Example 11-1 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();
    }
}
```

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. See [Localization in ADF Desktop Integration](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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 11-2](#).

Table 11-2 Configuring Locale User Preference

Property	Value
Name	Enter the name of the initialization parameter as follows <code>UserPreferences.Handler</code>

Table 11-2 (Cont.) Configuring Locale User Preference

Property	Value
Value	Complete path of the handler class.

3. Save the `web.xml` file.
4. Rebuild and restart your Fusion web application.

Example 11-2 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>
```

[Example 11-2](#) shows the `web.xml` file with `UserPreferences.Handler`.

In [Example 11-2](#), `myCompany.XYZ.CustomUserPrefsHandler` is the complete path of the handler class.

12

Securing Your Integrated Excel Workbook

Describes security related features in ADF Desktop Integration. This chapter includes the following sections:

- [About Security In Your Integrated Excel Workbook](#)
- [Authenticating the Excel Workbook User](#)
- [Checking the Integrity of an Integrated Excel Workbook's Metadata](#)
- [What You May Need to Know About Securing an Integrated Excel Workbook](#)
- [Authorizing the Excel Workbook User](#)

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 *Enabling ADF Security in a Fusion Web Application in Developing Fusion Web Applications with Oracle Application Development Framework*.

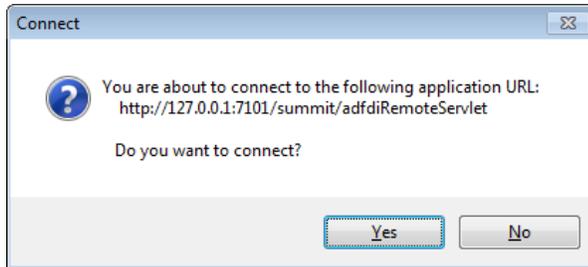
When you open the integrated Excel workbook, ADF Desktop Integration detects if the Fusion web application that the workbook runs against is a secure application and enforces authentication automatically. For authenticated web applications, the end user will always be prompted for credentials, even though 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 some form of Integrated Windows Authentication is used, such as Kerberos or NTLM). For information about Microsoft Kerberos, see <http://msdn.microsoft.com/en-us/library/aa378747%28v=vs.85%29.aspx>.

Integrated Excel Workbook Security Use Cases and Examples

When you open the integrated Excel workbook of a secure Fusion web application, a connection confirmation dialog appears and prompts you to connect to the Fusion web

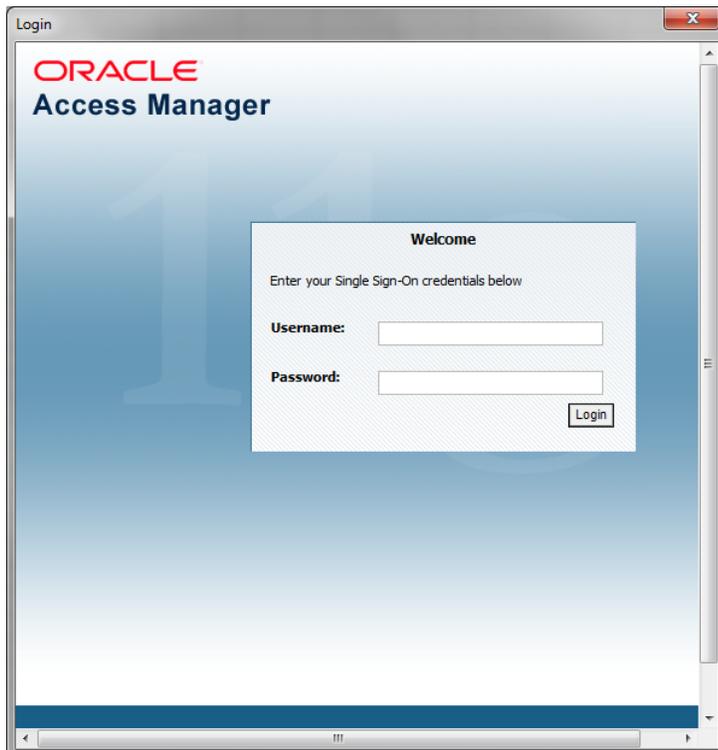
application, as shown in [Figure 12-1](#). Note that the connection confirmation dialog also appears when the Fusion web application is not secure.

Figure 12-1 Dialog to Verify Connection



If you click **Yes** to connect, another dialog appears that prompts you to enter user credentials. The dialog that appears depends on how the Fusion web application is configured to enforce authentication. [Figure 12-2](#), for example, shows the dialog that appears when the Fusion web application enforces form-based login using Oracle Access Management.

Figure 12-2 Form-Based Login Dialog



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. The following sections describe other functionality that you can use:

- **Validating integrated Excel workbook:** You can configure server-side and client-side data entry validation for the Fusion web application and the integrated Excel workbook. See [Adding Validation to Your Integrated Excel Workbook](#) .
- **Testing integrated Excel workbook:** Before publishing and deploying your integrated Excel workbook, you must test it. See [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. See [Deploying Your Integrated Excel Workbook](#) .

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.

What Happens at Runtime: How the Login Method Is Invoked

A modal dialog appears that contains a web browser control after the login method is invoked. The 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 dialog shown in [Figure 12-3](#). If the end-user successfully logs in, a new session between the integrated Excel workbook and the Fusion web application is created.

Figure 12-3 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.



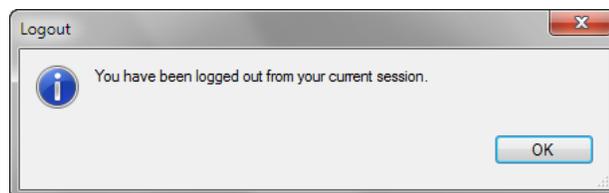
Note:

If the `Login` method is invoked when a session has already been established, it first invokes the `Logout` action internally to terminate that session.

What Happens at Runtime: How the Web Application Session is Terminated

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 12-4 Dialog That Appears When a User Logs Out



After logging out, the end user may continue to work with data in the spreadsheet. When the end user next attempts to interact with the server (for example, invoke an Upload action), the end user will be prompted to log in again.

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.

Checking the Integrity of an Integrated Excel Workbook's Metadata

ADF Desktop Integration verifies 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 [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 information about the workbook properties discussed here, see [Table A-19](#).

Note:

Tamper-check is not performed for customization-enabled workbooks.

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:

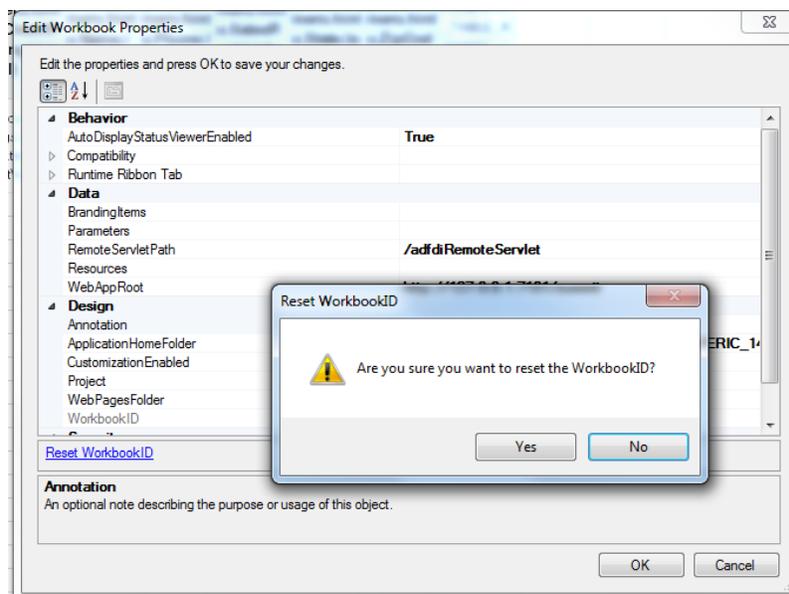
It may be helpful to have an understanding of how to verify the integrity of integrated Excel workbook's metadata. See [Checking the Integrity of an Integrated Excel Workbook's Metadata](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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 Reset WorkbookID dialog that appears., as shown in [Figure 12-5](#).

Figure 12-5 Reset Workbook ID Dialog



5. Click **OK**.

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.

What You May Need to Know About Securing an Integrated Excel Workbook

Describes additional steps to secure data in Excel workbooks that you have integrated 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 information about Excel security features, see Excel's documentation.

- Integrated Excel workbooks can be configured to cache data, as described in [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 connection confirmation 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 [What You May Need to Know About Configuring Security in a Fusion Web Application](#) .

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.

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. During early development, 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 while you fine tune your roles and resource associations.

For information about authorization, roles, and resource grants, see *Enabling ADF Security in a Fusion Web Application* in *Developing Fusion Web Applications with Oracle Application Development Framework*.

 **Note:**

ADF Desktop Integration only enforces authorization for resource grants that have the Web Page (page definition) resource type. Other resource types are not supported.

You can configure resources and grants from the Resource Grants page of the overview editor for the `jazn-data.xml` file. For information, see *Defining ADF Security Policies* in *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 web application.
```

What You May Need to Know About ADF Desktop Integration-Disabled Worksheet

The following limitations apply to an ADF Desktop Integration-disabled worksheet:

- All worksheet-level ribbon commands and worksheet-level events are disabled.
- If the authorization failure occurs during worksheet initialization, no form labels or table column headers are drawn on the worksheet.
- If the authorization failure occurs for an initialized worksheet, 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.

13

Adding Validation to Your Integrated Excel Workbook

Describes how to provide server-side and data entry validation for your integrated Excel workbook, how to report errors such as validation failures and data conflict, and how to configure error reports using a custom error handler.

This chapter includes the following sections:

- [About Adding Validation to an Integrated Excel Workbook](#)
- [Using the Status Viewer to Report Error Messages to End Users](#)
- [Providing Data Entry Validation for an Integrated Excel Workbook](#)
- [Providing Server-Side Validation for an Integrated Excel Workbook](#)
- [Providing a Row-by-Row Status on an ADF Table Component](#)
- [Adding Detail to Error Messages in an Integrated Excel Workbook](#)
- [Handling Data Conflicts When Uploading Data from a Workbook](#)

About Adding Validation to an Integrated Excel Workbook

You configure server-side and data entry validation for the Fusion web application and the integrated Excel workbook to make use of the validation options offered by the ADF Model layer, ADF Desktop Integration, 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 modification in an integrated Excel workbook conflicts 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.

Integrated Excel Workbook Validation Use Cases and Examples

Validation rules protect the server by preventing the upload of invalid data. ADF Desktop Integration provides both data entry validation and server-side validation capabilities. [Figure 13-1](#) shows an example of server-side validation from the Summit sample application's `EditCustomers-DT.xlsx` workbook where an invalid zip code (12345x) fails an entity validation rule. This failure appears in the Status Viewer entry for the row that contains the invalid zip code.

Figure 13-1 Status Viewer Displaying Entity Validation Rule Failure

Changed	Status	* Name	Address	City	State	Zip Code	Country
▲	Update failed	Zebra's Bicycles	3910 Colley Ave	Norfolk	VA	12345x	USA
	Row updated successfully	Superior Bicycle	538 Superior Ave E	Cleveland	OH	44114	USA
	Row updated successfully	Bicycle World	5300 SW 17th St	Topeka	KS	66604	USA
	Row updated successfully	Schindler's Sports	4479 Forest Park Ave	St Louis	MO	63108	USA
	Row updated successfully	Barry's Basketball	56 E Superior St	Chicago	IL	60611	USA
	Row updated successfully	Gavin Sporting Goods	1935 SE Hawthorne Blvd	Portland	OR	97214	USA
		Hot Stuff	25613 Dollar St	Hayward	CA	94544	USA
		Acme Sporting Goods	770 4th Ave	San Diego	CA	92101	USA
		MoreAndMoreStuffz	3501 McKinney Ave	Dallas	TX	75204	USA

Figure 13-2 shows an example of a data entry validation failure from the same workbook where no value appears in a cell that requires a value.

Figure 13-2 Data Entry Validation Message

Changed	Status	* Name	Credit Rating	Phone	Address	City	State	Zip Code	Region
		Zebra's Bicycles	Fair		3910 Colley Ave	Norfolk	VA	23508	North America
▲	Invalid		Good		538 Superior Ave E	Cleveland	OH	44114	North America
			Good		5300 SW 17th St	Topeka	KS	66604	North America
			Good		4479 Forest Park Ave	St Louis	MO	63108	North America
		Barry's Basketball	Good		56 E Superior St	Chicago	IL	60611	North America
		Gavin Sporting Goods	Good		1935 SE Hawthorne Blvd	Portland	OR	97214	North America

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. The following sections describe other functionality that you can use:

- **Testing integrated Excel workbook:** Before publishing and deploying your integrated Excel workbook, you must test it. See [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. See [Deploying Your Integrated Excel Workbook](#).
- **Trigger Server-side Validation in an ADF Table Component:** ADF Table components can be configured to automatically refresh a row where an end user

edits a value. Enabling this functionality is one way of triggering server-side validation. See [Configuring Automatic Row Refresh in an ADF Table Component](#).

Using the Status Viewer to Report Error Messages to End Users

The Status Viewer displays information to end users in Excel's task pane. End users can use the information that appears to review and correct errors at the same time.

Information that the Status Viewer always displays includes the worksheet-level status of the current integrated worksheet. In addition, if the worksheet includes an ADF Table component and the currently selected cell is a row in the ADF Table component, the Status Viewer displays the status of the row.

Information about the result of action set invocation also appears in the Status Viewer. For example, an end user enters a value that violates a declarative validation rule in the Fusion web application's ADF Model layer. When the end user attempts to upload the change, a failure is reported for the failed row. In this scenario, the Status Viewer appears and displays a message about the validation failure.

[Figure 13-3](#) shows the Status Viewer that appears in the `EditCustomers-DT.xlsx` workbook when an end user enters a zip code (12345x) that fails an entity validation rule defined in the Fusion web application's ADF Model layer. Selecting a cell anywhere in the table row that contains the validation failure causes the validation failure message to appear in the Status Viewer. The worksheet-level status that appears in the Status Viewer in [Figure 13-3](#) indicates that the most recent action set from this worksheet completed successfully.

Figure 13-3 Status Viewer

The screenshot shows an Excel spreadsheet titled "Edit Customers" with a table of customer records. The "City" cell for the first row is selected. The Status Viewer pane on the right displays the following information:

- Messages for this worksheet are listed below:**
 - ✓ No error.
- Messages for this table row are listed below:**
 - ✗ The zip code needs 5 digits (eg. 98101) or a "zip+4" code (eg. 98101-1234)

Changed	Status	* Name	Address	City	State	Zip Code	Country
	Update failed	Zebra's Bicycles	3910 Colley Ave	Norfolk	VA	12345x	USA
	Row updated successfully	Superior Bicycle	538 Superior Ave E	Cleveland	OH	44114	USA
	Row updated successfully	Bicycle World	5300 SW 17th St	Topeka	KS	66604	USA
	Row updated successfully	Schindler's Sports	4479 Forest Park Ave	St Louis	MO	63108	USA
	Row updated successfully	Barry's Basketball	56 E Superior St	Chicago	IL	60611	USA
	Row updated successfully	Gavin Sporting Goods	1935 SE Hawthorne Blvd	Portland	OR	97214	USA
		Hot Stuff	25613 Dollar St	Hayward	CA	94544	USA
		Acme Sporting Goods	770 4th Ave	San Diego	CA	92101	USA
		MoreAndMoreStuffz	3501 McKinney Ave	Dallas	TX	75204	USA

Integrated Excel workbooks that you create using this release of ADF Desktop Integration display the **Status Viewer** ribbon command in the Excel ribbon by default, as shown in [Figure 13-4](#).

Figure 13-4 Status Viewer Ribbon Command in Excel Ribbon

End users click the **Status Viewer** ribbon command to display or hide the Status Viewer in Excel's task pane. By default, the Status Viewer appears automatically when integrated Excel workbooks encounter errors at runtime. You can configure this behavior for integrated Excel workbooks created using earlier releases so that they automatically display the Status Viewer when errors occur. See [How to Manage the Automatic Display of the Status Viewer](#).

You add the Status Viewer ribbon command to the Excel ribbon by adding the `ToggleStatusViewer` workbook action, as described in [How to Define a Workbook Ribbon Command for the Runtime Ribbon Tab](#). For information about workbook actions, see [Workbook Actions and Properties](#).

Although you add the Status Viewer ribbon command the Excel ribbon as a workbook command, the Status Viewer is worksheet specific and displays information for the integrated Excel worksheet that is in focus. If your end users navigate to a non-integrated worksheet and click the Status Viewer ribbon command, a message appears that tells the end user the Status Viewer cannot be used in that worksheet.

How to Manage the Automatic Display of the Status Viewer

You set the value of the `AutoDisplayStatusViewerEnabled` workbook property to `True` or `False` to manage the automatic display of the Status Viewer in Excel's task pane.

Before you begin:

It may be helpful to have an understanding of the Status Viewer provided by ADF Desktop Integration. See [Using the Status Viewer to Report Error Messages to End Users](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Adding Validation to an Integrated Excel Workbook](#).

To manage the automatic display of the Status Viewer:

1. Open the integrated Excel workbook.
2. From the Excel Ribbon, in the Oracle ADF tab, click **Workbook Properties**.
3. In the Edit Workbook Properties dialog, expand Behavior and set the `AutoDisplayStatusViewerEnabled` property appropriately:
 - **True:** Status Viewer automatically appears when an error occurs.
 - **False:** End user must click the **Status Viewer** ribbon command in the Excel ribbon to display the Status Viewer.
4. Click **OK**.

Providing Data Entry Validation for an Integrated Excel Workbook

ADF Desktop Integration automatically performs basic data entry validation after end users modify cells bound to ADF components.

Basic data entry validation includes verifying the expected data type (for example, user entered a number for a numerical attribute) and that required fields are not empty. ADF Desktop Integration performs this validation as soon as end users leave the cell.

Metadata from the ADF Model layer is used to perform basic data entry validation. No additional workbook configuration is needed. You can disable this validation using the `Compatibility.DataEntryValidationEnabled` workbook properties described in [How to Enable or Disable ADF Desktop Integration Data Entry Validation](#). ADF Desktop Integration enables basic data validation by default.

ADF Desktop Integration performs additional validation during upload. See [Providing Server-Side Validation for an Integrated Excel Workbook](#).

One other validation that ADF Desktop Integration performs is to reject Excel cell error values that the integrated Excel workbook sends to the Fusion web application. An example of an Excel cell error is the `#DIV/0!` error that occurs when a number is divided either by zero (0) or by a cell that contains no value. Excel cell error values return large negative numbers. The `#DIV/0!` error, for example, returns `-2146826281`. ADF Desktop Integration rejects these values because they are unlikely to be appropriate for upload and, in the absence of ADF Model layer data validation, can be committed to the Fusion web application's database.

An ADF Table component displays an `Update failed` message in its Status column when an ADF Table component performs an `Upload`, `RowUpSync`, or `RowUpSyncNoFail` action on a row with a cell containing an Excel cell error value. Similarly, a worksheet's `UpSync` action fails to synchronize pending changes from cells that contains Excel cell error values.

Apart from the `#DIV/0!` error, other error cell values that ADF Desktop Integration rejects include `#NULL!`, `#REF!`, `#N/A`, `#NAME?`, `#NUM!`, and `#VALUE!` For information about Excel cell error values, see Microsoft's documentation.

You can enable an integrated Excel workbook to upload Excel cell error values to the Fusion web application by having ADF Desktop Integration accept Excel cell error values. You do this by setting the `WorkbookCompatibility.RejectExcelErrorsEnabled` property to `False`. The default value is `True` for workbooks created using this release of ADF Desktop Integration. See [How to Upload Excel Cell Errors to the Fusion Web Application](#).

Providing Data Entry Validation Using ADF Desktop Integration

ADF Desktop Integration performs data entry validation to verify that:

- Mandatory fields contain a value. ADF Desktop Integration reports a validation failure if an Excel cell that contains an ADF component which requires a mandatory value (for example, ADF Input Text component) is blank.

- The correct data type is entered. If, for example, you enter a string ("Bob") in an input field where the required data type is a date or a number, ADF Desktop Integration reports a validation failure.

ADF Desktop Integration performs the above types of validation without making a request to the Fusion web application.

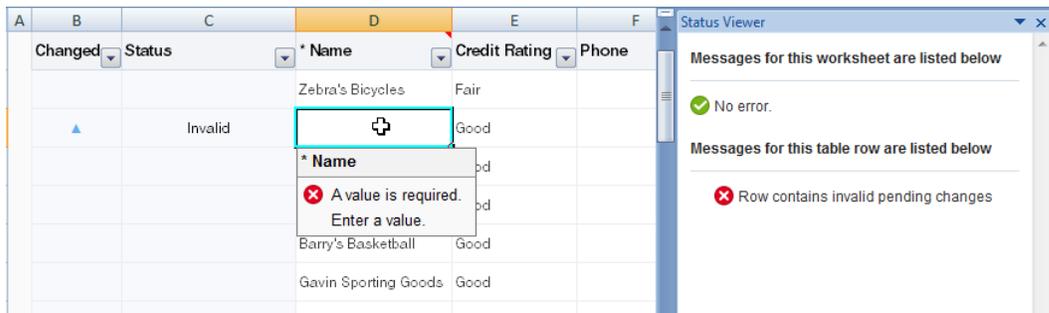
ADF Desktop Integration performs data entry validation on the ADF Input Text, ADF Input Date and ADF Table components. It does not perform data entry validation on read-only cells, label or headers cells, locked cells or cells in the columns described in [Special Columns in the ADF Table Component](#) . It also does not perform data entry validation on cells in the ADF Read-only Table or ADF List of Values components.

Data entry validation performed by ADF Desktop Integration identifies failures upon editing a single cell or multiple cells (simultaneously). Examples of edits that span multiple cells include a selection of a column in an ADF Table component or an end user pasting one or more rows of data into in an ADF Table component. ADF Desktop Integration performs data entry validation only after an end user edits a single cell or multiple cells and leaves edit mode for the cell(s). If a validation failure occurs, ADF Desktop Integration applies a red border to the cell that failed validation after the end user exits edit mode by pressing Enter, tabbing away, or selecting a different cell.

Once ADF Desktop Integration reports a validation failure, the end user can view a non-modal popup message by selecting the cell without entering edit mode. This non-modal popup message describes the validation failure and may suggest an action to resolve the validation failure. It remains visible as long as the end user selects the cell and the validation failure is present. No non-modal popup message appears if end users select multiple cells with validation failures. In ADF Table component cells, a message also appears in the Status column to indicate that a row contains at least one cell with a validation failure. ADF Table component actions such as `Download` and `Upload` clear this latter message. The ADF Table component's `Download` action also clears the red border around cells that contain validation failures. For information about the Status column, see [Special Columns in the ADF Table Component](#).

The Status Viewer displays a message for a row with cells that contain validation failures. It displays this message ("Row contains invalid pending changes") until the end user resolves the validation failures. In addition to data entry validation errors, the Status Viewer might display other messages, such as failure messages from the last Upload operation. See [Using the Status Viewer to Report Error Messages to End Users](#).

[Figure 13-1](#) shows a cell with a data entry validation failure in the Summit sample application's `EditCustomers-DT.xlsx` workbook. The end user has not entered a value in an ADF Table component cell that requires a value. The non-modal popup message and the Status Viewer both display information about this failure.

Figure 13-5 ADF Desktop Integration Data Entry Validation

ADF Desktop Integration applies a red border to a cell that fails validation until the end user resolves the issue that causes the validation failure. If you or an end user set a cell border to red, ADF Desktop Integration does not consider the cell invalid until a validation error occurs (missing mandatory value, for example). To prevent visual confusion for end users, avoid the use of red borders on cells in your integrated Excel workbook so that its use is reserved to ADF Desktop Integration reporting validation failures.

Validation failures do not prevent end users from continuing to edit or enter data in the integrated Excel workbook nor does the presence of data entry validation failures prevent the upload of data from the integrated Excel workbook.

Note:

If an end user modifies a large number of cells at the same time, data validation can take a significant amount of time. In such cases, a progress bar may appear to provide the end user with an indication of progress. If the end user clicks Cancel, the validation stops at that point.

How to Enable or Disable ADF Desktop Integration Data Entry Validation

Integrated Excel workbooks enable ADF Desktop Integration data entry validation by default. You enable or disable ADF Desktop Integration data entry validation by configuring the `DataEntryValidationEnabled` workbook property, described in [ADF Desktop Integration Compatibility Properties](#).

Before you begin:

It may be helpful to have an understanding of ADF Desktop Integration data entry validation. See [Providing Data Entry Validation Using ADF Desktop Integration](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Adding Validation to an Integrated Excel Workbook](#).

To enable or disable ADF Desktop Integration Data Entry Validation:

1. Open the integrated Excel workbook.
2. From the Excel Ribbon, in the Oracle ADF tab, click **Workbook Properties**.

3. In the Edit Workbook Properties dialog, expand Behavior > Compatibility and set the `DataEntryValidationEnabled` property appropriately:
 - **True:** Enables ADF Desktop Integration data entry validation.
 - **False:** Disables ADF Desktop Integration data entry validation.
4. Click **OK**.

Providing Data Validation Using Excel

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 `EditWarehouseInventory-DT.xlsx` workbook so that users can enter only whole numbers in the cells of this field.

If you apply custom validation to cells that render lists of values, the validation is propagated when ADF Desktop Integration populates cells with lists of values at runtime. Note, however, that ADF Desktop Integration overwrites at runtime any custom validation applied for components with lists of values. This is because ADF Desktop Integration applies its own list-constraint validation, which is invoked at runtime. For information about lists of values, see [Working with Lists of Values](#).

For information about data validation in Excel, see Excel's documentation.

How to Upload Excel Cell Errors to the Fusion Web Application

By default, ADF Desktop Integration rejects Excel cell error values that integrated Excel workbooks send to the Fusion web application. As a result, you cannot successfully upload these values from your integrated Excel workbook. You change this default behavior by configuring the `RejectExcelErrorsEnabled` workbook property.

The `RejectExcelErrorsEnabled` workbook property is one of a number of properties that you can configure in integrated Excel workbooks if you want to use a feature that was not available in ADF Desktop Integration when you originally created your integrated Excel workbook. See [ADF Desktop Integration Compatibility Properties](#).

Before you begin:

It may be helpful to have an understanding of the data entry validation options that you can provide for an integrated Excel workbook. See [Providing Data Entry Validation for an Integrated Excel Workbook](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Adding Validation to an Integrated Excel Workbook](#).

To successfully upload Excel cell error values:

1. Open the integrated Excel workbook.
2. From the Excel Ribbon, in the Oracle ADF tab, click **Workbook Properties**.
3. In the Edit Workbook Properties dialog, expand Behavior > Compatibility and set the `RejectExcelErrorsEnabled` property appropriately:

- **True:** ADF Desktop Integration rejects the Excel cell error value so that it is not committed to the Fusion web application's database. This is the default value.
 - **False:** ADF Desktop Integration accepts the Excel cell error value and commits it to the Fusion web application's database if server-side data validation permits.
4. Click **OK**.

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 end users enter or edit in one of the ADF Desktop Integration components can be validated against set rules and conditions in the ADF Model layer.

For an ADF Table component, server-side validation failures can be reported promptly to end users if automatic row refresh is configured for the component. Enable automatic row refresh in the component and identify the column(s) that will contain the data values you want to validate using the `TriggersRowRefresh` property. At runtime, the automatic row refresh sends the modified value to the ADF Model layer which triggers validation. As a result, end users do not need to invoke the `Upload` action to view validation failures.

Row data may become inconsistent in an ADF Table component that is configured to support automatic row refresh if end users enter invalid values. Add entity-level validation rules to help prevent the integrated Excel workbook uploading inconsistent data.

Note:

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 information about automatic row refresh, see [Configuring Automatic Row Refresh in an ADF Table Component](#).

See the following content in the *Developing Fusion Web Applications with Oracle Application Development Framework* for information about:

- Defining validation rules in Oracle ADF, see [Defining Validation and Business Rules Declaratively](#).
- Adding ADF Model layer validation, see [Defining Validation Rules in the ADF Model Layer](#).
- Entity-level validation rules, see [Types of Entity Object Validation Rules](#).

Providing a Row-by-Row Status on an ADF Table Component

The Status Viewer appears by default if errors occur during the attempted invocation of a number of ADF Table component actions.

Errors that occur during the invocation of the following actions cause the automatic display of the Status Viewer:

- `DeleteFlaggedRows`
- `Upload`
- `UploadAllOrNothing`
- `DoubleClickActionSet` invoked from an ADF Table component's column

End users can view a status message in the Status Viewer for each row in the ADF Table component by selecting a cell in the ADF Table component row that interests them.

In addition, 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 13-6 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 value `12345x` has been entered in one row where `12345` or `12345-6789` is expected.

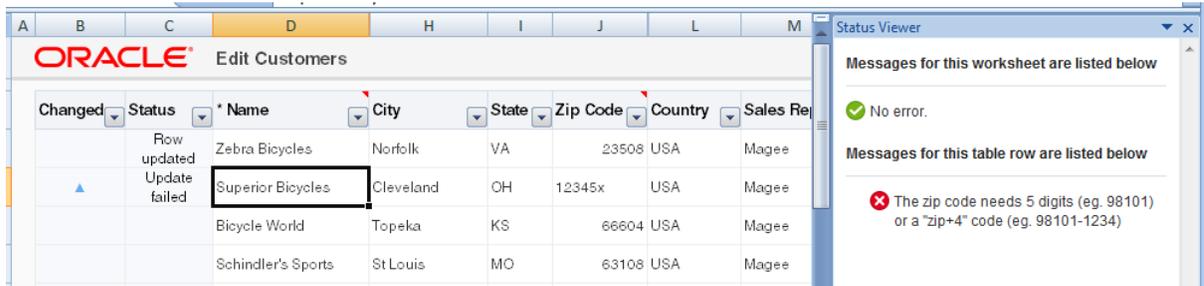
Figure 13-6 ADF Table Component with Changed Rows Before Upload

Changed	Status	Name	Credit Rating	Phone	Address	City	State	Zip Code	Region	Country
▲		Zebra Bicycles	Fair		3910 Colley Ave	Norfolk	VA	23508	North America	USA
▲		Superior Bicycles	Good		538 Superior Ave E	Cleveland	OH	12345x	North America	USA
		Bicycle World	Good		5300 SW 17th St	Topeka	KS	66604	North America	USA
		Schindler's Sports	Good		4479 Forest Park Ave	St Louis	MO	63108	North America	USA
		Barry's Basketball	Good		56 E Superior St	Chicago	IL	60611	North America	USA

Figure 13-7 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. If a row fails to update, the Status Viewer appears automatically, as shown in Figure 13-7 and displays a message describing why the row failed to update.

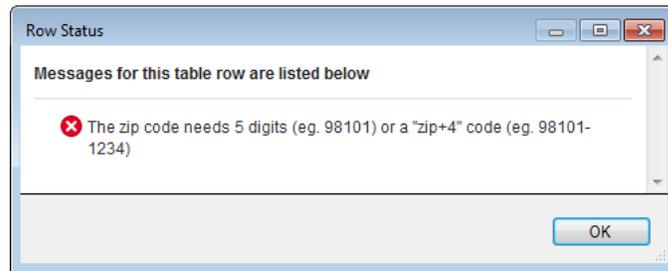
 **Note:**
A number of columns have been hidden in order to display the Status Viewer in [Figure 13-7](#).

Figure 13-7 ADF Table Component with Changed Rows After Upload



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 13-8](#) shows the dialog that appears if the end user double-clicks the cell in [Figure 13-7](#) that displays `Update failed` in the **Status** column.

Figure 13-8 Dialog Displaying Row Error Message



For information about the `_ADF_StatusColumn` column, see [Special Columns in the ADF Table Component](#).

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.

 **Note:**

For security reasons, the implementation of the customer error handler must ensure that the HTML content returned is properly filtered and escaped so that it can be considered trusted content by the browser.

For information about creating a custom error handler, see *Customizing Error Handling in Developing Fusion Web Applications with Oracle Application Development Framework*.

Handling Data Conflicts When Uploading Data from a Workbook

Expose a `DownloadFlaggedRows` action to allow users selectively download rows that may have been modified in another session and specify a row-specific attribute of the tree binding for the `ChangeIndicatorAttribute` property to determine whether a row has been modified.

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 *How to Protect Against Losing Simultaneously Updated Data in 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.

[Using an Integrated Excel Workbook Across Multiple Web Sessions](#) provides information about using an integrated Excel workbook across multiple sessions. For information about flagging rows, see [Row Flagging in an ADF Table Component](#). For information about invoking component actions, see [How to Invoke Component Actions in an Action Set](#). For information about the components that the ADF Table component supports, see [ADF Table Component Properties and Actions](#).

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**.

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.

14

Testing Your Integrated Excel Workbook

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:

- [About Testing Your Integrated Excel Workbook](#)
- [Testing Your Fusion Web Application](#)
- [Validating the Integrated Excel Workbook Configuration](#)
- [Testing Your Integrated Excel Workbook](#)

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

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 14-1](#) shows the buttons of the **Oracle ADF** tab in design mode and in test mode.

Figure 14-1 Run and Stop buttons in Oracle ADF tab



Buttons in Design mode

Buttons in Test mode

Additional Functionality for Testing an Integrated Excel Workbook

After you have validated and tested your integrated Excel workbook, you may need to perform additional steps to make your workbook available to end users.

- **Publishing your integrated Excel workbook:** After you test and validate your workbook, you must publish it. See [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. See [Deploying a Published Workbook with Your Fusion Web Application](#).

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. You may also want to test the view instances of the ADF application module before you test the Fusion web application.

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 [Verifying That Your Fusion Web Application Supports ADF Desktop Integration](#). For information about testing ADF application module, see Using the Oracle ADF Model Tester for Testing and Debugging in *Developing Fusion Web Applications with 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 [How to Reload a Page Definition File in an Excel Workbook](#).

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.

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:

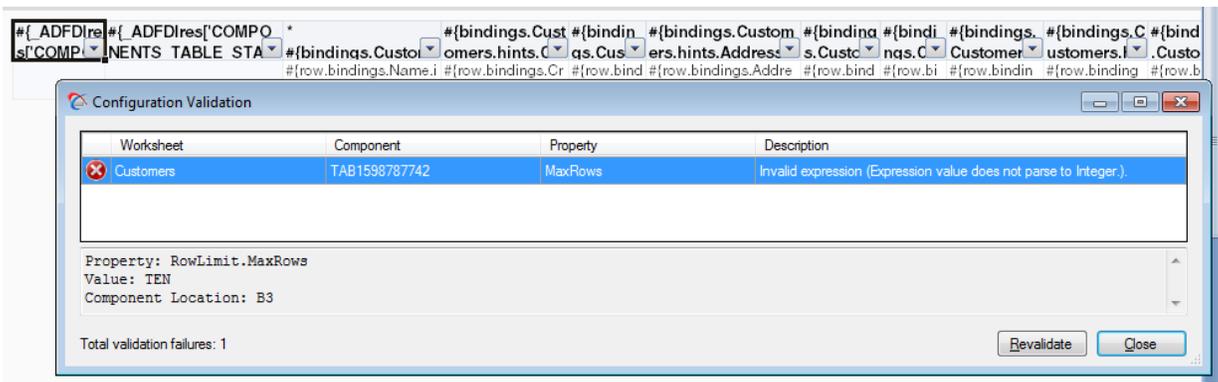
1. Open the integrated Excel workbook.
2. In your integrated Excel workbook, click the **Oracle ADF** tab.
3. In the Test group, click **Validate**.

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 14-2](#) illustrates a validation failure message of an invalid EL expression.

Figure 14-2 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.

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.

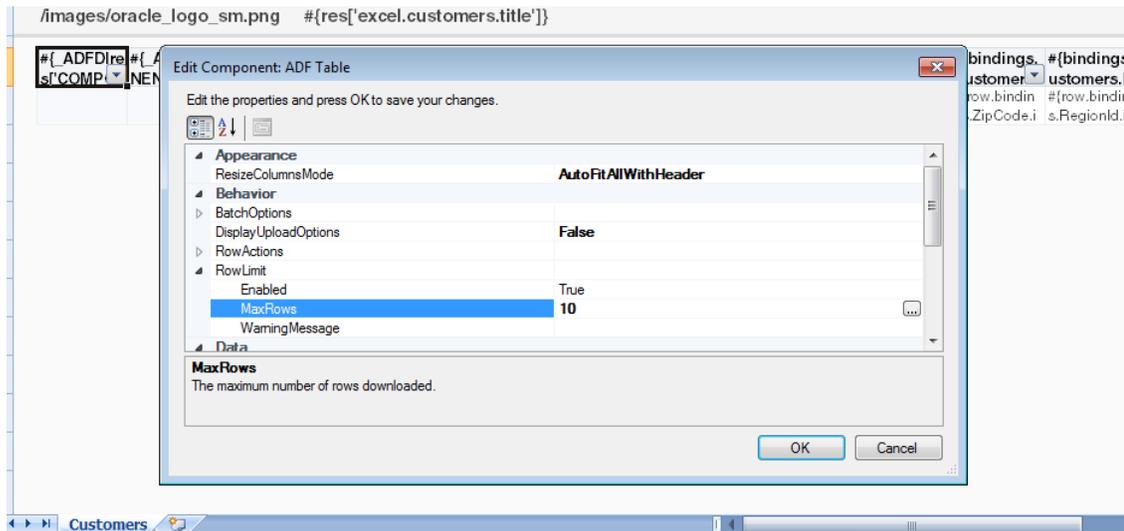
How to Fix Validation Failures

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 14-2](#), note the component location and other details (for example, property name) that the Configuration Validation dialog provides.
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 14-3](#) illustrates the property editor for the ADF Table component with a valid value of 10 for the `RowLimit.MaxRows` property.

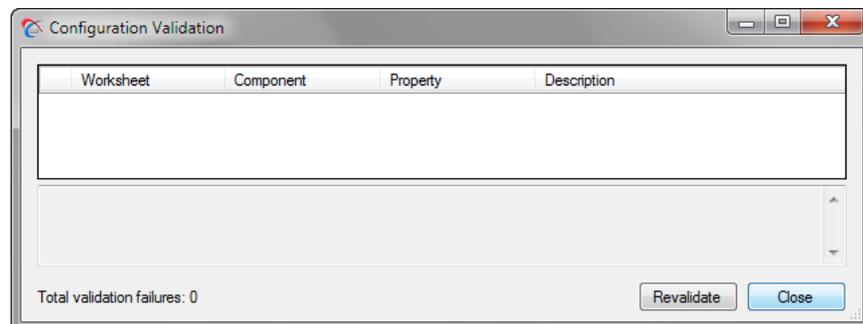
Figure 14-3 Resolving Validation Failure



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 14-4](#) illustrates the Configuration Validation dialog with no warnings or error messages.

Figure 14-4 Configuration Validation Dialog with No Validation Failure Messages



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 `Verbose`. For information about enabling client-side logging, see [About Client-Side Logging](#).

Testing Your Integrated Excel Workbook

As you integrate your Excel workbook 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 information about test mode and design mode, see [About Development Tools](#).

There are some differences between the test mode and the runtime mode when you run the integrated Excel workbook. [Table 14-1](#) lists these differences.

Table 14-1 Differences between Test mode and Runtime mode

Test mode	Runtime mode
Does not perform tamper check	Performs tamper check, if enabled by the server
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 information about these log files, see [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 client version is upgraded if a newer version is available on the server.

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.

Deploying Your Integrated Excel Workbook

Describes how to publish and deploy a workbook integrated with a Fusion web application to end users, how to pass parameters from the Fusion web application to the integrated Excel workbook, and how to integrate the ADF Workbook Composer into your Fusion web application.

This chapter includes the following sections:

- [About Deploying Your Integrated Excel Workbook](#)
- [Making ADF Desktop Integration Available to End Users](#)
- [Publishing Your Integrated Excel Workbook](#)
- [Deploying a Published Workbook with Your Fusion Web Application](#)
- [Passing Parameter Values from a Fusion Web Application Page to a Workbook](#)
- [Customizing Workbook Integration Metadata at Runtime](#)
- [Integrating ADF Workbook Composer into Your Fusion Web Application](#)

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 [Publishing Your Integrated Excel Workbook](#). After you have published the Excel workbook, you can deploy it using one of the methods outlined in *Deploying Fusion Web Applications of Developing Fusion Web Applications with Oracle Application Development Framework*.

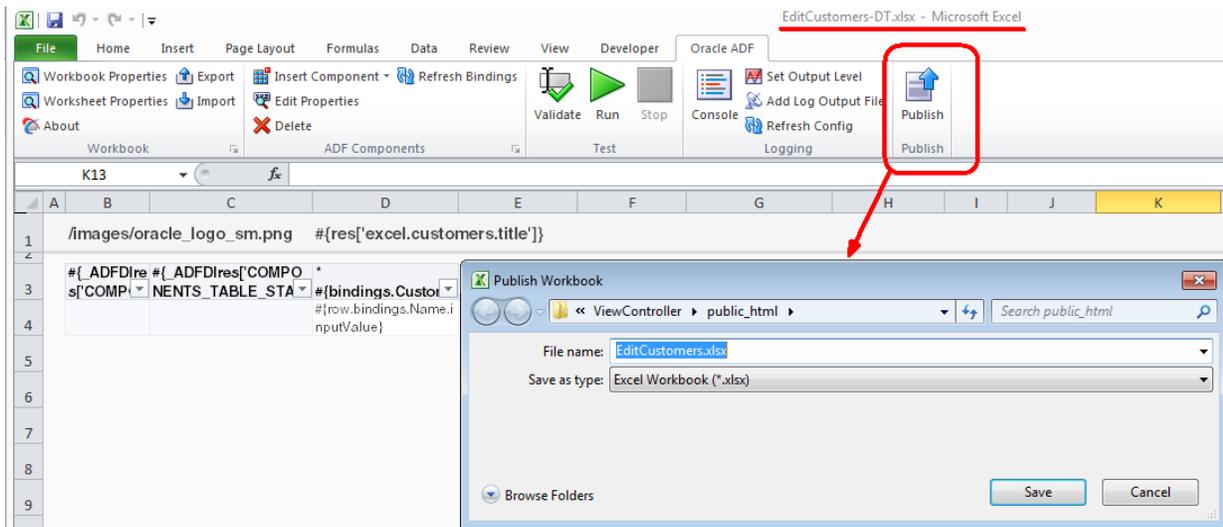
You should make ADF Desktop Integration's `adfdi-excel-addin-installer.exe` available to end users so that they can install the ADF Desktop Integration add-in. For information, see [Making ADF Desktop Integration Available to End Users](#).

The end users that you deploy an integrated Excel workbook to must install the ADF Desktop Integration add-in for Excel on their Windows-based computers.

Integrated Excel Workbook Deployment Use Cases and Examples

You use the **Publish** button of the **Oracle ADF** tab to save a published copy of the workbook. [Figure 15-1](#) 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 15-1 Publish Workbook Dialog



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. The following sections describe other functionality 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. See [Passing Parameter Values from a Fusion Web Application Page to a Workbook](#).

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 ADF Desktop Integration.

For information about how end users install ADF Desktop Integration, see [How to Install ADF Desktop Integration on Your System](#).

For information about how to make the installer available to end users, see [How to Install the ADF Desktop Integration Add-in From a Web Server](#).

The installation program (`adfdi-excel-addin-installer.exe`) is available in the following directory:

```
MW_HOME\oracle_common\modules\oracle.adf.desktopintegration
```

where `MW_HOME` is the Middleware Home directory.

By default, ADF Desktop Integration-enabled web applications display a system check that verifies the end user's environment when the end user attempts to download an integrated Excel workbook. A Java applet performs this system check to verify that the ADF Desktop Integration add-in is present on the end user's machine. If the add-in is found, the workbook download begins automatically. Otherwise, ADF Desktop Integration prompts the end user to install the add-in. If Java is not installed on the end

user's machine or is disabled by the end user's security settings, the Java applet is unable to verify the presence of the ADF Desktop Integration add-in. ADF Desktop Integration informs the end user that the installation of the add-in cannot be verified. It presents the end user with the option to download the workbook and/or install the add-in.

Publishing Your Integrated Excel Workbook

After you finish configuring the Excel workbook with Oracle ADF functionality, you must publish it. Publishing a workbook prepares the integrated Excel workbook for use by end users at runtime.

ADF Desktop Integration 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 scripts, such as an Ant script, to publish an integrated Excel workbook from your Fusion web application.

Note:

- 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.

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.

Before you begin:

It may be helpful to have an understanding about how to publish your integrated Excel workbook. See [Publishing Your Integrated Excel Workbook](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Deploying Your Integrated Excel Workbook](#).

For information configuring properties, such as the `ApplicationHomeFolder` and `WebPagesFolder` properties, see [How to Configure a New 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.
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 15-1](#). 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.

ADF Desktop Integration prompts you to save the workbook in the `ViewController\public_html` directory of the JDeveloper application workspace. Remove any suffixes, such as `-DT`, that may have been appended to the workbook when it was in design mode so that end users see a meaningful filename. For example, the Summit sample application publishes the workbook to edit customers using the `EditCustomers.xlsx` filename rather than `EditCustomers-DT.xlsx` filename that it used in design mode.

5. Click **Save** to save changes.

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. See [Publishing Your Integrated Excel Workbook](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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\sampldemo\excel\workbook-DT.xlsx -out D:
\Application1\Project1\ViewController\public_html\excel\published
\workbook.xlsx
```

 **Tip:**

For information about the arguments required by the publish tool, run the following command:

```
publish-workbook -help (-h)
```

 **Note:**

- 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.

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.

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. See [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 information about workbook definition, see [Exporting and Importing Excel Workbook Integration Metadata](#).
4. Clears the `ApplicationHomeFolder`, `WebAppRoot`, and `WebPagesFolder` properties from the workbook settings of the published workbook.
5. Clears all design time component placeholders.
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.

Deploying a Published Workbook with Your Fusion Web Application

If you published your integrated Excel workbook, your Fusion web application automatically includes the published workbook when you deploy the web application. Otherwise, add the workbook to the JDeveloper project of your Fusion web application if it is not packaged with the other files that constitute your JDeveloper project.

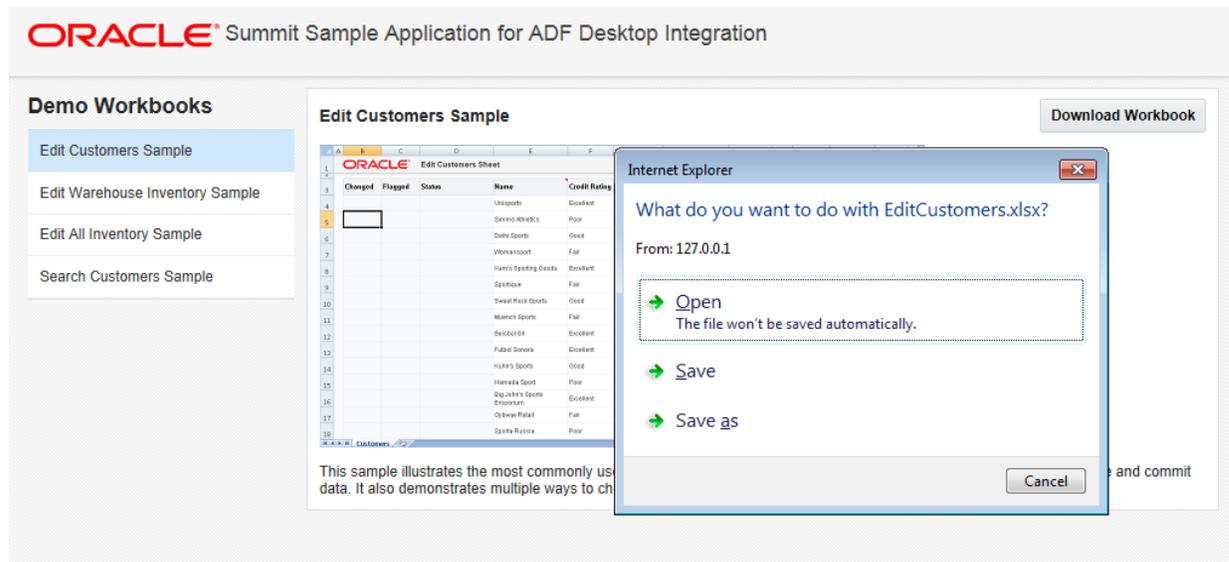
This makes sure that the Excel workbooks you integrate with your Fusion web application get deployed when you deploy your finalized Fusion web application. For example, the Summit sample application for ADF Desktop Integration stores the deployed Excel workbooks that it integrates at the following location:

```
<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 15-2](#) shows Internet Explorer's File Download dialog, which was invoked by clicking the **Download Workbook** button for the Edit Customers Sample workbook on the `MainPage.jsf` page of the Summit sample application for ADF Desktop Integration.

Figure 15-2 Invoking an Integrated Excel Workbook from a Fusion Web Application



To enable the functionality illustrated in [Figure 15-2](#), 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 [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 download integrated Excel workbooks from web page buttons and menus. The following list provides some examples:

- **Button**
Display a button on the web page that, when clicked, invokes the integrated Excel workbook. For example, the **Download Workbook** button in [Figure 15-2](#) is a button component that the `MainPage.jsf` page exposes.
- **Selection list**
Use the ADF Faces `selectOneChoice` component with a button to invoke an integrated Excel workbook.

- Menu
Use the ADF Faces `goMenuItem` component.

For information about creating web pages for a Fusion web application, see *Getting Started with ADF Faces and JDeveloper in Developing Web User Interfaces with Oracle ADF Faces*.

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.

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`, in the `oracle.adf.desktopintegration.model.ear` file.

- 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`, in the `oracle.adf.desktopintegration.war` file.

What Happens at Runtime: End User Requests a Published Workbook

When `web.xml` is configured for a Fusion web application that uses ADF Desktop Integration,

The following events occur when you configure a Fusion web application to use ADF Desktop Integration:

- The `DIExcelDownloadFilter` filter is defined.
- Filter mappings are defined for `*.xlsx` and `*.xlsm` files.

At runtime, 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. Parameter values can also be passed from the web application to the workbook, as described in [Passing Parameter Values from a Fusion Web Application Page to a Workbook](#).

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()`).

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.

Workbook parameters can be used to pass context from the user's web page to the integrated workbook. The passed context may be sent back to the web application from the integrated workbook to affect application state (for example, what data renders in the workbook). The Summit sample application, for example, displays a list of warehouses to the end user, as shown in [Figure 15-3](#). When an end user clicks a **Download Workbook** button, the Summit sample application passes the value of the `WarehouseID` parameter to the workbook to download. The passed `WarehouseID` parameter controls which warehouse's data renders in the worksheet for editing.

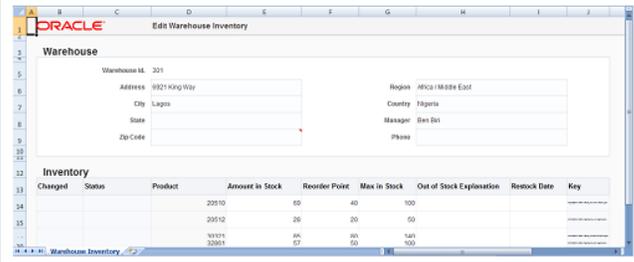
Figure 15-3 Downloading Workbooks According to Parameter Value

ORACLE Summit Sample Application for ADF Desktop Integration

Demo Workbooks

- Edit Customers Sample
- Edit Warehouse Inventory Sample**
- Edit All Inventory Sample
- Search Customers Sample

Edit Warehouse Inventory Sample



This sample illustrates how to use ADF Desktop Integration Form components with a detail table. You can download and update data in a master form and its detail table. This sample also demonstrates how to use workbook parameters to control the workbook initialization.

Warehouses

201	68 Via Centrale Sao Paulo, Brazil	Download Workbook
112	2350 N Main Roswell, USA	Download Workbook
111	7120 N Academy Blvd Colorado Springs, USA	Download Workbook
110	1105 E Boxelder Road Gillette, USA	Download Workbook

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 [Configuring the ADF Desktop Integration Excel Download Filter](#).
2. Use Name/Value pairs as URL arguments in the web page of the Fusion web application that allows end user to download the workbook. See [How to Configure the Fusion Web Application's Page to Pass Parameters](#).

 **Note:**

The runtime URL-encoded value of the entire query string to the right of ? 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.

For example, the total size of the string result to the right of ? when the following EL expression is evaluated and then URL-encoded must be less than 2048 bytes.

```
"/excel/EditWarehouseInventory.xlsx?  
WarehouseId=#{item.bindings.Id.inputValue}"
```

If you need to pass information that exceeds this limit, consider storing it temporarily in a (custom) database table and only pass a unique token to look up that information later. This technique also protects the context information from undesirable exposure.

3. Define the parameter name in the Edit Workbook Properties dialog and in the Edit Worksheet Properties dialog. See [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. See [How to Configure the Page Definition File for the Worksheet to Receive Parameters](#).

Figure 15-4 illustrates the steps implemented in the Summit sample application to pass a parameter from the web application to the `EditWarehouseInventory-DT.xlsx` workbook. For information about the Summit sample application, see [Introduction to the ADF Desktop Integration Sample Application](#).

Figure 15-4 Configuring Workbook and Fusion Web Application to Pass Parameters

The figure illustrates the configuration process for passing parameters from a Fusion Web Application page to a workbook. It consists of several components:

- MainPage.jsf:** A snippet of code showing a button with the following attributes:


```
id="gc5">
<af:button text="Download Workbook" id="b2"
destination="/excel/EditWarehouseInventory.xlsx?WarehouseId=#{item.bindings.Id.inputValue}"/>
```
- Warehouses Table:** A table with one row:

201	68 Via Centrale	Download Workbook
	Sao Paulo, Brazil	
- Edit Workbook Properties:** A dialog box showing the configuration for the workbook. The **Parameters** section is expanded, showing a parameter named **WarehouseId** with a value of **WarehouseId**. The **RemoteServletPath** is set to **/adf:RemoteServlet** and the **WebAppRoot** is **http://127.0.0.1:7101/summit/**.
- Edit Worksheet Properties:** A dialog box showing the configuration for the worksheet. The **Parameters** section is expanded, showing a parameter named **WarehouseIdParam** with a value of **#{workbook.params.WarehouseId}**. A **Parameter** dialog box is also shown, allowing the selection of **WarehouseIdParam (parameter)** as the value.
- ExcelWarehouseInventory.xml:** A snippet of XML code showing the configuration for the workbook:


```
<parameters>
  <parameter id="WarehouseIdParam" value="value"/>
</parameters>
<executables>

<methodAction id="FilterWarehouseMasterById" RequiresUpdateModel="true" Action="invokeMethod"
MethodNames="FilterWarehouseMasterById" IsViewObjectMethod="false"
DataControl="SummitAppModuleDataControl"
InstanceName="data.SummitAppModuleDataControl.dataProvider">
  <NamedData NDName="warehouseId" NDValue="#{bindings.WarehouseIdParam}"
NDType="java.lang.String"/>
</methodAction>
```

How to Configure the Fusion Web Application's Page to Pass Parameters

A component, such as `<af:button>`, can be used to allow end users to download a published copy of an integrated workbook. The component's destination URL references 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 information about downloading files using action components, see *How to Use an Action Component to Download Files* 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. See [Passing Parameter Values from a Fusion Web Application Page to a Workbook](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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:button`) 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 15-1](#) describes the properties of `af:button` component.

Table 15-1 Properties for `af:button` Tag

Property	Value
Text	Write the text that appears to end users at runtime. For example, write text such as the following to appear at runtime: Download Workbook
Destination	Invoke the expression builder to write an EL expression that specifies the integrated Excel workbook and the values to download as a URL argument: For example, write an EL expression such as the following: <pre>destination="/excel/EditWarehouseInventory.xlsx?WarehouseId=#{item.bindings.Id.inputValue}"</pre> Per RFC3986 , URL argument names and values must not contain any reserved characters, for example: #, ?, and &. You may find it necessary to use a Java bean method to retrieve the values of binding container attributes when using them as URL argument values. The bean method can URL-encode the values so that the URL conforms to the specification. See Java's <code>java.net.URLEncoder</code> class for more details about URL encoding.

4. (Optional) Expand the **Behavior** section and specify component IDs for the `partialTriggers` property that, when invoked, update the values of the `af:button` 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 required examples in this procedure:

```
<af:button text="Download Workbook" id="b2"
          destination="/excel/EditWarehouseInventory.xlsx
                    ?WarehouseId=#{item.bindings.Id.inputValue}"/>
```

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. See [Passing Parameter Values from a Fusion Web Application Page to a Workbook](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Deploying Your Integrated Excel Workbook](#).

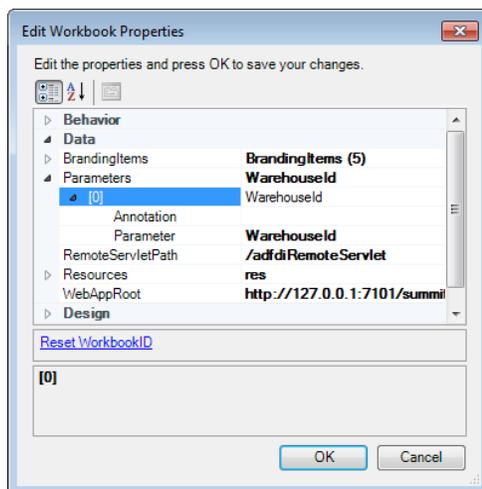
To configure the workbook `Parameters` property:

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 parameter and configure its properties as follows:
 - In the **Parameter** field, define the parameter name that you plan to use as a URL argument for the `af:button` tag's `Destination` property and later bind to a page definition parameter, as described in [How to Configure the Fusion Web Application's Page to Pass Parameters](#).

For example, the `EditWarehouseInventory-DT.xlsx` workbook defines the `WarehouseID` parameter value, as illustrated in [Figure 15-5](#).

Tip:

Make sure that the value you define will be valid for use in a standard URL query string. The parameter name you use should be a simple identifier so that it functions properly when referenced in EL expressions.

Figure 15-5 Workbook Parameters

- (Optional) In the **Annotation** field, enter a description of the workbook parameter.
5. Repeat Step 4 as necessary to add other workbook parameters.
 6. Click **OK**.

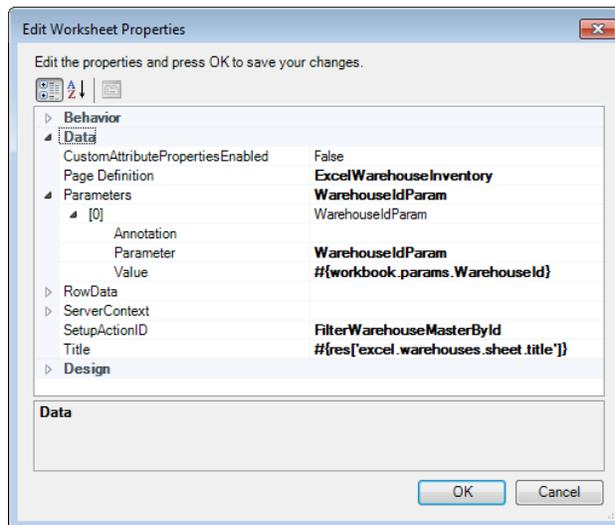
For information about the workbook `Parameters` property, see [Table A-19](#).

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 illustrated in [Figure 15-6](#) from the `EditWarehouseInventory-DT.xlsx` workbook:
 - In the **Parameter** field, specify a parameter element that you added to the page definition file associated with the worksheet, as described in [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 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 parameter.
 - (Optional) In the **Annotation** field, enter a description of the worksheet parameter.

Figure 15-6 Worksheet Parameters

5. Repeat Step 4 as necessary to add other workbook parameters.
6. Click **OK**.

For information about the worksheet `Parameters` property, see [Table A-20](#).

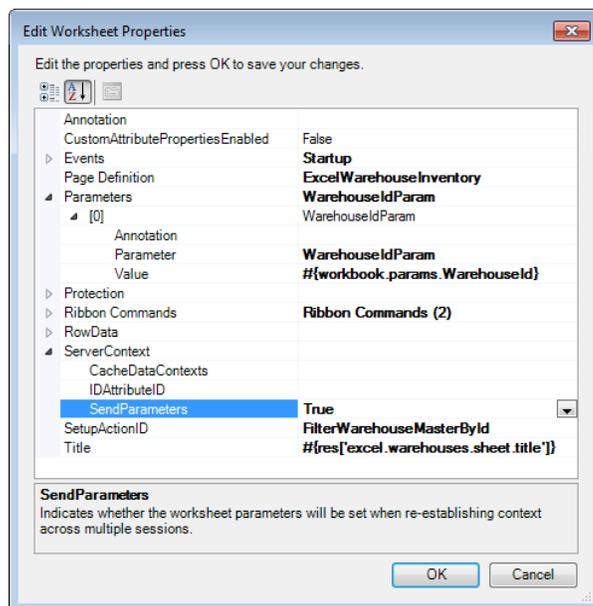
For use cases where the workbook parameter values are necessary to set up the initial server state on each new user session, set the `SendParameters` property to `True`. Additionally, you should specify a method action binding to invoke for the worksheet's `SetupActionID` that initializes the server state using the workbook parameter values.

To configure the worksheet `SendParameters` and `SetupActionID` properties:

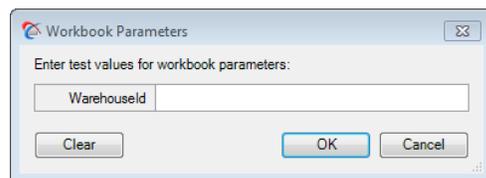
1. Open the integrated Excel workbook.
2. In the Workbook group of the Oracle ADF tab, click **Worksheet Properties**.
3. In the Edit Worksheet Properties dialog, set the values of `SendParameters` and `SetupActionID` as shown in the [Table 15-2](#) and [Figure 15-7](#):

Table 15-2 SendParameters and SetupActionID Properties

Set this property to...	This value...
<code>SendParameters</code>	True to make sure that the worksheet parameters are set in the binding container for the worksheet. When set to <code>True</code> , parameters are sent every time when the metadata is requested and the first time when data is requested, during each user session. When set to <code>False</code> (the default value), the explicit sending of worksheet parameters does not take place.
<code>SetupActionID</code>	Specify a method action binding to invoke that initializes the server state using the workbook parameter values. See Using Explicit Worksheet Setup Action .

Figure 15-7 SendParameters and SetupActionID Properties**4. Click OK.**

When entering the Test mode, the Workbook Parameter dialog prompts you to enter test parameter values. Figure 15-8 shows the Workbook Parameters dialog that accepts test values for the workbook in an input field.

Figure 15-8 Workbook Parameters dialog

While testing, the values entered here are used for the workbook parameter values. If you have bound the workbook parameters to page definition parameters in the worksheet, the values you enter here will be sent to the binding container. 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 from the `EditWarehouseInventory-DT.xlsx` workbook, the `FilterWarehouseMasterById` method action can be used as the worksheet's setup action (`SetupActionID`). This causes 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 information about `SetupActionID`, see [Using Explicit Worksheet Setup Action](#).

The same method action could also be configured as a part of an 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 information about ribbon commands and `Startup` event, see [How to Configure a Worksheet Ribbon Command for the Runtime Ribbon Tab](#) and [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.

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 [How to Configure the Fusion Web Application's Page to Pass Parameters](#).

The following example shows the `WarehouseIdParam` parameter defined in the `ExcelWarehouseInventory.xml` page definition file that is associated with the `EditWarehouseInventory-DT.xlsx` workbook:

```
<parameters>
  <parameter id="WarehouseIdParam" value="value"/>
</parameters>
```

- Add a method action binding that invokes an application module method. The following example shows an implementation in the `ExcelWarehouseInventory.xml` page definition file that is associated with the `EditWarehouseInventory-DT.xlsx` workbook.

```
<methodAction id="FilterWarehouseMasterById" RequiresUpdateModel="true"
  Action="invokeMethod" MethodName="FilterWarehouseMasterById"
  IsViewObjectMethod="false" DataControl="SummitAppModuleDataControl"
  InstanceName="data.SummitAppModuleDataControl.dataProvider">
  <NamedData NDName="warehouseId" NDValue="{bindings.WarehouseIdParam}"
    NDType="java.lang.String"/>
</methodAction>
```

For information about configuring a page definition file, see [Working with Page Definition Files for an Integrated Excel Workbook](#) and [Working with Page Definition](#)

Files in *Developing Fusion Web Applications with Oracle Application Development Framework*.

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:button` tag) is evaluated, the current parameter value (for example, `warehouseID`) 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 DCPParameters` before the binding container is refreshed. For information about how worksheet parameters are mapped to binding containers, see [How to Configure the Page Definition File for the Worksheet to Receive Parameters](#).

For use cases where workbook parameter values are necessary to set up the initial server state on each new user session, set the `Worksheet.ServerContext.SendParameters` property to `True`. Additionally, you should specify a method action binding to invoke for the worksheet's `SetupActionID` that initializes the server state using the workbook parameter values. For information about the worksheet `SetupActionID` property, see [Using Explicit Worksheet Setup Action](#).

In the `EditWarehouseInventory.xlsx` workbook, the `FilterWarehouseMasterById` method is invoked on each user session to set up the correct server state using the workbook `WarehouseId` parameter value stored in the downloaded workbook.

To reset the initialization state for all worksheets in the workbook, invoke the `ClearAllData` action. For information about the `ClearAllData` action, see [Table A-18](#).

Customizing Workbook Integration Metadata at Runtime

ADF Desktop Integration supports Oracle Metadata Services (MDS) based runtime customization.

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 information about publishing a customization-enabled workbook, see [Publishing Your Integrated Excel Workbook](#).

For information about MDS, see [Customizing Applications with MDS in *Developing Fusion Web Applications with Oracle Application Development Framework*](#).

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 information about Metadata Repository, see [Managing the Metadata Repository in *Administering Oracle Fusion Middleware*](#).

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. See [Customizing Workbook Integration Metadata at Runtime](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [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.

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.

An application developer might include seeded customizations with the application and/or integrate the ADF Workbook Composer, as described in [Integrating ADF Workbook Composer into Your Fusion Web Application](#). For example, an application can provide a web page where users can customize the columns of a table in a customization-enabled workbook. Users can remove certain columns from the table on the web page and then download the customization-enabled workbook and see changes takes effect in the workbook.

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-DT.xlsx-workbook-defintion.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.

Integrating ADF Workbook Composer into Your Fusion Web Application

The ADF Workbook Composer is an ADF Task Flow that enables an authorized user to customize an integrated Excel workbook from the runtime web user interface of the Fusion web application.

To use the ADF Workbook Composer, you must have a customization-enabled workbook integrated into your Fusion web application and have its metadata managed by MDS. For information about customization-enabled workbook, see [Customizing Workbook Integration Metadata at Runtime](#).

Using the ADF Workbook Composer, the end user may perform the following actions at runtime:

- Edit or delete ADF components of the integrated Excel workbook
- Reposition components in the worksheet
- Edit tooltips, labels, and source of ADF components
- Delete worksheets

How to Integrate ADF Workbook Composer into Your Fusion Web Application

The ADF Workbook Composer task flow is available in the `adf-workbook-composer.jar` file as an ADF Library. The jar file is available in the `MWHOME/oracle_common/modules/oracle.adf.desktopintegration` directory.

To integrate ADF Workbook Composer in your Fusion web application:

1. Open your Fusion web application in JDeveloper.
2. Add the `adf-workbook-composer.jar` file as an ADF Library jar to your Fusion web application.
 - a. In the Applications window, select and right-click the project (ViewController, for example) and choose **Project Properties**.
 - b. In the Project Properties dialog, select **Libraries and Classpath**.
 - c. In the Libraries and Classpath page, click **Add Library**.
 - d. In the Add Library dialog, click **New**.

- e. In the Create Library dialog, enter ADF Workbook Composer Runtime as **Library Name**.
- f. Click **Add Entry**.
- g. Navigate to the `MWHOME/oracle_common/modules/oracle.adf.desktopintegration` directory, select the `adf-workbook-composer.jar` file, and click **Open**.

 **Note:**

Make sure to clear the **Deployed by Default** checkbox to avoid duplicate copies of the `adf-workbook-composer.jar` file appearing on the class path at runtime. The `oracle.adf.desktopintegration` shared library includes the `adf-workbook-composer.jar` file, as described in [What Happens at Runtime: ADF Workbook Composer is Invoked](#).

3. Select and expand the ADF Workbook Composer Runtime Library in the Applications window.
If libraries are not visible, select **View > Application Projects > Show Libraries**.
4. Locate the `workbook-customization-task-flow.xml` file under `WEB-INF\oracle\adf\workbookcomposer\view\taskflows` and drag-and-drop the file to import the task flow within the host page.
5. If necessary, set up the desired customization context.
6. Configure the MDS repository in `adf-config.xml` and make sure that workbook metadata files are accessible on the metadata path.
7. Provide the required workbook metadata path and workbook name parameters for the task flow.
8. If the Fusion web application is authorization-enabled, you would need to configure security policies to grant resource access to users for the following task flows available in the `/WEB-INF/oracle/adf/workbookcomposer/view/taskflows/` directory of the workbook composer jar file.
 - `button-customization-task-flow.xml`
 - `form-component-customization-task-flow.xml`
 - `image-customization-task-flow.xml`
 - `not-supported-task-flow.xml`
 - `read-only-table-customization-task-flow.xml`
 - `ribbon-command-customization-task-flow.xml`
 - `table-customization-task-flow.xml`
 - `workbook-customization-task-flow.xml`
9. Run the host web page to make sure that the workbook composer renders correctly.

What Happens at Runtime: ADF Workbook Composer is Invoked

The ADF Workbook Composer task flow is available in the `adf-workbook-composer.jar` as an ADF Library jar file. This jar is included in the `oracle.adf.desktopintegration` shared library. The `oracle.adf.desktopintegration` shared library is installed as part of the Application Development Runtime installation process and is included in the JRF domain extension template. If you have installed the Application Development Runtime, the ADF Workbook Composer task flow will be available at runtime when the Fusion web application runs on WebLogic Server. For information about the installation of the Application Development Runtime, see *Deploying ADF Applications* in *Administering Oracle ADF Applications*.

At runtime, the customization made from the ADF Workbook Composer takes effect immediately without restarting the Fusion web application. End users that match the customization context associated with the workbook customization will see the customization after they download and open a new copy of the integrated Excel workbook, or invoke the `ClearAllData` workbook action on an initialized workbook and then log in.

What You May Need to Know About ADF Workbook Composer

The ADF Workbook Composer task flow requires two parameters:

- `WorkbookName` – The name of the published workbook that the users will be customizing at runtime. The name will be displayed in the composer.
- `WorkookMetadataPath` – The path to the workbook metadata file. This is the path used by MDS to locate the metadata file for the workbook to be customized. The workbook metadata file is generated when the design-time workbook is published. The metadata path is determined by the location to which the workbook is published.

16

Using an Integrated Excel Workbook Across Multiple Web Sessions

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:

- [About Using an Integrated Excel Workbook Across Multiple Web Sessions](#)
- [Restore Server Data Context Between Sessions](#)
- [Caching of Static Information in an Integrated Excel Workbook](#)
- [Caching Lists of Values for Use Across Multiple Web Sessions](#)
- [Using Explicit Worksheet Setup Action](#)

About Using an Integrated Excel Workbook Across Multiple Web Sessions

ADF assigns a session each time a user logs on to a Fusion web application from an integrated Excel workbook. A session timeout or a logout action terminates this session.

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.

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 allows the session to time out by leaving the workbook open and idle, 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

Test your integrated Excel workbook's behavior across multiple web application sessions. To do this, run the integrated Excel workbook. As you go through the steps of your use case, click the **Logout** workbook ribbon command at various points to end the current web application session. Make a special point of ending the current session between invocations of the ADF Table component's `Download` and `Upload` actions. New web application sessions will be created as needed. If the results are not what you expect, you may need to configure the properties described in subsequent sections of this chapter.

Using an Integrated Excel Workbook Across Multiple Web Sessions 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.

Additional Functionality for Using an Integrated Excel Workbook Across Multiple Web Sessions

After you have validated and tested your integrated Excel workbook across multiple web sessions, you may find that you need to add additional functionality for your workbook. The following sections describe other functionality that you can use:

- **Troubleshooting integrated Excel workbook:** You might encounter some problems while developing or deploying an integrated Excel workbook. See [Troubleshooting an Integrated Excel Workbook](#).
- **Installing ADF Desktop Integration:** You must install ADF Desktop Integration to enable end users to use ADF Desktop Integration and integrated Excel workbooks. See [End User Actions](#).

Restore Server Data Context Between Sessions

For use cases where the behavior of one or more action sets in a worksheet rely on the current model state of the Fusion web application, you must configure your integrated Excel workbook and page definition file to capture and restore the correct model state whenever a new session is established.

A new session can occur whenever:

- The user saves, closes, and re-opens the integrated Excel workbook
- The user invokes the workbook Logout action
- The time between invocation of action sets that contact the Fusion web application exceeds the session timeout value specified for a Fusion web application session

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 [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-20](#).

Before you begin:

It may be helpful to have an understanding of how to restore server data context. See [Restore Server Data Context Between Sessions](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Using an Integrated Excel Workbook Across Multiple Web Sessions](#).

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 16-1](#).

Table 16-1 ServerContext Properties to Restore Server Data Context

For this property...	Enter or select this value...
<code>CacheDataContexts</code>	<p>Typically, you add an element to this collection to restore a non-trivial query that you cannot configure directly in the page definition file. Adding an element to this collection is optional if you do not have to address this scenario. If you add an element to the collection of <code>CacheDataContexts</code>, configure it as follows:</p> <ul style="list-style-type: none"> • <code>RestoreDataContextActionID</code> Specify the action binding (for example, the <code>Execute</code> action binding) that connects to the Fusion web application to restore the data specified by <code>CachedServerContexts</code>. • <code>CachedServerContexts</code> An array that identifies the attribute binding values to cache and set before the action binding specified by <code>RestoreDataContextActionID</code> is invoked. Each element in the array (<code>CachedServerContext</code>) supports the <code>CachedAttributeID</code> and <code>RestoredAttributeID</code> properties. <p>For information about the <code>CacheDataContexts</code> property and its subproperties, see Worksheet Actions and Properties.</p>

Table 16-1 (Cont.) ServerContext Properties to Restore Server Data Context

For this property...	Enter or select this value...
IDAttributeID	Specify the attribute binding that uniquely identifies the row displayed in the current worksheet. At runtime, the value that this property references determines if the server data context has been correctly restored. Typically, you use this property to handle a form. It may be optional otherwise. For information about this property and its subproperties, see Worksheet Actions and Properties .

If your integrated Excel workbook uses parameters and you have deployed it by downloading it from your Fusion web application, see [How to Configure Parameters Properties in the Integrated Excel Workbook](#).

4. Click **OK**.

**Note:**

For integrated Excel workbooks that use a worksheet setup action (or a Parameters and <invokeAction> executable), you may not need to configure RestoreDataContextActionID and CachedServerContexts, if the worksheet setup action can restore server data context when a new session is created.

What Happens at Runtime: How the Integrated Excel Workbook Restores Server Data Context

During 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.

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 16-2](#) describes the types of data that an integrated Excel workbook caches.

Invoking the ClearAllData workbook action described in [Workbook Actions and Properties](#), refreshes all types of cached data described in [Table 16-2](#). [Table 16-2](#) also describes other scenarios where an integrated Excel workbook refreshes cached data.

Table 16-2 Types of Data an Integrated Excel Workbook Caches

This type of data...	Is cached when...	And refreshed when...
Page definition metadata that is not expected to change between user sessions 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 metadata exists.	The page definition metadata is not refreshed unless you download a new copy of the integrated Excel workbook.
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.
Resource bundle strings	The integrated Excel workbook is first initialized. A workbook is initialized when it is opened for the first time after publishing.	The cache of resource bundle strings is not refreshed unless you download a new copy of the integrated Excel workbook.

Caching Lists of Values for Use Across Multiple Web Sessions

ADF Desktop Integration caches the values referenced by the ADF List of Values components 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.

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 use a model-driven list picker, as described in [Adding a Model-Driven List Picker to an ADF Table Component](#), where a list of values references more than two hundred and fifty values. For information about client-side log files, see [Generating Log Files for an Integrated Excel Workbook](#).

Cached lists 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.

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 invoke the `ClearAllData` workbook action. For information about the `ClearAllData` workbook action, see [Workbook Actions and Properties](#).

The changes in your Fusion web application might include changing the definitions of the list bindings associated with the ADF List of Values components exposed in the worksheet. Changing list binding configuration can cause unexpected exceptions in workbooks that have been downloaded and run prior to the change.

**Note:**

An integrated Excel workbook never caches the values that a `ModelDrivenColumnComponent` subcomponent displays in a model-driven list picker. For information about model-driven list pickers, see [Adding a Model-Driven List Picker to an ADF Table Component](#).

For information about lists of values, see [Working with Lists of Values](#).

Using Explicit Worksheet Setup Action

Describes how to specify a setup action that is invoked before the client retrieves the binding container metadata.

ADF Desktop Integration provides several features for configuring a worksheet after the binding container's metadata has been obtained from the server at runtime. However, at times, you might want to configure 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 configure 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 server 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. The `EditWarehouseInventory-DT.xlsx` workbook in the Summit sample application demonstrates an implementation of this feature where the Warehouse Inventory worksheet invokes a method action binding named `FilterWarehouseMasterById`.

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 configuration on the data and binding container.

Before you begin:

It may be helpful to have an understanding of the Explicit Worksheet Setup Action feature. See [Using Explicit Worksheet Setup Action](#).

You may also find it helpful to understand the functionality that can be added using other ADF Desktop Integration features. See [Additional Functionality for Using an Integrated Excel Workbook Across Multiple Web Sessions](#).

To use the worksheet SetupActionID property:

1. Open the worksheet in the integrated Excel workbook.
2. From the Excel Ribbon, click **Worksheet Properties**.
3. In the Edit Worksheet Properties dialog, expand Data and click the browse icon (...) beside the input field for the **SetupActionID** property
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**.

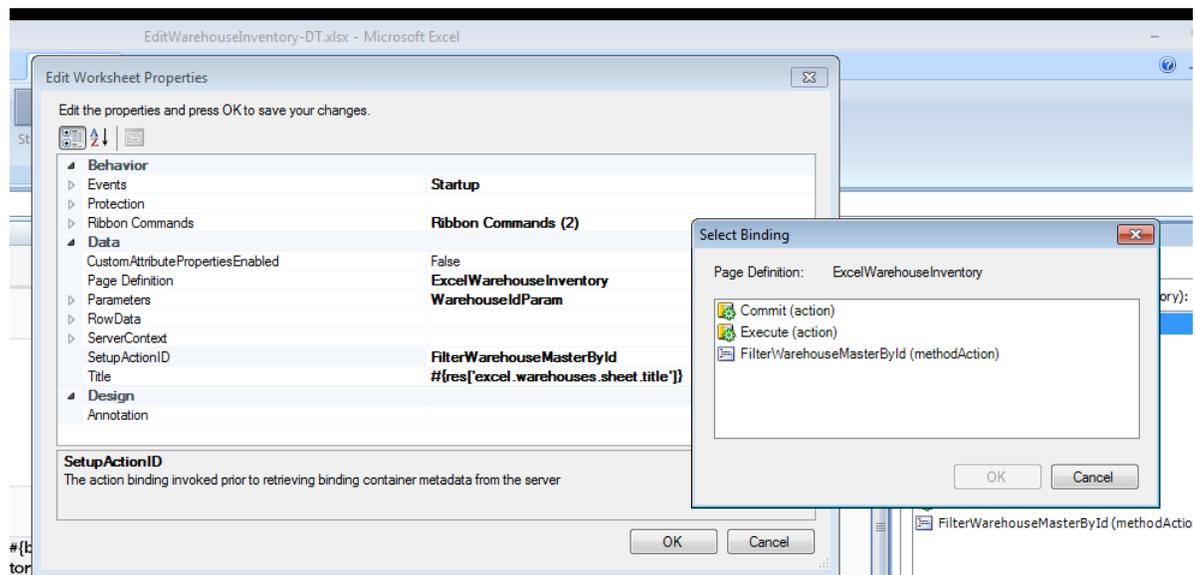
 **Note:**

The SetupActionID property accepts ADFmAction only. A validation error is reported if an invalid method is set for the property.

5. Click **OK** to close the Edit Worksheet Properties dialog.

[Figure 16-1](#) shows the configuration in the EditWarehouseInventory-DT.xlsx workbook where the FilterWarehouseMasterById methodAction binding has been selected.

Figure 16-1 SetupActionID Property in Edit Worksheet Properties Dialog



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 changes 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.

Administering ADF Desktop Integration

Describes system administration tasks for ADF Desktop Integration such as running the ADF Desktop Integration client installer from a web server and adjusting server configuration settings.

Note that before an end user can use the integrated Excel workbook, the ADF Desktop Integration add-in must be installed on the end user's system.

This chapter includes the following sections:

- [Installing and Upgrading ADF Desktop Integration](#)
- [Running the Client Health Check Tool](#)
- [ADF Desktop Integration Logs](#)
- [Security in ADF Desktop Integration](#)
- [Verifying the Client Version of ADF Desktop Integration](#)
- [Verifying Integrated Excel Workbook Metadata](#)
- [Common ADF Desktop Integration Error Messages and Problems](#)

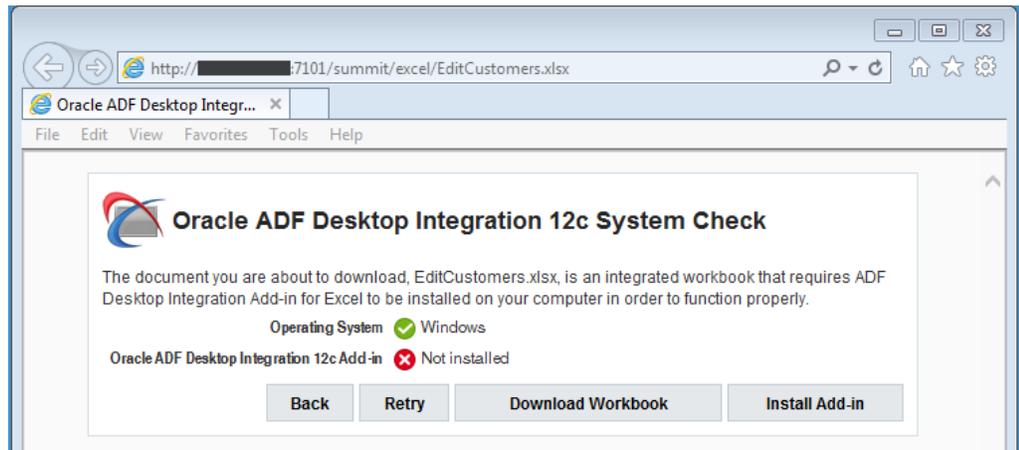
ADF Desktop Integration also provides connection failure reports to help diagnose the cause of connection failures from integrated Excel workbooks to Fusion web applications. For information, see [Troubleshooting Connection Problems to Fusion Web Applications](#).

Installing and Upgrading ADF Desktop Integration

End users must have the ADF Desktop Integration add-in installed on their Windows machine to use Excel workbooks that are integrated with Fusion web applications.

When an end user attempts to download an integrated Excel workbook from an ADF Desktop Integration-enabled Fusion web application, a Java applet verifies that the add-in is present on their machine. If the add-in is found, the workbook download begins automatically. Otherwise, ADF Desktop Integration prompts the end user to install the add-in, as shown in [Figure 17-1](#). If Java is not installed on the end user's machine or is disabled by the end user's security settings, the Java applet is unable to verify the presence of the ADF Desktop Integration add-in. ADF Desktop Integration informs the end user that the installation of the add-in cannot be verified. It presents the end user with the option to download the workbook and/or install the add-in.

An ADF Desktop Integration-enabled Fusion web application is a web application where you have added an integrated Excel workbook, as described in [Adding an Integrated Excel Workbook to a Fusion Web Application](#).

Figure 17-1 Dialog Prompting End Users to Install ADF Desktop Integration Add-in

You can control the display of this system check message (show or hide it) by setting the appropriate value for the `<param-name>SystemCheck.Enabled</param-name>` in the application's `web.xml` file. See [How to Manage the Display of the System Check to End Users](#).

If you disable the display of the system check message, you can make the ADF Desktop Integration installer available to end users to install, as described in [How to Install the ADF Desktop Integration Add-in From a Web Server](#).

 **Note:**

Installation of the ADF Desktop Integration add-in is specific to the current Windows user profile. If you have multiple Windows user profiles on a system, and you want to use ADF Desktop Integration integrated Excel workbooks in more than one of these user profiles, you must log in to each user profile and install the ADF Desktop Integration add-in.

When the ADF Desktop Integration installer runs, it verifies whether the required software is installed on the system. See [Prerequisites for Installing ADF Desktop Integration Add-in](#).

Post-installation, you or your end users can use the Client Health Check tool, described in [Running the Client Health Check Tool](#), to determine if the end user's environment is configured correctly. See also the "How to use ADF Desktop Integration Client Health Check Tool" document that you can retrieve from My Oracle Support (<https://support.oracle.com>) if you search for Doc ID 2010222.1.

Prerequisites for Installing ADF Desktop Integration Add-in

Before you install the ADF Desktop Integration add-in, make sure that you have the required Oracle ADF modules and third-party software installed and configured:

- Microsoft Windows

Microsoft Windows operating systems support the development and deployment of Excel workbooks that integrate with Fusion web applications. For 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/jdev-088164.html>

- Microsoft Excel

For 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/jdev-088164.html>

- 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.

The following software is required before ADF Desktop Integration add-in is installed. If this software is missing, the ADF Desktop Integration installer automatically downloads and installs it before installing the ADF Desktop Integration add-in.

- Microsoft .NET Framework 4.5.2

The Microsoft .NET Framework 4.5.2 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/>.

- 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/>.

How to Install the ADF Desktop Integration Add-in From a Web Server

You can make the ADF Desktop Integration installer available from the web server where your Fusion web application is running. The installer is embedded in the `oracle.adf.desktopintegration.war` file and can be downloaded by the end user from the ADF Desktop Integration-enabled Fusion web application.

For information about running the installer on a Windows system once it has been downloaded, see [Installing ADF Desktop Integration](#).

To download the installer from a web server:

1. Provide your end users with a URL in the following format that they can open in a web browser:

```
http://<hostname>:<portnumber>/<context-root>/adfdiRemoteServlet?  
excel-addin-installer
```

For example:

```
http://127.0.0.1:7101/summit/adfdiRemoteServlet?excel-addin-installer
```

2. Depending upon the browser, the end user will be prompted to download, or download and run, the `adfdi-excel-addin-installer.exe` installer file.



Note:

Making the ADF Desktop Integration add-in installer available for download only works if the Fusion web application is an ADF Desktop Integration-enabled Fusion web application. The developer of a Fusion web application can implicitly enable ADF Desktop Integration by adding an integrated Excel workbook to the Fusion web application, as described in [Adding an Integrated Excel Workbook to a Fusion Web Application](#), or explicitly, by configuring the application's `web.xml` file, as described in [ADF Desktop Integration Settings in the Web Application Deployment Descriptor](#). Adding an integrated Excel workbook to the Fusion web application may be preferable to configuring the `web.xml` file directly because JDeveloper makes many of the required configuration changes when the developer adds an integrated Excel workbook to the Fusion web application.

How to Upgrade the ADF Desktop Integration Add-in

An end user can upgrade ADF Desktop Integration in two ways.

- Run the ADF Desktop Integration installer to upgrade.

For information about downloading the installer, see [How to Install the ADF Desktop Integration Add-in From a Web Server](#).

- Open and run the integrated Excel workbook.

Each time the end user logs into the Fusion web application from an integrated Excel workbook, ADF Desktop Integration checks whether a new version of the add-in is available on the server. If a new version is available, the end user will be prompted to download the latest version of ADF Desktop Integration. If the end user accepts the prompt to install the latest version, the end user must also restart the Excel application after the installation completes for the change to take effect. For information, see [Verifying the Client Version of ADF Desktop Integration](#).

How to Run ADF Desktop Integration Installer from Command Line

The ADF Desktop Integration installer also supports optional command line switches. [Table 17-1](#) lists switches that you can specify with the installer executable file.

Table 17-1 ADF Desktop Integration Installer Command Line Switches

Switch	Description
/help	Displays a list of supported switches with description.

Table 17-1 (Cont.) ADF Desktop Integration Installer Command Line Switches

Switch	Description
/quiet	<p>Suppresses the interactive mode of the installer and does not install any missing prerequisite software.</p> <p>Before you install ADF Desktop Integration using the quiet mode, make sure that prerequisite software is installed on the end user's system. See Prerequisites for Installing ADF Desktop Integration Add-in.</p>
/designer <0 1>	<p>Installs the add-in with designer tools enabled.</p> <p>Application developers use the designer tools to configure integrated Excel workbooks. The designer tools are not intended for end users. For this reason, do not enable designer tools if installing the ADF Desktop Integration add-in for end users.</p> <p>By default, the ADF Desktop Integration add-in for end users is installed with designer tools disabled unless enabled during a previous installation.</p> <p>Use the following switch to disable the designer tools that have been enabled by a prior installation:</p> <p>/designer 0</p>
/log <path>	<p>Runs the installer and directs the log output to the specified log file. The default log file location is %TEMP%\adfdi-installer-log.txt. See ADF Desktop Integration Logs.</p>
/roaming <0 1>	<p>Use this switch as follows:</p> <ul style="list-style-type: none"> • 0 to install the add-in to the local application data folder (%localappdata%\Oracle\Oracle ADF Desktop Integration 12c Add-In for Excel). This is the default installation location. Use /roaming 0 to install to the local application data folder during an upgrade from a prior installation that was installed to the roaming application data folder. • 1 to install the add-in to the end user's roaming application data folder (%appdata%\Oracle\Oracle ADF Desktop Integration 12c Add-In for Excel).

How to Manage the Display of the System Check to End Users

ADF Desktop Integration-enabled web applications display a system check if the end user attempts to download an integrated Excel workbook.

The system check informs the end user if their system meets the requirements to use integrated Excel workbooks.

You can prevent the display of the system check if, for example, your end users do not have the permissions to download and execute the installer.

ADF Desktop Integration-enabled web applications created using this release and later of Oracle ADF display the system check by default. Applications created using prior releases of Oracle ADF do not. You can change this behavior using the appropriate value for the `SystemCheck.Enabled` parameter, as discussed in the following procedure.

To manage the display of the system check message to end users:

1. Open the `web.xml` file of your Fusion web application.
2. Add an initialization parameter to the `adfdiExcelDownload` filter, as described in the following table.

Table 17-2 Initialization Parameter Name-Value Pair

Property	Description
Name	Enter the name of the initialization parameter as follows: <code>SystemCheck.Enabled</code> Note that the initialization parameter name is case sensitive.
Value	Set to: <ul style="list-style-type: none"> • <code>False</code> so that no system check message appears. • <code>True</code> to display the system check message to end users where a link to download the ADF Desktop Integration add-in installer appears. ADF Desktop Integration interprets any value other than <code>True</code> as <code>False</code> . If you do not specify a value, no system check appears.

3. Save the `web.xml` file.
4. Rebuild and restart your Fusion web application.

The following example shows the entry that appears in an application's `web.xml` file when you disable the system check.

```
<filter>
  <filter-name>adfdiExcelDownload</filter-name>
  <filter-class>oracle.adf.desktopintegration.filter.DIExcelDownloadFilter</filter-
class>
  <init-param>
    <param-name>SystemCheck.Enabled</param-name>
    <param-value>False</param-value>
  </init-param>
</filter>
```

Application developers can exempt individual integrated Excel workbooks from the system check by passing the `skip_adfdi_check` parameter to the download URL that the Fusion web application page renders. For example, the following URL downloads the `EditCustomers.xlsx` workbook from the Summit sample application without displaying the system check to the end user:

```
http://127.0.0.1:7101/summit/excel/EditCustomers.xlsx?skip_adfdi_check
```

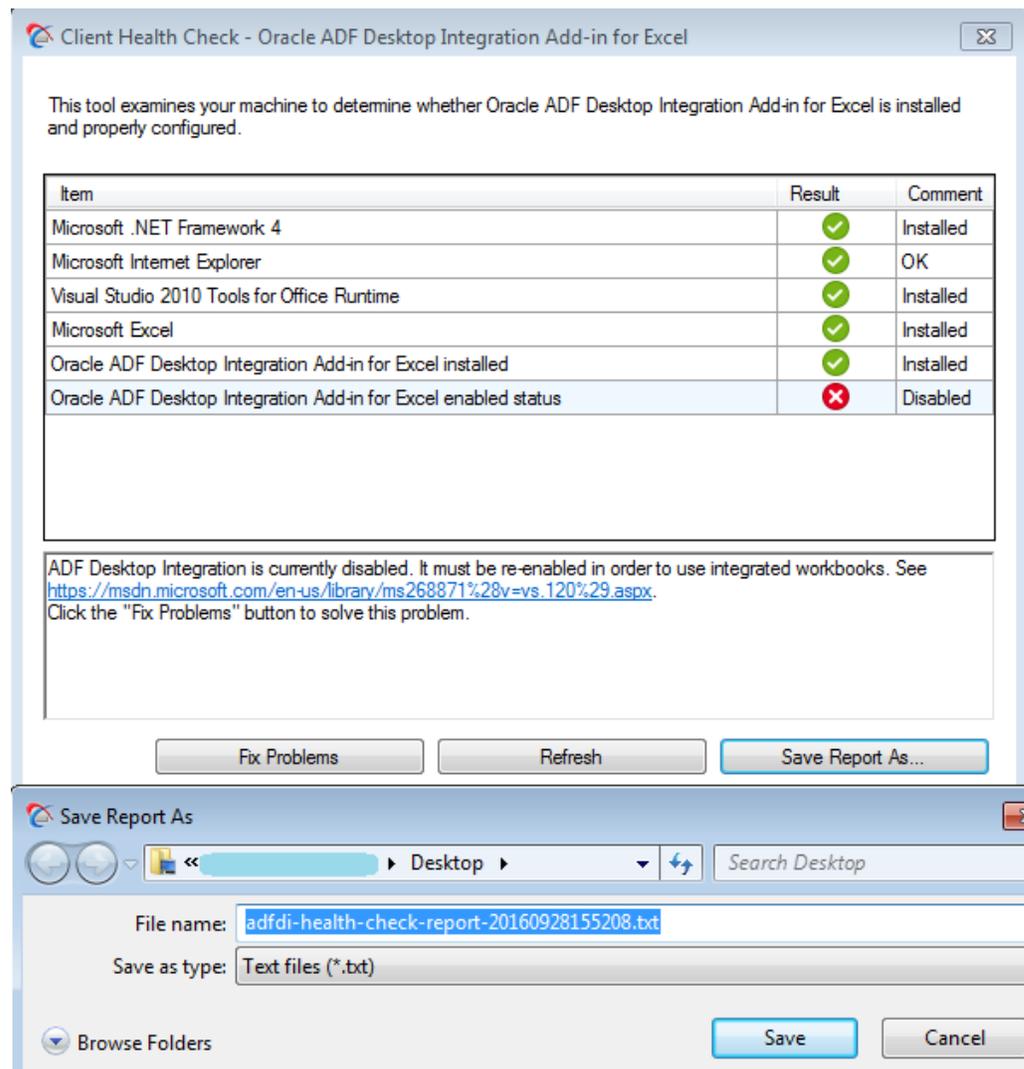
For information about passing parameters, see [Passing Parameter Values from a Fusion Web Application Page to a Workbook](#).

Running the Client Health Check Tool

Use the Client Health Check tool to determine whether an end user's environment is configured correctly to use integrated Excel workbooks and ADF Desktop Integration.

The Client Health Check tool is an executable (.EXE) that reviews the end user's environment, and, in some cases, offers the opportunity to fix problems it identifies. It also produces a report that end users can save to a location they choose, as shown in Figure 17-2.

Figure 17-2 Client Health Check Tool



End users can download the Client Health Check tool from an ADF Desktop Integration-enabled Fusion web application. The URL that end users download the Client Health Check tool from has the following format:

```
<protocol>://<hostname>:<portnumber>/<context-root>/adfdiRemoteServlet?excel-addin-health-check
```

End users can also download the tool by clicking the **Run client health check tool** link that appears in `<protocol>://<hostname>:<portnumber>/<context-root>/adfdiRemoteServlet`.

If, for example, the Summit sample application for ADF Desktop Integration runs on your machine, the Client Health Check tool can be downloaded from `http://127.0.0.1:7101/summit/adfdiRemoteServlet?excel-addin-health-check`.

Once the tool has been downloaded to the end user's machine, run it to determine if the ADF Desktop Integration add-in is installed and properly configured. Review the result of each item in the report to verify that it passes verification. Click any item that the tool flags as a problem to view additional information. Consider clicking the **Fix Problems** button so that the tool attempts to resolve identified problems. If the tool resolves all problems, quit the tool.

The tool also produces a report that you can save to the end user's machine by clicking the **Save Report As** button. This report contains technical information that may assist in resolving issues that the tool did not resolve using the **Fix Problems** button. Attach the report that the tool produces with any request that you submit for technical support.

See also the document that you can retrieve from My Oracle Support (<https://support.oracle.com>) if you search for Doc ID 2010222.1.

ADF Desktop Integration Logs

ADF Desktop Integration generates log files during installation and in response to various client and server activity.

See the "How To Obtain Log Files For ADF Desktop Integration" document that you can retrieve from My Oracle Support (<https://support.oracle.com>) if you search for Doc ID 2012985.1.

Installation Log File

The default location of the ADF Desktop Integration installation log file is `%TEMP%\adfdi-installer-log.txt`. For example, `C:\Users\UserID\AppData\Local\Temp\adfdi-installer-log.txt`. You can redirect the location of the install log file using the `/log <path>` command-line switch described in [How to Run ADF Desktop Integration Installer from Command Line](#).

Server-side Log Files

You configure the generation of server-side log files for ADF Desktop Integration the same way as for other Oracle ADF modules. For information, see [About Server-Side Logging](#).

For more general information about logging in an Oracle Fusion Middleware environment, see *Managing Log Files and Diagnostic Data in Administering Oracle Fusion Middleware*.

Client-side Log Files

ADF Desktop Integration, by default, enables logging at an `Information` level on the client. This level of logging is always on. Verbose logging can be enabled for one user session using a menu that ADF Desktop Integration adds to integrated Excel

workbooks. A log level different to the `Information` level can be configured for logging that spans multiple user sessions. For information, see [About Client-Side Logging](#).

See also the documents that you can retrieve from My Oracle Support (<https://support.oracle.com>) if you search for Doc IDs 2094378.1 and 2094434.1.

Diagnostic Reports

Diagnostic reports collect information about the end user environment where the ADF Desktop Integration add-in is installed. This report can be generated at any time by the end user. For information about how an end user can generate a diagnostic report, see [Generating ADF Desktop Integration Diagnostic Reports](#). See also the “ADFDi Diagnostic Report” document that you can retrieve from My Oracle Support (<https://support.oracle.com>) if you search for Doc ID 2012576.1.

Security in ADF Desktop Integration

If your Fusion web application enforces authentication, the integrated Excel workbooks help the end user authenticate properly before data transfer happens between the workbooks and application.

For information, see [About Security In Your Integrated Excel Workbook](#).

End User Authentication

If end users are not prompted for user credentials while using integrated Excel workbooks and interacting with a secure Fusion web application, you need to investigate the security configuration of the Fusion web application. For information, see [Verifying End-User Authentication for Integrated Excel Workbooks](#).

End users who have difficulty connecting to a Fusion web application may see the Connection Failure dialog shown in [Figure C-3](#). Ask these users to save the connection failure report by clicking the **Save Report** button in the dialog. The report contains diagnostic information that may help resolve the connection failure. As an administrator, you may want to review the connection failure report for clues to solving the problem. See the documents that you can retrieve from My Oracle Support (<https://support.oracle.com>) if you search for Doc IDs 2014348.1 and 2094772.1.

For more information about ADF Desktop Integration security, see the “*Oracle ADF Desktop Integration Security whitepaper*” on OTN at:

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

What You May Need to Know About Configuring Security in a Fusion Web Application

Note the following points before you secure your application:

- In order for the end-user login sequence to complete successfully, the authentication provider must redirect the browser back to the originally requested ADF Desktop Integration servlet URL after a successful login.
- For applications running in an environment using Oracle Access Manager, the system administrator should make sure that the URL for the ADF Desktop

Integration Remote servlet is configured as a protected resource for Oracle Access Manager.

For information, see *Introducing Oracle Access Management in Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.

- Make sure that applications using ADF Desktop Integration have a security constraint configured in `web.xml` that protects the ADF Desktop Integration remote servlet.

The following code extract from `web.xml` shows an example security constraint protecting the remote servlet:

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>adfdiRemote</web-resource-name>
    <url-pattern>/adfdiRemoteServlet</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>valid-users</role-name>
  </auth-constraint>
</security-constraint>
```

- When using Oracle WebGate and a SSL URL to access the Fusion web application (such as `https:// . . .`) it may be necessary to configure WebGate's `mod_wl_ohs.conf` configuration file as follows:

```
<IfModule mod_weblogic.c>
  WLProxySSLPassThrough ON
  WLProxySSL ON
  MatchExpression /TestApp
  WebLogicHost=test.host.com|WebLogicPort=7101|
</IfModule>
```

where `/TestApp` is the context root of your application, `test.host.com` is the host name and domain, and `7101` is the port number for the web application.

- When opening an integrated Excel workbook, or any Microsoft Office document, directly (without downloading the file) from a link in the Fusion web application, the Windows Login dialog may appear twice asking for user credentials. This happens because Microsoft Office sends its own authentication request to the web server, making the Login dialog appear twice. End users may click **Cancel** and ignore the first authentication request.
- Applications secured via a digital certificate where clients use `https` URLs to access the application should make sure that the certificate is valid. Valid certificates have host names that match the host to which they are deployed, have not expired, and have a valid path to a trusted issuing authority. In the case where the certificate is invalid, the client will be prompted during login to accept the invalid certificate.
- ADF uses chunked encoding for some requests to the server. If you have any network devices between Excel and the web application server configured to block requests that do not contain a content length header, you should configure them to allow chunked encoding (no content length header). Some network devices such as content caching servers may have a default configuration that blocks requests with no content length header. For information, see the "ADFDI-07528 WebException During TamperCheck" document that you can retrieve from My Oracle Support (<https://support.oracle.com>) if you search for Doc ID 2013517.1.

- Before you secure your application, note that the HTTPS communication that the ADF Desktop Integration add-in initiates during the login sequence requires a successful SSL/TLS protocol handshake. This handshake can fail if the server and ADF Desktop Integration add-in cannot agree on a protocol to use. For example, if the client computer supports SSL 3.0 and TLS 1.0, but the server only supports TSL 1.1 and TLS 1.2. The ADF Desktop Integration add-in makes HTTPS connections using portions of the Microsoft Internet Explorer technology stack as well as the Microsoft .NET Framework. For best results, ensure that:
 - Client computers have Microsoft .NET Framework 4.5.2 (or higher) installed
 - Microsoft Internet Explorer is configured on the client computer to support TLS 1.1 and TLS 1.2. For information, see the "ADFDI-00500: unable to execute Wininet method HttpSendRequest; error code: 12029" During Login" document that you can retrieve from My Oracle Support (<https://support.oracle.com>) if you search for Doc ID 2025331.1.
 - Ensure that the Oracle WebLogic Server that hosts the web application is configured to support TLS 1.1 and TLS 1.2. For information, see "Specifying the SSL Protocol Version" in the Oracle WebLogic Server security guide for the release of Oracle WebLogic Server that you use.

The ADF Desktop Integration add-in may initiate HTTPS communication by offering the TLS 1.2 protocol for the SSL/TLS protocol handshake. While ADF Desktop Integration does not require TLS 1.2, servers (such as Oracle WebLogic Server, Oracle HTTP Server, and Oracle Application Server WebCache) must successfully negotiate a mutually agreed-upon protocol when offered the TLS 1.2 protocol. Some older versions of servers are known to reject TLS 1.2 offers rather than negotiate to use a lower version (for example, TLS 1.0). Such server versions are not supported. If older versions of server are in use, please make sure that the [most recent Critical Patch Updates](#) are applied. See also the "WebException: The request was aborted: Could not create SSL/TLS secure channel - during or right after ADFdi login sequence" document that you can retrieve from My Oracle Support (<https://support.oracle.com>) if you search for Doc ID 2087746.1.

For information about securing integrated Excel workbooks, see [What You May Need to Know About Securing an Integrated Excel Workbook](#).

What You May Need to Know About Resource Grants for Web Pages

In an integrated Excel workbook, each worksheet is bound to a specific page definition. Users' access to pages may be controlled by resource grants. If an end user is not authorized to work with a page definition, ADF Desktop Integration disallows all data transactions in worksheets bound to that page definition, displays a failure message, and disables those integrated worksheets. The end user can alter any existing data in the worksheet, but cannot download or upload it. The tracking of changes in ADF Table components is also disabled. The end user can continue to use ADF Desktop Integration features in other worksheets in the same workbook, provided those worksheets are bound to page definitions that the end user is authorized to work with.

The worksheet is re-enabled when the end user reopens the workbook and establishes a new session, provided that the end user has obtained the necessary resource grants for the corresponding page definition.

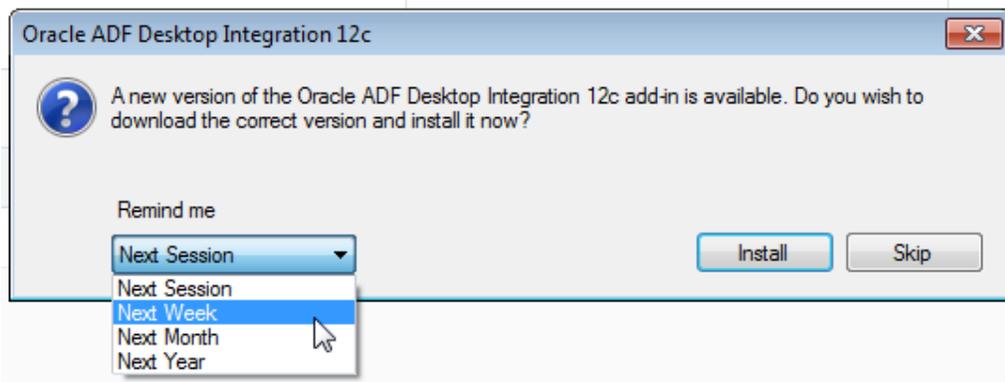
For information about securing your Fusion web application, see Enabling ADF Security in a Fusion Web Application in *Developing Fusion Web Applications with Oracle Application Development Framework*.

Verifying the Client Version of ADF Desktop Integration

ADF Desktop Integration verifies whether a new version of the add-in is available on the server each time that an end user establishes a session with the Fusion web application from the runtime integrated Excel workbook.

If a new version is available, ADF Desktop Integration displays the dialog shown in [Figure 17-3](#). If a new version is not available, no dialog appears to the end users.

Figure 17-3 Client-Server Version Check Dialog

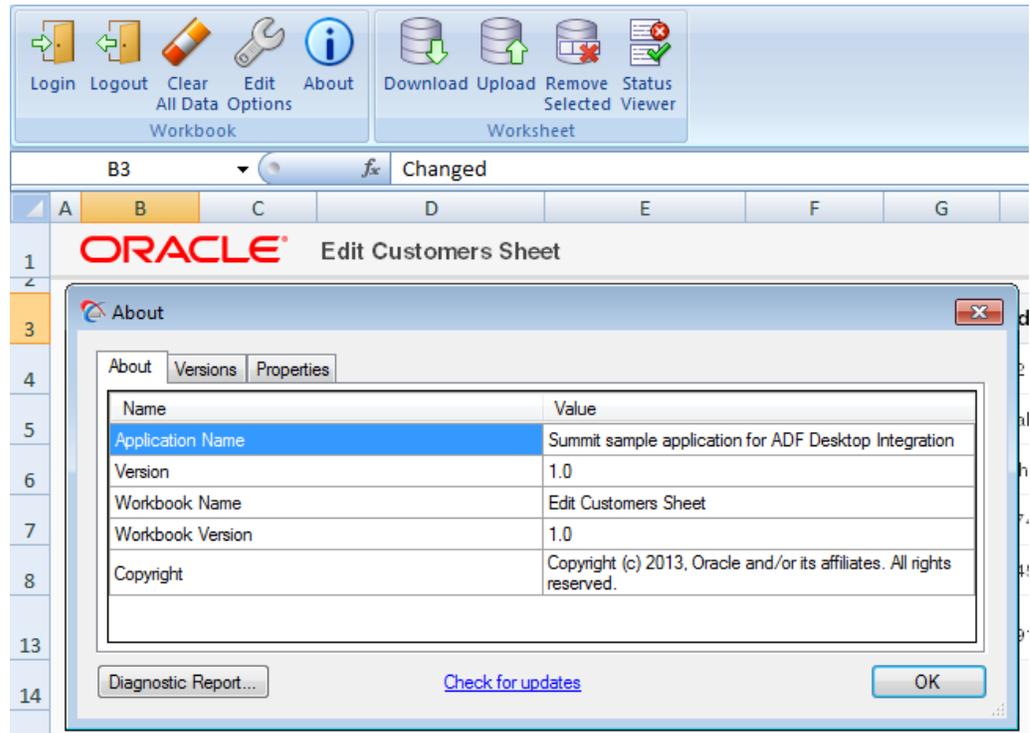


If the end user clicks:

- **Install:** ADF Desktop Integration initiates the download of the installer from the server to update the client to the matching server version
- **Skip:** ADF Desktop Integration attempts to continue to function normally. The dialog appears the next time the end user starts a session with the integrated Excel workbook that connects to the Fusion web application unless the end user chooses a later reminder time from the **Remind Me** dropdown list, as shown in [Figure 17-3](#).

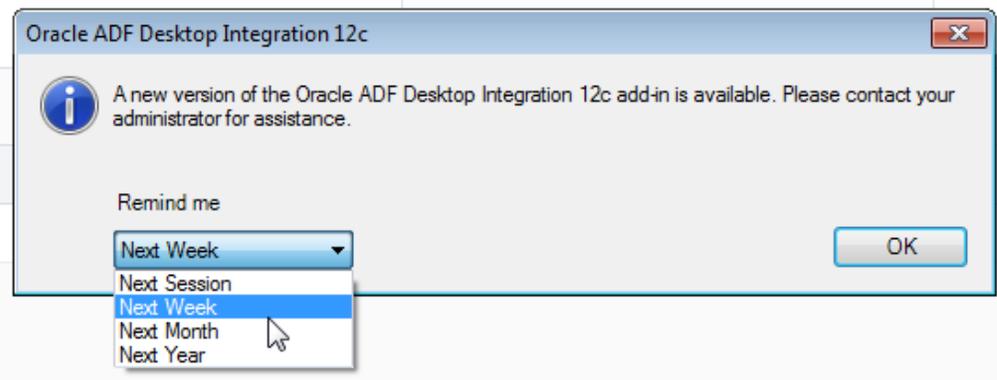
Always using a client version that matches the server version is highly recommended to avoid unexpected behavior or errors. If end users skip the installation of a newer client version of ADF Desktop Integration, they can install at a later time by clicking the **Check for updates** link that appears in the About dialog of the integrated Excel workbook, as shown in [Figure 17-4](#).

Figure 17-4 Check for Updates Link for End Users



For scenarios where you do not want end users to install a newer client version or they cannot because they do not have the required privileges to install software on their machines, the default behavior where ADF Desktop Integration displays an option to install a newer version can be disabled. When you disable the option to install a newer client version, the Client-Server Version Check dialog appears and informs the end user of the mismatch, but does not present an option to install a newer version. [Figure 17-5](#) shows this dialog. Furthermore, the About dialog shown in [Figure 17-4](#) will no longer have a **Check for updates** link to start an install process. For information about how to disable the option to upgrade, see [How to Disable the Install Option on the Client-Server Version Check Dialog](#).

Figure 17-5 Client-Server Version Check Dialog without Install Option



Note that:

- ADF Desktop Integration performs the client-server version verification every time that the integrated Excel workbook establishes a session with the Fusion web application.
- The version of ADF Desktop Integration running on the server can change at any time (for example, server upgrade), but ADF Desktop Integration only performs the client-server version verification when the user session is re-established.
- Consider employing other mechanisms for situations where end users cannot install a version that matches the server version. For example, automatically push out software updates from a centrally-managed IT source to make sure that the matching version of the client software is installed.

How to Disable the Install Option on the Client-Server Version Check Dialog

By default, ADF Desktop Integration displays an option to end users to install a newer client version from the Client-Server Version check dialog. You can disable this option so that ADF Desktop Integration informs end users of the mismatch, as shown in [Figure 17-5](#), but does not permit end users to install a newer client version. Disabling this option is not recommended. However, it may be useful in cases where end users do not have permission to install software.

Before you begin:

It may be helpful to have an understanding of how ADF Desktop Integration checks whether a new version of the client is available from the server. See [Verifying the Client Version of ADF Desktop Integration](#).

To disable the install option on the Client-Server Version Check dialog:

1. Open the `web.xml` file of your Fusion web application.
2. Add an initialization parameter to the `adfdiRemote` servlet to disable the option to install from the Client-Server Version Check dialog, as described in [Table 17-3](#).

Table 17-3 Disabling the install option on the Client-Server Version Check dialog

Property	Value
Name	Enter the name of the initialization parameter as follows: <code>ClientUpgradePrompt.Enabled</code> Note that the name is case-sensitive.
Value	Set the value of <code>ClientUpgradePrompt.Enabled</code> to <code>False</code> . Note that any value other than <code>False</code> will be interpreted as <code>True</code> .

3. Save the `web.xml` file.

The `web.xml` file of your Fusion web application will now have the `ClientUpgradePrompt.Enabled` entry:

```
<servlet>
  <servlet-name>adfdiRemote</servlet-name>
```

```

<servlet-class>oracle.adf.desktopintegration.servlet.DIRemoteServlet</servlet-class>
<init-param>
  <param-name>ClientUpgradePrompt.Enabled</param-name>
  <param-value>False</param-value>
</init-param>
</servlet>

```

4. Restart your Fusion web application.

Verifying Integrated Excel Workbook Metadata

To give end users the confidence that the workbook configuration has not been altered maliciously, ADF Desktop Integration verifies the integrity of the workbook metadata automatically using the *Tamper-Check* feature.

For information, see [Checking the Integrity of an Integrated Excel Workbook's Metadata](#).

How to Disable the Metadata Tamper-Check in the Fusion Web Application

By default, ADF Desktop Integration verifies that the workbook configuration metadata is not tampered with after the workbook's developer publishes the Excel workbook for end users. You can disable the metadata tamper-check by configuring a parameter in the deployment descriptor file (`web.xml`) of the Fusion web application.

Before you begin:

It may be helpful to have an understanding of how ADF Desktop Integration verifies the integrity of an integrated Excel workbook's metadata. See [Verifying Integrated Excel Workbook Metadata](#).

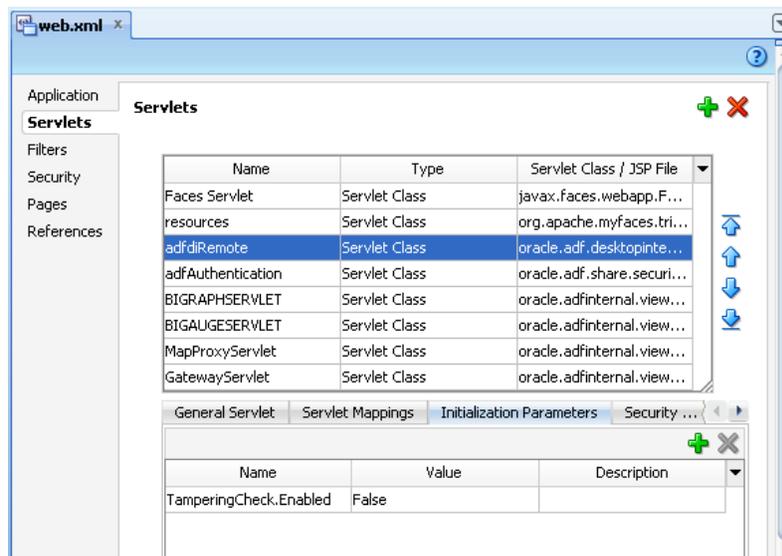
To disable the metadata tamper-check in the Fusion web application:

1. Open the `web.xml` file of your Fusion web application.
2. Add an initialization parameter to the `adfdiRemote` servlet to disable the metadata tamper-check, as described in [Table 17-4](#).

Table 17-4 Disabling Metadata Tamper-Check

Property	Value
Name	Enter the name of the initialization parameter as follows: <code>TamperingCheck.Enabled</code> Note that the name is case-sensitive.
Value	Set the value of <code>TamperingCheck.Enabled</code> to <code>False</code> . Note that any value other than <code>False</code> will be interpreted as <code>True</code> .

[Figure 17-6](#) shows the `web.xml` editor in JDeveloper.

Figure 17-6 Disabling the Metadata Tamper Check In JDeveloper

3. Save the web.xml file.

The web.xml file of your Fusion web application will now have the TamperingCheck.Enabled entry:

```
<servlet>
  <servlet-name>adfdiRemote</servlet-name>
  <servlet-class>...</servlet-class>
  <init-param>
    <param-name>TamperingCheck.Enabled</param-name>
    <param-value>False</param-value>
  </init-param>
</servlet>
```

4. Restart your Fusion web application.

If the TamperingCheck.Enabled parameter is not present in web.xml, tamper check is enabled. For information about the web.xml file, see [ADF Desktop Integration Settings in the Web Application Deployment Descriptor](#).

Common ADF Desktop Integration Error Messages and Problems

While using or configuring the ADF Desktop Integration-enabled Fusion web application or workbooks, end users might see error messages or encounter problems. The following information may assist you in resolving these problems for your end users:

- The "Troubleshooting Oracle ADF Desktop Integration" document that you can retrieve from My Oracle Support (<https://support.oracle.com>) if you search for Doc ID 2012600.2.
- ADFDI-00100 to ADFDI-55516 in *Oracle Fusion Middleware Error Messages*.

ADF Desktop Integration also provides the Client Health Check tool that determines if an end user's machine is configured correctly to use integrated Excel workbooks. Ask end users who encounter problems using integrated Excel workbooks to download

and run this tool, as described in [Providing Diagnostic and Logging Information to Technical Support](#).

A

ADF Desktop Integration Component Properties and Actions

Lists and describes the properties of ADF Desktop Integration components. It also describes the actions that certain components expose.

This appendix includes the following sections:

- [Frequently Used Properties in the ADF Desktop Integration](#)
- [ADF Input Text Component Properties](#)
- [ADF Output Text Component Properties](#)
- [ADF Label Component Properties](#)
- [ADF List of Values Component Properties](#)
- [ADF Image Component Properties](#)
- [ADF Input Date Component Properties](#)
- [ModelDrivenColumnComponent Subcomponent Properties](#)
- [TreeNodeList Subcomponent Properties](#)
- [ADF Table Component Properties and Actions](#)
- [ADF Read-only Table Component Properties and Actions](#)
- [Action Set Properties](#)
- [Workbook Actions and Properties](#)
- [Worksheet Actions and Properties](#)
- [ADF Desktop Integration Compatibility Properties](#)

Frequently Used Properties in the ADF Desktop Integration

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 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.

Table A-1 (Cont.) Frequently Used Properties in ADF Desktop Integration

Name	Type	EL	Description
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 ADF Desktop Integration EL Expressions . For information about using labels, see 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	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. The cells can also be locked if this setting is used in combination with automatic worksheet protection, as described in Using Worksheet Protection . 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 Working with Styles .
StyleName	String	Y	Specifies the style in the current Excel workbook to apply when the Oracle ADF component is rendered. For information, see 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 information, see Displaying Tooltips in ADF Desktop Integration Components .
Value	Varies	Y	This property typically references an EL binding value expression that gets evaluated during the invocation of the ADF Table component's <code>Download</code> and <code>RowDownSync</code> actions or a worksheet's <code>DownSync</code> action. The resulting data value gets displayed in the worksheet at runtime.

Many label-type properties are optional and default to empty. At runtime, if the value of such properties 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.

ADF Input Text Component Properties

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 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 .

ADF Output Text Component Properties

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 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 .

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 [Using Labels in an Integrated Excel Workbook](#).

[Table A-4](#) lists alphabetically the properties that the ADF Label component references.

Table A-4 ADF Label Component Properties

Name	Description
Annotation	For information about this property, see Table A-1 .

Table A-4 (Cont.) ADF Label Component Properties

Name	Description
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 .

ADF List of Values Component Properties

Lists the properties of the ADF List of Values component.

For information about creating lists of values in your integrated Excel workbook, see [Working with Lists of Values](#).

Table A-5 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 information about dependent list of values, see 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 .

ADF Image Component Properties

Lists the properties of the ADF Image component which displays an image at runtime.

For information about adding an ADF Image component, see [Inserting an ADF Image Component](#).

Table A-6 ADF Label Component Properties

Name	Description
Source	Enter the absolute, or relative, URL of the image file.

Table A-6 (Cont.) ADF Label Component Properties

Name	Description
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 .

ADF Input Date Component Properties

Lists alphabetically the properties of the ADF Input Date component.

For information about the ADF Input Date component, see [Inserting an ADF Input Date Component](#).

Table A-7 ADF Input Date Component Properties

Name	Description
Annotation	For information about this property, see Table A-1 .
ComponentID	For information about this property, see Table A-1 .
InputDate.ReadOnly	For information about this property, see Table A-1 .
InputDate.Value	Specify an EL expression that resolves to a date-time value at runtime. For information about ADF Input Date component, see Inserting an ADF Input Date Component .
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 .

ModelDrivenColumnComponent Subcomponent Properties

Describes the properties that you configure for the ModelDrivenColumnComponent subcomponent.

The ModelDrivenColumnComponent 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 [ADF Table Component Column Properties](#).

The column subcomponent type is determined at runtime by the column's attribute Control Type hint specified on the server. For example, if there is a model-driven list associated with the attribute, then the column uses a dropdown list containing the model-driven list items at runtime. See [Adding a ModelDrivenColumnComponent Subcomponent to Your ADF Table Component](#).

Table A-8 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 Action Set Properties .
ReadOnly	Boolean	Y	<p>Set the ReadOnly property to <code>False</code> if you do want users to edit the values in the column, set to <code>True</code> otherwise. The default value is <code>False</code>.</p> <p>If you create the ADF Table component by double-clicking a tree binding in the Bindings palette, the property's value is set to an EL expression in the following format that evaluates to <code>True</code> or <code>False</code> at runtime:</p> <pre>#{bindings.{tree-id}.hints.{attr-id}.readOnly}</pre> <p>For example,</p> <pre>#{bindings.Customers.hints.Address.readOnly}</pre> <p>For information about this property, see Table A-1.</p>
Value	Varies	Y	For information about this property, see Table A-1 .

TreeNodeList Subcomponent Properties

Describes the properties that you configure for the `TreeNodeList` subcomponent.



Note:

The `ModelDrivenColumnComponent` subcomponent also renders dropdown menus for tree binding attributes that have a model-driven list. Consider using a `ModelDrivenColumnComponent` subcomponent rather than a `TreeNodeList` subcomponent. See [ModelDrivenColumnComponent 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. For information about creating lists of values in your integrated Excel workbook, see [Working with Lists of Values](#) .

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 [ADF Table Component Column Properties](#).

Table A-9 TreeNodeList Subcomponent Properties

Name	Type	EL	Description
DependsOnList	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 information about dependent list of values, see Creating Dependent Lists of Values in an Integrated Excel Workbook .
List	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.
ReadOnly	Boolean	Y	Always set this property's value to <code>True</code> because it is obsolete when used with this subcomponent. For read-only columns, consider using the <code>ModelDrivenColumnComponent</code> subcomponent or the <code>OutputText</code> component.

ADF Table Component Properties and Actions

The ADF Table component uses the properties and component actions listed here.

ADF Table Component Properties

[Table A-10](#) lists alphabetically the properties the ADF Table component uses.

Table A-10 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 information about how you use these properties, see 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.
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.

Table A-10 (Cont.) ADF Table Component Properties

Name	Type	EL	Description
<code>BatchOptions.StartBatchActionID</code>	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.
<code>DisplayUploadOptions</code>	Boolean	N	Set to <code>True</code> to display the Upload Options dialog when uploading data from ADF Table component. See What You May Need to Know About Upload Options .
<code>Columns</code>			An array of columns. For information about the properties that each column in the array supports, see ADF Table Component Column Properties .
<code>ComponentID</code>			For information about this property, see Table A-1 .
<code>Position</code>			For information about this property, see Table A-1 .
<code>ResizeColumnsMode</code>			Controls whether and how the columns in the entire table are resized. See Configuring an ADF Table Component to Resize Columns Based on Data at Runtime .
<code>RowActions</code>			This group of properties allows you specify which actions are enabled and can be invoked.
<code>RowActions.AutoConvertNewRowsEnabled</code>	Boolean	N	When <code>True</code> , end users can insert new data from non-integrated Excel workbooks directly into the row under the ADF Table component or edit the row under the ADF Table component to convert it to a row in the ADF Table component. See How to Insert or Paste Rows in an ADF Table Component . <code>True</code> is the default value. Set to <code>False</code> for ADF Table components that do not support row inserts or that need to have a calculated row under the table.
<code>RowActions.AutoRefreshRowActionID</code>	Method action binding	N	Specify a method action binding to invoke when an end user triggers a row refresh by editing a cell in a column that has its <code>TriggersRowRefresh</code> property set to <code>True</code> . You must also set the ADF Table component's <code>RowActions.AutoRefreshRowEnabled</code> property to <code>True</code> to enable this functionality. By default, no value is set for this property. See Configuring Automatic Row Refresh in an ADF Table Component .
<code>RowActions.AutoRefreshRowEnabled</code>	Boolean	N	When <code>True</code> , end users can trigger a row refresh by editing a cell in a column that has its <code>TriggersRowRefresh</code> property set to <code>True</code> . The default value for this property is <code>False</code> . See Configuring Automatic Row Refresh in an ADF Table Component .

Table A-10 (Cont.) ADF Table Component Properties

Name	Type	EL	Description
<code>RowActions.DeleteRowActionID</code>	Action binding	N	Specify an action binding to invoke for each row flagged for deletion. A value for this property is optional.
<code>RowActions.DeleteRowEnabled</code>	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.
<code>RowActions.FailureActionID</code>	Action binding	N	Specify an action binding to invoke if failures occur during the processing of rows. A value for this property is optional.
<code>RowActions.InsertAfterRowActionID</code>	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.
<code>RowActions.InsertBeforeRowActionID</code>	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.
<code>RowActions.InsertRowEnabled</code>	Boolean	N	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. If you set this property to <code>True</code> , you must specify a value for <code>RowActions.InsertBeforeRowActionID</code> . 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> . For information about inserting rows in an ADF Table component, see Inserting Data in an ADF Table Component .
<code>RowActions.InsertRowsAfterUploadEnabled</code>	Boolean	N	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. The property is ignored if <code>InsertRowEnabled</code> is set to <code>False</code> .
<code>RowActions.InsertTempRowActionID</code>	Action binding	N	When configured, this action is invoked to create a temporary row during row-level action set execution for insert rows. See Using Row-Level Action Sets in a Table Column .

Table A-10 (Cont.) ADF Table Component Properties

Name	Type	EL	Description
<code>RowActions.UpdateRowActionID</code>	Action binding	N	Specify an action binding to invoke for each row updated. A value for this property is optional.
<code>RowActions.UpdateRowEnabled</code>	Boolean	N	Set to <code>True</code> to allow a user update an existing row. <code>True</code> is the default value. A value for this property is required.
<code>RowData</code>			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. 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.
<code>RowData.CachedAttributes</code>	Array	N	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: <ul style="list-style-type: none"> • <code>Value</code> Select the tree binding attribute for which data is to be cached. • <code>Annotation</code> For information about this property, see Table A-1. <p>The table <code>RowDownSync</code> action caches the row attribute values for the configured <code>RowData.CachedAttributes</code>. The table <code>RowUpSync</code>, <code>Upload</code>, and <code>UploadAllOrNothing</code> actions send any cached row attribute values to the Fusion web application.</p> <p>Note: A cached attribute value will override any edits the end user makes to the same attribute binding exposed in a column component. Therefore, you should not configure the same attribute in <code>CachedAttributes</code> and in a table column component.</p> <p>Cached attributes are not included in copy-paste operations that users perform with rows in an ADF Table component.</p> <p>For information about using the <code>RowData.CachedAttributes</code> array to cache data in an ADF Table component, see Adding a Model-Driven List Picker to an ADF Table Component and How to Add a Custom Popup Picker Dialog to an ADF Table Column.</p>

Table A-10 (Cont.) ADF Table Component Properties

Name	Type	EL	Description
RowData.ChangeIndicatorAttributeID	Attribute Binding	Y	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. See Handling Data Conflicts When Uploading Data from a Workbook .
RowLimit			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. See Limiting the Number of Rows Your Table-Type Component Downloads .
RowLimit.Enabled	Boolean	N	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. A value for this property is required.
RowLimit.MaxRows	Integer	Y	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. 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.

Table A-10 (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 RowLimit.MaxRows. This expression is evaluated each time the Table's Download 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 MaxRows value.</p> <p>For information about resource bundles, see 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 information about the page definition requirements for an integrated Excel workbook, see Table 4-1.</p>

Table A-10 (Cont.) ADF Table Component Properties

Name	Type	EL	Description
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 key attribute.</p> <p>This value is optional when either of the following conditions is true:</p> <ul style="list-style-type: none"> • Rows in the tree binding iterator expose row keys, in which case row key values are used to position the tree binding iterator.

 **Note:**

Row keys are available whenever the tree is bound to a view object that exposes either single-attribute or multi-attribute keys.

- The ADF Table component is configured to be insert-only (`RowActions.InsertRowEnabled` is set to `True` and `RowActions.UpdateRowEnabled` is set `False`).

ADF Table Component Column Properties

[Table A-11](#) describes the properties that a column in the `TableColumn` array can use.

Table A-11 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 information about dynamic columns, see 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.

Table A-11 (Cont.) ADF Table Component Column Properties

Name	Type	EL	Description
GroupedHeader			Configure the GroupHeader properties to group together columns that render in an ADF Table component by displaying an extra table header row above the ADF Table component's regular table header row at runtime. See Grouping Columns Together in an ADF Table Component .
GroupedHeader.Boundary	String	Y	Set to <code>start</code> or <code>end</code> to specify a column as the start or end column in a grouped header. Write an EL expression that evaluates to <code>start</code> or <code>end</code> if you want to create a grouped header for dynamic columns.
GroupedHeader.Label	String	Y	Write an EL expression that, when evaluated at runtime, displays a label in the grouped header.
GroupedHeader.StyleName	String	Y	Write an EL expression that resolves to an Excel style name that is applied to each cell in the grouped header.
GroupedHeader.Tooltip			For information about the tooltip property, see Table A-1 .
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 Table component. The following IDs are reserved to the default columns in the ADF Table component: <ul style="list-style-type: none"> • <code>_ADF_ChangedColumn</code> • <code>_ADF_FlagColumn</code> • <code>_ADF_RowKeyColumn</code> • <code>_ADF_StatusColumn</code> For information about these columns, see Special Columns in the ADF Table Component .

Table A-11 (Cont.) ADF Table Component Column Properties

Name	Type	EL	Description
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"> • ModelDrivenColumnComponent For information about the properties that this component supports, see ModelDrivenColumnComponent Subcomponent Properties. • InputDate component For information about the properties that this component supports, see ADF Input Date Component Properties. • Input Text component For information about the properties that this component supports, see ADF Input Text Component Properties. • Output Text component For information about the properties that this component supports, see ADF Output Text Component 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. Typically, you will rarely use the <code>InsertComponent</code> property.</p>
InsertUsesUpdate	Boolean	N	<p>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.</p> <p>The default value is <code>True</code>.</p>
ResizeMode			<p>Specifies how ADF Desktop Integration changes the column width at runtime when the <code>ResizeColumns</code> action is invoked.</p> <p>See Configuring an ADF Table Component to Resize Columns Based on Data at Runtime.</p>
TriggersRowRefresh	Boolean	Y	<p>Set to <code>True</code> or write an EL expression that resolves to <code>True</code> so that an end user editing a row cell in this column automatically triggers a row refresh. You must also set the ADF Table component's <code>RowActions.AutoRefreshRowEnabled</code> property to <code>True</code> to enable this functionality.</p> <p>By default, no value is set for this property. This equals <code>False</code>.</p> <p>See Configuring Automatic Row Refresh in an ADF Table Component.</p>

Table A-11 (Cont.) ADF Table Component Column Properties

Name	Type	EL	Description
UpdateComponent	ADF component	N	<p>Specifies the properties of the component that represents the binding for update and download operations. This component can be one of the following:</p> <ul style="list-style-type: none"> ModelDrivenColumnComponent For information about the properties that this component supports, see ModelDrivenColumnComponent Subcomponent Properties. Input Date component For information about the properties that this component supports, see ADF Input Date Component Properties. Input Text component For information about the properties that this component supports, see ADF Input Text Component Properties. Output Text component For information about the properties that this component supports, see ADF Output Text Component Properties.
Visible	Boolean	Y	<p>Write an EL expression that resolves to True or False. If True, the column appears in the ADF Table component. If False, the column does not appear. True is the default value.</p> <p>If you make a column dynamic, the ADF Table component ignores the value of the Visible property. For information about dynamic columns, see Adding a Dynamic Column to Your ADF Table Component.</p>
Width	Double	Y	<p>Specify the width of the column in number of characters. You can specify an EL expression that evaluates to a number or a literal numerical value to determine the width of the column. The value can be a fractional value using a decimal point. A character is the unit of the width. The value is used when ResizeMode is SpecifiedWidth.</p> <p>See Configuring an ADF Table Component to Resize Columns Based on Data at Runtime.</p>

ADF Table Component Actions

[Table A-12](#) describes the component actions available for use with the ADF Table component.

Table A-12 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 DoubleClickActionSet in an ADF Table component's column should invoke this action.

Table A-12 (Cont.) ADF Table Component Actions

Component Action	Description
DeleteFlaggedRows	<p>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.</p> <p>The <code>Actions' Options.AbortOnFailure</code> property lets you determine if the <code>DeleteFlaggedRow</code> component action continues execution after it encounters an error.</p> <p>See Deleting ADF Table Component Rows in the Fusion Web Application.</p>
DisplayRowErrors	<p>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 Special Columns in the ADF Table Component is configured with an action set that invokes this action.</p>
DisplayTableErrors	<p>Displays a list of errors that occurred during batch processing. This action is deprecated. It is no longer necessary. All relevant error messages can be viewed using the Status Viewer described in Using the Status Viewer to Report Error Messages to End Users.</p>
Download	<p>Download the rows corresponding to the current state of <code>TreeID</code>. For information about <code>TreeID</code>, see ADF Table Component Properties.</p>
DownloadFlaggedRows	<p>Downloads the flagged rows from the tree binding specified by <code>TreeID</code>. For information about <code>TreeID</code>, see Table A-10.</p> <p>This action applies to the downloaded rows only, and inserted rows are ignored. See Handling Data Conflicts When Uploading Data from a Workbook.</p>
DownloadForInsert	<p>This action is obsolete. See What You May Need to Know About DownloadForInsert Action.</p>
FlagAllRows	<p>Sets the flag for all rows.</p> <p>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:</p> <p>Character Code 25CF, Unicode(hex)</p> <p>It appears as a solid circle.</p> <p>For information about the <code>_ADF_FlagColumn</code> column, see Special Columns in the ADF Table Component.</p>
Initialize	<p>This action performs the following actions:</p> <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 <p>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.</p>
MarkAllRowsChanged	<p>Specify this component action to mark all rows in the table as changed in <code>_ADF_ChangeColumn</code>.</p>
MarkAllRowsUnchanged	<p>Specify this component action to clear all flags from the <code>_ADF_ChangedColumn</code> column.</p>

Table A-12 (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. See Configuring an ADF Table Component to Resize Columns Based on Data at Runtime .
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 information about column properties, see 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). See Using Row-Level Action Sets in a Table Column .
RowUpSyncNoFail	It is a variant of <code>RowUpSync</code> that tolerates failures. Like <code>RowUpSync</code> , <code>RowUpSyncNoFail</code> is only intended for use in a row-level action set. See How to Synchronize Changes from ADF Table Component Using RowUpSyncNoFail .
UnflagAllRows	Removes flags from cells in the <code>_ADF_FlagColumn</code> column. For information about the <code>_ADF_FlagColumn</code> , see Special Columns in the ADF Table Component .
Upload	Uploads to the Fusion web application all rows marked as changed for this table. See Uploading Changes from an ADF Table Component . For information about resolving data conflict between the Excel workbook and the Fusion web application, see Handling Data Conflicts When Uploading Data from a Workbook .
UploadAllOrNothing	Uploads to the Fusion web application all rows marked as changed for this table. Commits successful rows only if none of the rows fail. For information about <code>UploadAllOrNothing</code> action, see Uploading Changes from an ADF Table Component Using an UploadAllOrNothing Action .

ADF Read-only Table Component Properties and Actions

Describes the `TreeID` and other properties that the ADF Read-only Table component supports, including the properties that a column in the `ReadOnlyColumn` array can use.

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`.

For information about the ADF Read-only Table component, see [Creating an ADF Read-Only Table Component](#).

Table A-13 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-14 .
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-10 .
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-14](#) lists alphabetically the properties that a column in the `ReadOnlyColumn` array can use.

Table A-14 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 ADF Output Text Component Properties .

Action Set Properties

Lists alphabetically the properties that you can configure for an action set.

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

Table A-15 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.
ActionOptions.NonBlocking	Boolean	N	Set to <code>True</code> so that the associated <code>ActionSet</code> does not block the user from using other Excel features while the user waits for the <code>ActionSet</code> to complete. The default value is <code>False</code> . For information, see How to Allow End Users to Continue Working in Excel While an ActionSet Executes .
ActionOptions.SuccessActionID	Action binding	N	Specify an action binding to invoke if an action set completes successfully. A value for this property is optional.

Table A-15 (Cont.) Action Set Properties

Name	Type	EL	Description
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"> • <code>ADFmAction</code> 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 Working with Page Definition Files for an Integrated Excel Workbook. • <code>ComponentAction</code> 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. The ADF Table component is the only component in ADF Desktop Integration that expose actions. For information about these actions, see ADF Table Component Properties and Actions. For information about invoking component actions, see How to Invoke Component Actions in an Action Set. • <code>WorksheetMethod</code> Invokes a worksheet action. For information about worksheet actions, see Worksheet Actions and Properties. • <code>Confirmation</code> Invokes a confirmation dialog. For information about the properties that this action uses, see Confirmation Action Properties. • <code>Dialog</code> Invokes a web page in a popup dialog or Excel's task pane. For information, see Displaying Web Pages from a Fusion Web Application.
Alert			<p>This group of properties determines if and how an alert-style dialog appears to the user to indicate that the action set 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 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 information about using EL expressions, see Using Resource Bundles in an Integrated Excel Workbook.</p>
Alert.Enabled	Boolean	N	<p>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 information, see How to Provide an Alert After the Invocation of an Action Set.</p>
Alert.FailureMessage	String	Y	<p>(Optional) Specify an EL expression that evaluates to a message to appear in the dialog if errors occur during execution of the action set.</p>

Table A-15 (Cont.) Action Set Properties

Name	Type	EL	Description
Alert.OKButtonLabel	String	Y	(Optional) Specify an EL expression that evaluates to a message to appear in the OK button of the dialog.
Alert.SuccessMessage	String	Y	(Optional) 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	(Optional) Specify an EL expression that evaluates to a message to appear in the title area of the dialog.
Annotation			For information about Annotation, see Table A-1 .
FailureMessage	String	Y	(Optional) Specify an EL expression that evaluates to a message to appear to the end user if the action set fails. A default message appears if you do not specify an EL expression. For information, see How to Configure Error Handling for an Action Set .
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 How to Display a Progress Bar while an Action Set Executes . Many properties in this group make use of EL expressions that reference string keys defined in resource bundles. For information, see Using Resource Bundles in an Integrated Excel Workbook .
Status.AllowCancel	Boolean	N	If True, a Cancel button is displayed in the status dialog box. For information, see How to Display a Progress Bar while an Action Set Executes .
Status.Enabled	Boolean	N	If True (default), a status window appears during the execution of the action set. If False, 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 Automatic, BothBarsAlways, MainBarOnly, DetailBarOnly, and MainMessageOnly.
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.

Confirmation Action Properties

[Table A-16](#) 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-15](#).

For information, see [How to Invoke a Confirmation Action in an Action Set](#).

Table A-16 Confirmation Action Properties

Name	Type	EL	Description
Annotation			(Optional) For information about Annotation, see Table A-1 .
CancelButtonLabel	String	Y	(Optional) An EL expression that is evaluated and displayed in the Cancel button at runtime. By default, no value is specified.
OKButtonLabel	String	Y	(Optional) An EL expression that is evaluated and displayed in the OK button at runtime. By default, no value is specified.
Prompt	String	Y	(Optional) An EL expression that is evaluated and displayed in the main area of the confirmation dialog at runtime. By default, no value is specified.
Title	String	Y	(Optional) An EL expression that is evaluated and displayed in the title area of the confirmation dialog at runtime. By default, no value is specified.

Dialog Action Properties

[Table A-17](#) 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-15](#).

For information about how to use the properties in [Table A-17](#) to invoke a web page from a Fusion web application, see [Displaying Web Pages from a Fusion Web Application](#).

Table A-17 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 <code>True</code> (default) to run the web page specified by the <code>Dialog.Page</code> 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 <code>Dialog.ShareFrame</code> property.
Target	List	N	Specifies how the web page the action invokes is rendered. Select: <ul style="list-style-type: none"> <code>Popup</code> to render the web page in a modal dialog within an embedded web browser. <code>TaskPane</code> to render the web page in runtime task pane.
Title	String	Y	Write an EL expression that resolves to the title of the <code>Dialog</code> 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 <code>Height</code> is 625 and 600 for <code>Width</code> .

Workbook Actions and Properties

Describes the actions that a workbook can invoke.

For information about configuring ribbon commands to invoke these actions, see [How to Define a Workbook Ribbon Command for the Runtime Ribbon Tab](#).

Table A-18 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> <p>See About Security In Your Integrated Excel Workbook.</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 component, 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.</p>
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>

Table A-18 (Cont.) Workbook Actions

Action	Description
ViewAboutDialog	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. The dialog also allows end users to generate a diagnostic report and, if logged in, check for an upgrade of the ADF Desktop Integration add-in. See Generating an ADF Desktop Integration Diagnostic Report and How to Upgrade ADF Desktop Integration On a Local System .
ToggleStatusViewer	When invoked, this action shows or hides a Status Viewer in Excel's task pane to display status information to end users. The ribbon command that end users click to invoke this action shows and hides the Status Viewer. See Using the Status Viewer to Report Error Messages to End Users .

[Table A-19](#) lists alphabetically the ADF Desktop Integration properties that an Excel workbook can use.

Table A-19 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 (<code>.jws</code>) where you developed the desktop integration project. The path must be less than the Windows maximum path length of 260 characters.</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 may prompt you to specify a value for this property the first time that you open an integrated Excel workbook.</p> <p>See How to Configure a New Integrated Excel Workbook.</p>
AutoDisplayStatusViewerEnabled	Boolean	N	<p>Set to <code>True</code> to display the Status Viewer automatically if an error occurs. Set to <code>False</code> to require end users to click the Status Viewer ribbon command in the Excel ribbon to display the Status Viewer. The default value is <code>True</code>.</p> <p>See Using the Status Viewer to Report Error Messages to End Users.</p>
BrandingItems	Array	N	<p>An array of name-value pairs of literal string or EL resources expressions (for example, <code>#{res['myAppName']}</code>).</p> <p>For information about branding your integrated Excel workbook, see Branding Your Integrated Excel Workbook.</p>
Compatibility	Array	N	<p>Ensures that workbooks created with a different release of ADF Desktop Integration version that did not include a particular feature do not change their behavior in another release.</p> <p>For information about compatibility properties, see ADF Desktop Integration Compatibility Properties.</p>

Table A-19 (Cont.) Workbook Properties

Name	Type	EL	Description
CustomizationEnabled	Boolean	N	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. For information about enabling customization, see Customizing Workbook Integration Metadata at Runtime .
Login.WindowSize	Integer	N	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.
Parameters	Array	N	An array of workbook 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 parameters in the Fusion web application's page. Each workbook parameter (parameter that matches 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. Each element in the array supports the following properties: <ul style="list-style-type: none"> Annotation For information about this property, see Table A-1. Parameter You specify the name of the workbook parameter you defined in the page of the Fusion web application from which the end user downloads the integrated Excel workbook. For information about using this property, see Passing Parameter Values from a Fusion Web Application Page to a Workbook.
Project	String	N	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. 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. ADF Desktop Integration loads the names of the available projects from the <code>application_name.jws</code> file specified by <code>ApplicationHomeFolder</code> .
RemoteServletPath	String	N	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: <code>/adfdiRemoteServlet</code>

Table A-19 (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 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>See 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 information about the ribbon tab and its commands, see Configuring the Runtime Ribbon Tab.</p>
Runtime Ribbon Tab.Annotation	String	N	<p>For information about this property, see 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 K for the <code>Runtime Ribbon Tab.Title</code>. Excel does not allow the letter K to be used here when the workbook is running in the <code>ar_SA</code> culture.</p>

Table A-19 (Cont.) Workbook Properties

Name	Type	EL	Description
Runtime Ribbon Tab.Commands	Workbook 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 information about this property, see Table A-1. Label For 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 command invokes. For information about workbook actions, see Table A-18.
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> and must be less than the Windows maximum path length of 260 characters.
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 Checking the Integrity of an Integrated Excel Workbook's Metadata.</p>

Worksheet Actions and Properties

Describes the worksheet-level actions that action sets can invoke and the ADF Desktop Integration properties that an Excel worksheet can use.

Action sets, as described in [Using Action Sets](#), can invoke the following worksheet-level 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 data values 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 list of error messages from the most recent action set invocation.



Note:

This action is deprecated. It is no longer necessary. All relevant error messages are available via the Status Viewer, as described in [Using the Status Viewer to Report Error Messages to End Users](#).

When you configure a ribbon command to invoke an action binding or method action binding, the action set to invoke when a user clicks the ribbon command at runtime is populated as follows by default:

1. UpSync
2. Action or method action binding that you specify for the ribbon command
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 ribbon command's action set or one of the worksheet events (Startup or Activate) described in [Table A-20](#) to invoke the DownSync worksheet action first.

For information about configuring ribbon commands, see [Configuring the Runtime Ribbon Tab](#).

[Table A-20](#) describes the ADF Desktop Integration properties that an Excel worksheet can use.

Table A-20 Worksheet Properties

Name	Type	EL	Description
Annotation	String	N	For information about this property, see Table A-1 .
CustomAttributePropertiesEnabled	Boolean	N	Specifies whether custom attribute properties defined in a view object on the server are supported in EL expressions. The default value of this property is False. See Using ADF Desktop Integration EL-based Properties with Custom Attribute Properties .
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 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 <code>Events.n</code> describe the properties that an element in this array supports where <i>n</i> refers to a specific element in the array.
Events.n.ActionSet	ActionSet	N	For information about the properties of action sets, see Action Set Properties .

Table A-20 (Cont.) Worksheet Properties

Name	Type	EL	Description
Events.n.InvokeOnceOnly	Boolean	N	<p>The default value of this property is <code>False</code>.</p> <p>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.</p>
Events.n.Annotation	String	N	<p>For information about the annotation property, see Table A-1.</p>
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"> • <code>Startup</code> Excel workbook opens and the worksheet is activated for the first time. • <code>Shutdown</code> Excel workbook closes or Excel application exits. • <code>Activate</code> User navigates to the current worksheet. • <code>Deactivate</code> User navigates away from the current worksheet or <code>Shutdown</code> event triggered. <p>Note that the worksheet events complete execution even if the action sets that they invoke fail.</p> <p>For information about worksheet events and action sets, see 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"> • <code>Off</code> Worksheet protection is not used at runtime. • <code>Automatic</code> Worksheet protection is enabled automatically at runtime. <p>The default value for this property is <code>Off</code>.</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>

Table A-20 (Cont.) Worksheet Properties

Name	Type	EL	Description
Ribbon Commands	Array	N	<p>Specify one or more actions that appear as worksheet ribbon 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 Image <ul style="list-style-type: none"> Specifies the image to display as the worksheet-level ribbon command at runtime. ADF Desktop Integration provides the images that you can use. Label SelectActionSet <p>For information about the <code>Annotation</code> and <code>Label</code> properties, see Table A-1. For information about the <code>SelectActionSet</code> property, see 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> <p>See How to Configure a Worksheet Ribbon Command for the Runtime Ribbon Tab.</p>
Page Definition	String	N	<p>Specify the page definition file to associate with the worksheet. The fully qualified path to the page definition file must be less than the Windows maximum path length of 260 characters. For information about page definition files, see Working with Page Definition Files for an Integrated Excel Workbook.</p>
Parameters	Array	N	<p>An array of parameters defined in this worksheet's page definition file and bound here to workbook parameters. Each element in the array supports the following properties:</p> <ul style="list-style-type: none"> Annotation <ul style="list-style-type: none"> For information about this property, see Table A-1. Parameter <ul style="list-style-type: none"> Specify the ID of a <code>parameter</code> element that you added to the page definition file associated with the worksheet. Value <ul style="list-style-type: none"> Write an EL expression that references the value of the <code>Parameter</code> property you specified for the workbook parameter (workbook <code>Parameters.Parameter</code> property). <p>For information about using this property, see Passing Parameter Values from a Fusion Web Application Page to a Workbook.</p>

Table A-20 (Cont.) Worksheet Properties

Name	Type	EL	Description
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> <p>See Using an Integrated Excel Workbook Across Multiple Web Sessions.</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"> <li data-bbox="786 789 1377 968"> <p>• <code>AttributeID</code></p> <p>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.</p> <li data-bbox="786 978 1377 1041"> <p>• <code>Annotation</code></p> <p>For information about this property, see Table A-1.</p>
RowData.ChangeIndicatorAttribute Binding ID		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>See Handling Data Conflicts When Uploading Data from a Workbook.</p>
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>See Restore Server Data Context Between Sessions.</p>

Table A-20 (Cont.) Worksheet Properties

Name	Type	EL	Description
<code>ServerContext.CacheDataContexts</code>	Array	N	<p>Add elements to the <code>CacheDataContexts</code> array for cases where there is more than one iterator defined in the binding container whose server-side context must be reestablished. The <code>CacheDataContexts</code> array supports the following properties to store the worksheet's cached data context:</p> <ul style="list-style-type: none"> <code>RestoreDataContextActionID</code> References an action binding to invoke. <code>CachedServerContexts</code> An array that identifies the attribute binding values to cache and set before the action binding specified by <code>RestoreDataContextActionID</code> is invoked. Each element in the <code>CachedServerContext</code> array supports the <code>CachedAttributeID</code> and <code>RestoredAttributeID</code> properties. <code>CachedAttributeID</code> identifies the attribute binding value to cache in the worksheet. <code>RestoredAttributeID</code> 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 <code>RestoredAttributeID</code>, the value of <code>CachedAttributeID</code> is used as the destination attribute binding value and its value is set before invoking the action set. <code>Annotation</code> For information about this property, see Frequently Used Properties in the ADF Desktop Integration.
<code>ServerContext.IDAttributeID</code>	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>
<code>ServerContext.SendParameters</code>	Boolean	N	<p>The default value of this property is <code>False</code>. When set to <code>True</code>, the workbook sends initialization parameters for this worksheet when reestablishing context across multiple sessions.</p>
<code>SetupActionID</code>	Binding	N	<p>Specify the <code>ADFmAction</code> binding to be invoked before the binding container metadata is retrieved. A value for this property is optional. 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. See Using Explicit Worksheet Setup Action.</p>

Table A-20 (Cont.) Worksheet Properties

Name	Type	EL	Description
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 Compatibility Properties

Various ADF Desktop Integration features have been added in different releases of the product. The compatibility properties ensure that workbooks created with ADF Desktop Integration versions that did not include a given feature do not change their behavior.

[Table A-21](#) lists the ADF Desktop Integration compatibility properties. Integrated Excel workbook developers may want to review these properties and the associated feature to determine whether to enable them. To access these properties in your integrated Excel workbook, click **Workbook Properties** in the Workbook group of the Oracle ADF tab to display the Edit Workbook Properties and expand the **Behavior** and **Compatibility** properties.

Table A-21 ADF Desktop Integration Compatibility Properties

Name	Type	EL	Description
<code>DataEntryValidationEnabled</code>	Boolean	N	<p>Specifies whether ADF Desktop Integration performs data entry validation.</p> <p>See Providing Data Entry Validation for an Integrated Excel Workbook.</p>
<code>RejectExcelErrorsEnabled</code>	Boolean	N	<p>Specifies whether ADF Desktop Integration rejects Excel cell error values that integrated Excel workbooks send to the Fusion web application. The default value is <code>True</code>.</p> <p>See Providing Data Entry Validation for an Integrated Excel Workbook.</p>
<code>TableComponents.ModelDrivenColumns.DatePickerEnabled</code>	Boolean	N	<p>Specifies whether the date picker can be used in model-driven columns.</p> <p>See Inserting an ADF Input Date Component and Adding a ModelDrivenColumnComponent Subcomponent to Your ADF Table Component.</p>

Table A-21 (Cont.) ADF Desktop Integration Compatibility Properties

Name	Type	EL	Description
TableComponents.ModelDrivenColumns.InputListOfValuesPickerEnabled	Boolean	N	Specifies whether model-driven columns can leverage existing Model layer metadata and provide a Search and Select user interface in a picker dialog. See Adding a Model-Driven List Picker to an ADF Table Component .
TableComponents.RowActionSetModelMgmtEnabled	Boolean	N	Specifies whether row-level action sets can manage the server-side model state. See Using Row-Level Action Sets in a Table Column .
TableComponents.SmartRowFailureReportingEnabled	Boolean	N	The default value (True) enables enhanced error reporting for the ADF Table component Upload and DeleteFlaggedRows actions.

B

ADF Desktop Integration EL Expressions

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:

- [Guidelines for Creating EL Expressions](#)
- [EL Syntax for ADF Desktop Integration Components](#)
- [Attribute Control Hints in ADF Desktop Integration](#)

Guidelines for Creating EL Expressions

EL expressions that you write in your integrated Excel workbook can include literal values, a valid Excel formula, binding, component, and resource bundle 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`
- Binding expressions to evaluate control binding values or hints. For example, `#{row.bindings.ProductId.inputValue}`.
- Component expressions to evaluate component properties. For example, `#{components.TAB416222534.rowCount}`.
- Resource bundle expressions to evaluate locale specific resources defined on the server. For example: `#{res['excel.saveButton.label']}`

For information about the supported binding, component, and resource bundle expression syntax, see [EL Syntax for ADF Desktop Integration Components](#).

- 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 better strategy 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.

- 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.

EL Syntax for ADF Desktop Integration Components

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"
readOnly	Table.Column	Boolean	True False	False

Write EL expressions with the following syntax to retrieve:

- Workbook parameters

```
#{workbook.params.parameterName}
```

where *parameterName* is the name of the workbook parameter. For information about using these parameters, see [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 information about resource bundles, see [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 [Using the Expression Builder](#) to generate some of the EL expressions described 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	inputValue	""
<p>Write an EL expression with the following syntax when adding a dynamic column to an ADF Table component as described in 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>			

Attribute Control Hints in ADF Desktop Integration

ADF Desktop Integration can read the values of attribute control hints. 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 *Defining Attribute Control Hints for Entity Objects of [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 *Defining UI Hints for View Objects of 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 <code>true</code> if the associated attribute binding is updatable.
readOnly	Boolean	<p>This attribute control hint is unique to ADF Desktop Integration. Returns <code>true</code> 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><code>#{bindings.attributeID.hints.readOnly}</code></pre> <p>rather than:</p> <pre><code>=NOT(#{bindings.attributeID.hints.updateable})</code></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 Frequently Used Properties in the ADF Desktop Integration.</p>
mandatory	Boolean	Returns <code>true</code> 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"> • <code>string</code> • <code>number</code> • <code>date</code> • <code>boolean</code> • <code>other</code>
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> <p>For information about display width, see <i>How to Set User Interface Hints on View Criteria to Support Search Forms in Developing Fusion Web Applications with Oracle Application Development Framework</i>.</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.

ADF Desktop Integration also supports custom attribute properties in table EL-based properties. For information, see [Using ADF Desktop Integration EL-based Properties with Custom Attribute Properties](#).

C

Troubleshooting an Integrated Excel Workbook

Describes how to troubleshoot an integrated Excel workbook and generate log files when you encounter problems during development. This appendix includes the following sections:

- [Verifying That Your Fusion Web Application Supports ADF Desktop Integration](#)
- [Generating an ADF Desktop Integration Diagnostic Report](#)
- [Troubleshooting Connection Problems to Fusion Web Applications](#)
- [Verifying End-User Authentication for Integrated Excel Workbooks](#)
- [Generating Log Files for an Integrated Excel Workbook](#)

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 information about the property inspector, see [Using the Property Inspector](#).

Verifying That Your Fusion Web Application Supports ADF Desktop Integration

Using a server ping test, 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.

For Fusion web applications that enforce authentication, you can use the server ping test to confirm that the proper authentication configuration is in place for the ADF Desktop Integration servlet URL.

ADF Desktop Integration relies on various Internet Explorer specific settings. For this reason, please perform the verification test using Internet Explorer.

To verify that the ADF Desktop Integration remote servlet is running:

- 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/summit/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	12.2.1.1.0 (4.4.0.14749)
Build	ADFDI_4.4.0_160325.1305

[About ADF Desktop Integration](#) - [Install ADF Desktop Integration Add-in for Excel](#) - [Run client health check tool](#)

Copyright © 1997, 2016, Oracle and/or its affiliates. All rights reserved.

Generating ADF Desktop Integration Diagnostic Reports

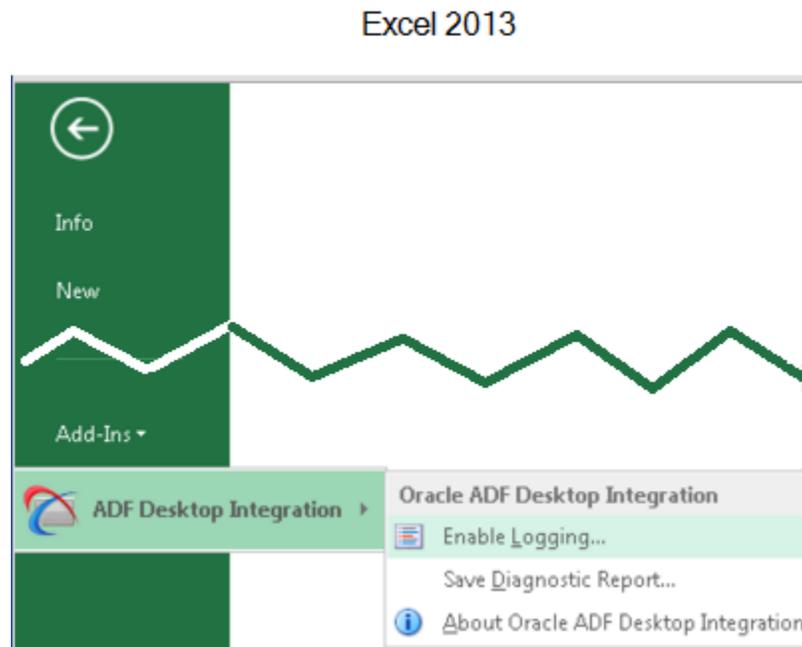
ADF Desktop Integration provides a number of tools so that you and end users can diagnose and resolve issues that may occur in the client-side environment.

These include the Client Health Check Tool, described in [Running the Client Health Check Tool](#). This determines if the client environment is configured correctly and provides options to resolve some commonly-encountered issues. You or end users can also generate a diagnostic report (adfdi-diagnostic-report.txt) from an integrated Excel workbook, as described in the following section.

How to Generate the ADF Desktop Integration Diagnostic Report

You or your end users can generate the diagnostic report by clicking the **Save Diagnostic Report** menu entry that ADF Desktop Integration adds to Microsoft Excel's **File > Add-Ins** menu, as shown in [Figure C-2](#).

Figure C-2 File Menu to Save Diagnostic Report



The location of this menu entry depends on the version of Microsoft Excel that you use. [Figure C-2](#) shows the location of this menu entry in Microsoft Excel 2013.

Alternatively, you or end users can generate the diagnostic report from the About dialog.

To generate the ADF Desktop Integration diagnostic report from the About dialog:

1. Open the integrated Excel workbook.
2. If you have opened the integrated Excel workbook in the design mode, click the **About** button in the Workbook group of the Oracle ADF tab.
 If you have opened the integrated Excel workbook in runtime mode, click the **About** button of the runtime ribbon tab.
3. Click the **Diagnostic Report** button of the About dialog.
4. Save the diagnostic report text file. By default, the file is saved as `adfdi-diagnostic-report.txt` in the Desktop directory (for example, `C:\Users\<USER_NAME>\Desktop`).
5. The Diagnostic Report dialog opens describing the location of the saved file. Click **OK** to open the file in the default text editor.

What You May Need to Know About the ADF Desktop Integration Diagnostic Report

The diagnostic report is a text file and includes a variety of information such as:

- ADF Desktop Integration add-in version

- Microsoft Windows version
- Microsoft Excel version
- Values of all properties from the Version tab of the About dialog
- Values of all properties listed in the Properties tab of the About dialog
- List of Excel COM add-ins
- Branding items from the About tab, if the report is generated at runtime
- ADF Desktop Integration servlet version, if the report is generated after a valid login

You can open and edit the text file in any text editor, or Excel. Each row in the file consists of a key-value pair separated by tabs.

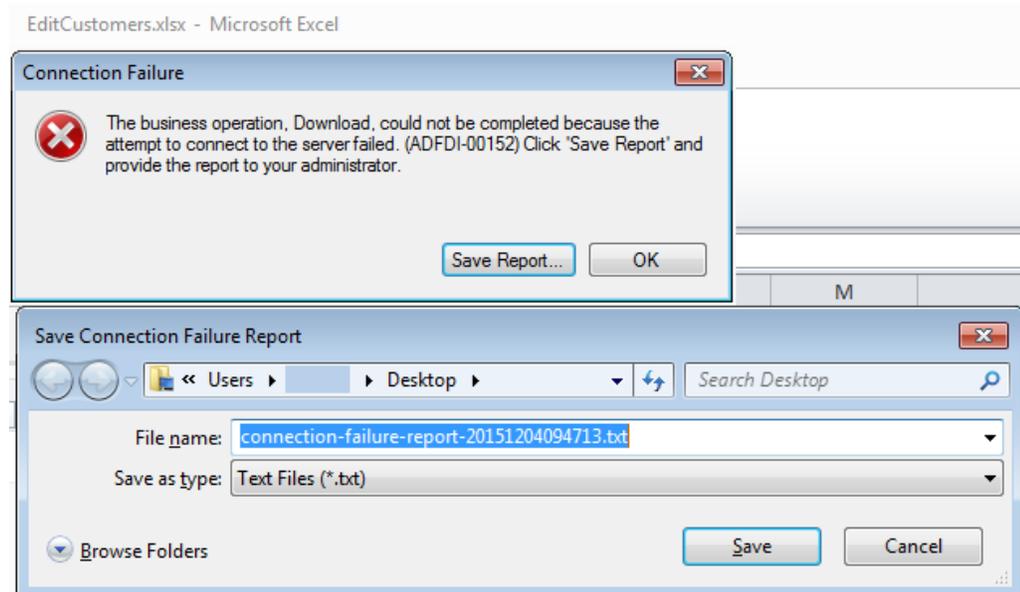
Before end users send the diagnostic file to you, ask them to review the report and remove any sensitive information that they do not want to share.

Troubleshooting Connection Problems to Fusion Web Applications

ADF Desktop Integration provides end users with connection failure reports to help diagnose the cause of connection failures from integrated Excel workbooks to Fusion web applications.

A connection failure report contains information that ADF Desktop Integration generates when an attempt to connect to a Fusion web application fails or the end user cancels the connection attempt. [Figure C-3](#) shows the dialog that appeared when the `EditCustomers.xlsx` workbook failed to connect to the Summit sample application because the latter application was offline. End users click **Save Report** to save the report to a directory on their machine.

Figure C-3 End User’s Dialog to Save a Connection Report



Before end users send the connection failure report file to you, ask them to review the report and remove any sensitive information that they do not want to share. The following example shows an extract of the report generated in [Figure C-3](#).

```
Report: Oracle ADF Desktop Integration (ADFdi) Connection Failure Report
Generated: (UTC) 04/12/2015 09:47:13
Language: en-US
```

```
*** NOTE: this file contains detailed diagnostic information. Review the
contents and edit out any information you do not wish to share with third
parties. ***
```

```
=== Summary ===
```

```
Failure Phase
  AuthenticationTest
```

```
Failure Reason
  UnexpectedHttpStatusException - ADFDI-00501: An unexpected status: 404
  (NotFound) was returned from the server while requesting the URL: http://
  127.0.0.1:7101/submit/adfdiRemoteServlet
```

```
ADFdi servlet URL
  http:// 127.0.0.1:7101/submit/adfdiRemoteServlet
  ...
```

For more information about troubleshooting connection problems, see the documents that you can retrieve from My Oracle Support (<https://support.oracle.com>) if you search for Doc IDs 2014348.1 and 2094772.1.

Verifying End-User Authentication for Integrated Excel Workbooks

If 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. See [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 [Securing Your Integrated Excel Workbook](#).

Generating Log Files for an Integrated Excel Workbook

ADF Desktop Integration can generate log files that capture information based on events triggered by the server-side and client-side software.

The following pieces of software within ADF Desktop Integration trigger events:

- HTTP filter and the ADF Desktop Integration remote servlet on the web server (server-side logging)
For information about server-side logging, see [About Server-Side Logging](#).
- Excel workbook which you integrate with your Fusion web application (client-side logging)
For information about client-side logging, see [About Client-Side Logging](#).

About Server-Side Logging

You configure the generation of server-side log files for ADF Desktop Integration the same way as for other Oracle ADF modules. This involves setting values that specify the verbosity level and output location in a configuration file named `logging.xml`. You can also use Oracle Diagnostic Logging Configuration of JDeveloper to configure the logging levels specified in the `logging.xml` file. For information about using the JDeveloper debugging tools and ADF Logger, see *Using the ADF Logger in Developing Fusion Web Applications with Oracle Application Development Framework*.

[Table C-1](#) describes the package names that you supply as attribute parameters to the `<logger>` elements in the `logging.xml` file to configure log file generation in ADF Desktop Integration.

Table C-1 Package Names for Log File Configuration

To generate log file entries for this component...	Enter this package name...
All ADF Desktop Integration server logic	<code>oracle.adf.desktopintegration</code>
ADF Desktop Integration HTTP filter	<code>oracle.adf.desktopintegration.filter</code>

Using the Oracle Diagnostics Log Analyzer to Analyze ADF Desktop Integration Servlet Requests

Using the Oracle Diagnostics Log Analyzer, you can view a hierarchical breakdown of elapsed time spent performing each ADF Desktop Integration servlet request. The hierarchical breakdown also includes the time spent in other ADF components, such as the ADF Model layer. For information about using the log analyzer for viewing web requests, see *How to Use the Log Analyzer to View Log Messages in Developing Fusion Web Applications with Oracle Application Development Framework*.

 **Tip:**

The hierarchical breakdown can be helpful in identifying performance bottlenecks due to unusually long execution times.

In order to log a complete hierarchy tree of ADF event messages, including ADF Desktop Integration events, specify log level `CONFIG` for the `oracle.adfdiagnostics` package. For more information about the `oracle.adfdiagnostics` logger, see [How to Create an Oracle ADF Debugging Configuration in *Developing Fusion Web Applications with Oracle Application Development Framework*](#).

About Client-Side Logging

ADF Desktop Integration provides a number of methods to create log files of activity that occur within the ADF Desktop Integration add-in. Logging is always enabled at a high level, the `Information` level. These logs are in plain text format and include the basic user steps as well as any errors and warning. For information, see [What You May Need To Know About Always-On Logging](#).

In some cases, you may need more detailed logs to troubleshoot a particular problem. There are several methods for producing detailed logs at the `Verbose` level. The simplest method is to enable transient verbose logging by selecting **Enable Logging...** from the Excel add-in menu that ADF Desktop Integration adds to Microsoft Excel. These logs are in XML format and include a very detailed account of client-side activity. For information, see [Enabling Transient Verbose Logging](#).

Additionally, there are two other methods for controlling client logs:

- Using the Logging options that the Oracle ADF tab exposes, as described in [How to Configure Logging in the Oracle ADF Tab](#).
- Using the ADF Desktop Integration Configuration file, as described in [About the ADF Desktop Integration Configuration File](#).
- Configuring environment variables, as described in [How to Configure Logging Using User Environment Variables](#).

Consider using one of the latter two options if you want to capture log information that spans one or more integrated Excel workbook restarts and/or you want to configure a lower logging level than `verbose`.

[Table C-2](#) describes the different log levels that you can enable for client-side logging.

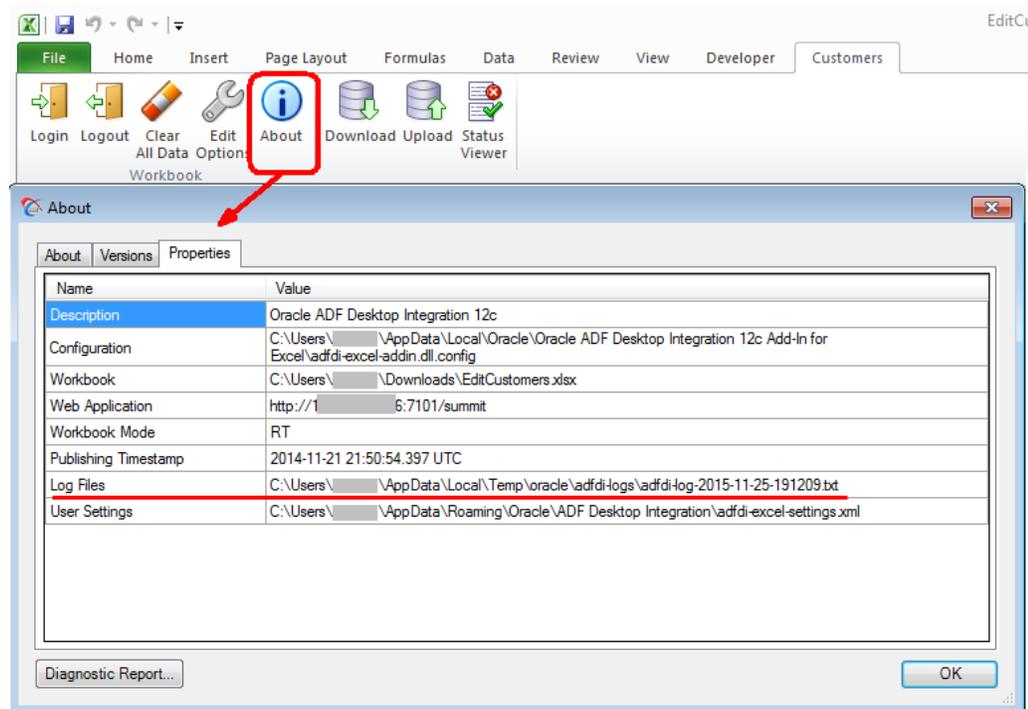
What You May Need To Know About Always-On Logging

By default, ADF Desktop Integration enables logging at the `Information` level to a text file in the `adfdi-logs` sub-directory of one of the following directories, listed in order of preference:

- `%TEMP%\oracle\adfdi-logs`
- `%TMP%\oracle\adfdi-logs`
- `%LocalAppData%\temp\oracle\adfdi-logs`
- `%SystemDrive%\oracle\adfdi-logs`

To determine which of the above directories ADF Desktop Integration uses to store log file information, see the Log Files property that you can view from the About dialog of your integrated Excel workbook, as shown in Figure C-4.

Figure C-4 Directory Location of Always-on Log Files



ADF Desktop Integration creates a new log file each time you start a new Excel session with an integrated Excel workbook. It uses the convention `adfdi-log-timestamp.txt` when naming log files where `timestamp` is the time at which you start the integrated Excel workbook (for example, `adfdi-log-2015-11-25-191209.txt`).



Note:

ADF Desktop Integration purges the oldest always-on log file(s) in the folder when the folder reaches a certain size.

For information, see the document that you can retrieve from My Oracle Support (<https://support.oracle.com>) if you search for Doc ID 2094378.1.

Enabling Transient Verbose Logging

You or your end users can enable verbose logging from a menu in the active integrated Excel workbook. ADF Desktop Integration writes verbose log information to a file in a directory you specify when you enable the logger from the menu.

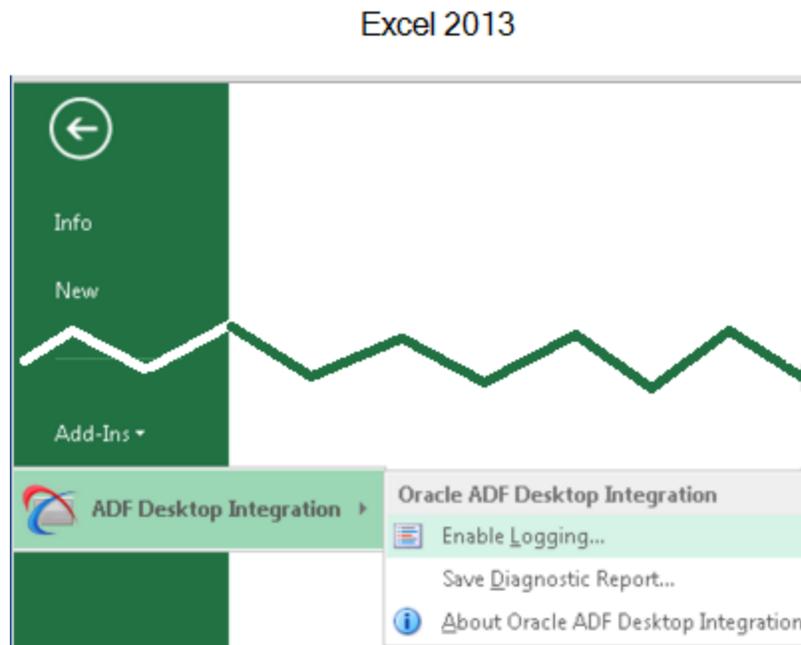
The filename that ADF Desktop Integration creates when you enable this logging has the following format:

adfdi-log-timestamp.xml

Where *timestamp* is the time at which you enable logging. For example, adfdi-log-2015-11-24-145055.xml.

The menu options you choose to enable this logging depend on the version of Microsoft Excel that you use. Figure C-5 shows the location of this menu entry in Microsoft Excel 2013.

Figure C-5 Excel File Menu to Enable Verbose Logging



Once you quit Microsoft Excel, ADF Desktop Integration stops logging information to the log file.

To enable transient verbose logging:

1. Launch Microsoft Excel from the Windows menu but do not open the integrated Excel workbook with the issue for which you want to capture verbose log data.
2. From the Office or File menu of Excel, select **Add-Ins > ADF Desktop Integration > Enable Logging**, as shown in Figure C-5.
3. Save the log file to a location of your choosing or use the default Desktop location that the Save Logging Data As dialog proposes. Note the log file name and location for later reference.
4. Open the integrated Excel workbook and repeat the steps to reproduce the issue for which you want to capture verbose log data.
5. Once you have completed the steps, you can quit Microsoft Excel. Once you quit Microsoft Excel, ADF Desktop Integration stops logging verbose data to the log file you created in Step 3.

 **Tip:**

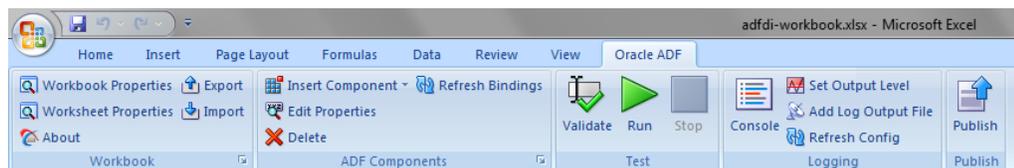
Save a diagnostic report with information about the environment where ADF Desktop Integration runs that you can submit to Technical Support with the verbose log data that you have captured for your issue. From the Office or File menu of Excel, select **Add-Ins > ADF Desktop Integration > Save Diagnostic Report**, as shown in [Figure C-5](#).

For information, see the document that you can retrieve from My Oracle Support (<https://support.oracle.com>) if you search for Doc ID 2094434.1.

How to Configure Logging in the Oracle ADF Tab

The Oracle ADF Tab, shown in [Figure C-6](#), provides a Logging group of menu options that are available in both design mode and test mode.

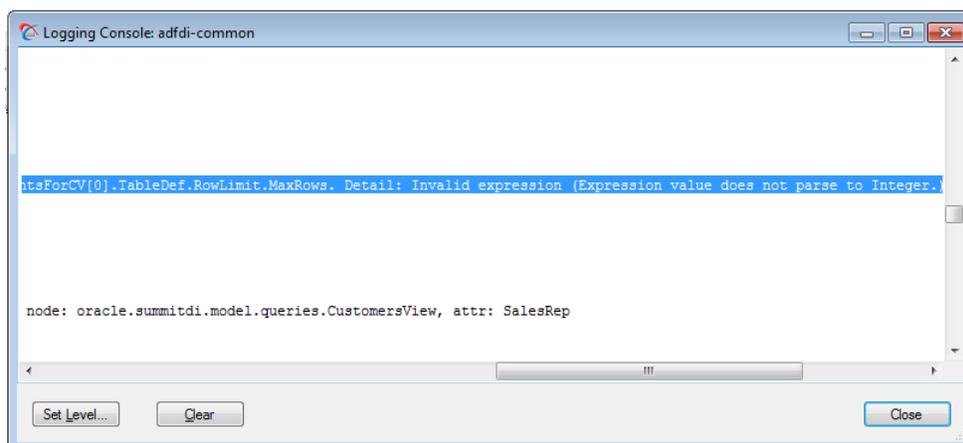
Figure C-6 Logging Tools in Oracle ADF Tab



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-7](#) 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.

Figure C-7 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-2](#) describes the log levels that client-side logging supports. The log levels are cumulative as you read down the list in [Table C-2](#). That is, the `Information` level includes the data logged in the `Critical`, `Error`, and `Warning` levels, but not the `Verbose` level.

Figure C-8 Logging Output Level Dialog

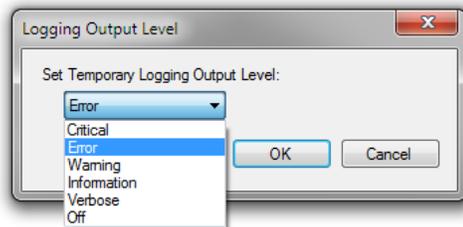


Table C-2 Client-Side Logging Levels

Level	Description
Critical	Captures critical information.
Error	Captures information about severe errors and exceptions.
Warning	Captures information about warnings.
Information	Captures lifecycle and control flow events. This is the default value.
Verbose	Captures detailed information about the execution flow of the application.

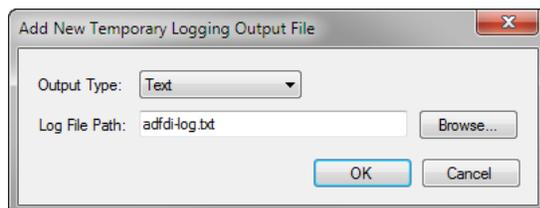
 **Note:**

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-9 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 exit Excel, 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 can determine the level of information logged by the ADF Desktop Integration add-in.

For information about the creation and configuration of the ADF Desktop Integration configuration file, see [About the ADF Desktop Integration Configuration File](#).

About the ADF Desktop Integration Configuration File

The ADF Desktop Integration configuration file is saved as `adfdi-excel-addin.dll.config`. 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 information about elements of the configuration file, see the Configuration File Schema for the .NET Framework section in [Microsoft Developer Network](#) documentation. For 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 an `.xml` log file. 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.XmlWriterTraceListener"
            name="adfdi-common-excel.xml"
            initializeData="c:\logs\adfdi-common-excel.xml"
            traceOutputOptions="ThreadId, ProcessId, DateTime"/>
        </listeners>
      </source>
    </sources>
  </system.diagnostics>
</configuration>
```

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.

For information, see the How To Obtain Log Files For ADF Desktop Integration document that you can retrieve from My Oracle Support (<https://support.oracle.com>) if you search for Doc ID 2012985.1.

 **Tip:**

Be sure to exit Excel completely prior to configuring the environment variables.

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-3](#).

Table C-3 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 adfdi-common-file variable. The generated log file will be in XML format.
adfdi-common-level	That specifies the level of logging. Table C-2 lists valid values.

5. Click **OK**.

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 information about the `TraceSource` class, see [Microsoft Developer Network](#) documentation.

D

ADF Desktop Integration Settings in the Web Application Deployment Descriptor

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:

- [Configuring the ADF Desktop Integration Servlet](#)
- [Configuring the ADF Desktop Integration Excel Download Filter](#)
- [Configuring the ADF Library Filter for ADF Desktop Integration](#)
- [Examples in a Deployment Descriptor File](#)

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 that contains the servlet is in the following directory:

```
MW_HOME\oracle_common\modules\oracle.adf.desktopintegration
```

where *MW_HOME* is the Middleware Home directory.

When you add ADF Desktop Integration to your project as described in [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

3. In Servlets page, click the **Servlet Mappings** tab, and then click the **Add** icon to create a row in the Servlet Mapping table.

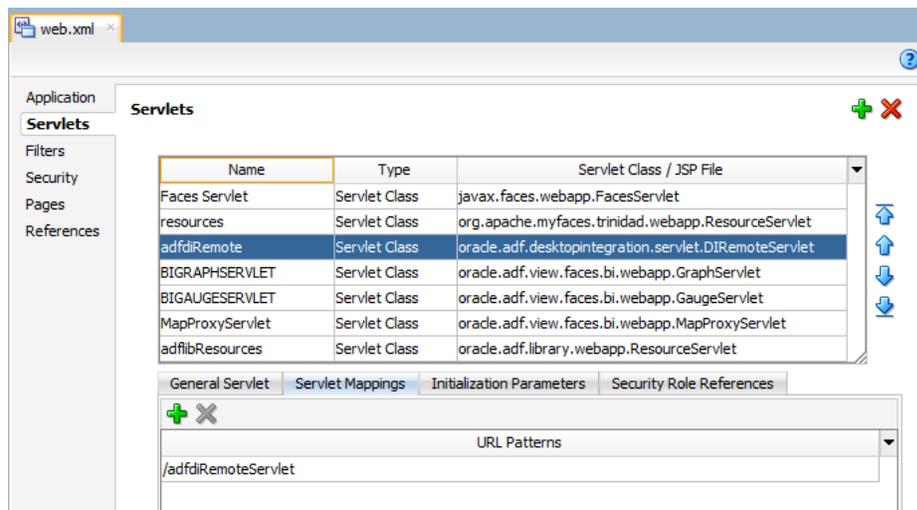
Enter the value as described in [Table D-2](#) to add a URL pattern for the `adfdiRemote` servlet in the Fusion web application. The value that you enter must match the value that you specify in the integrated Excel workbook for the `RemoteServletPath` workbook property. Note that values are case sensitive.

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



4. 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
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.

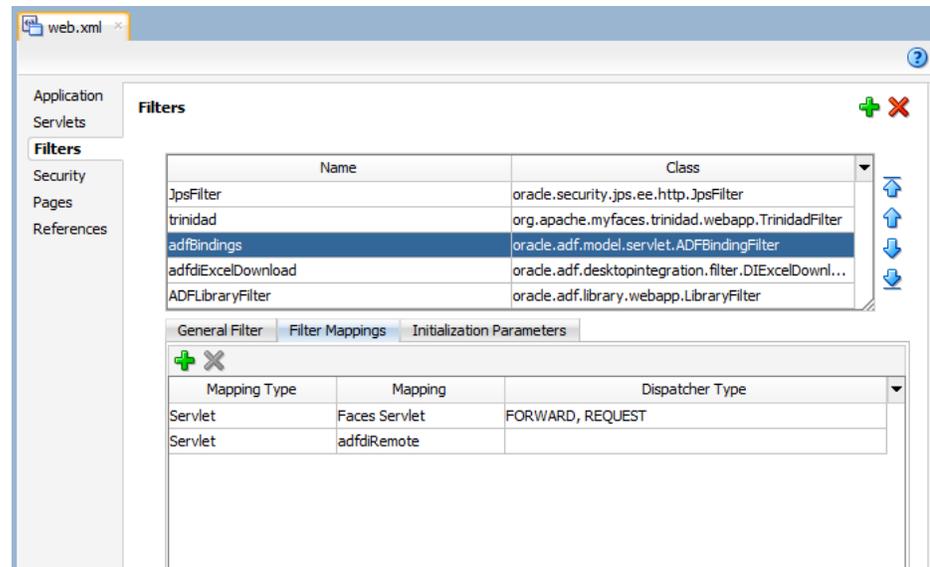
Enter the values as described in [Table D-4](#) to add the mapping filter to the adfdiRemote servlet. The filter mapping must match with the Servlet name in [Step 2](#).

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.

Configuring the ADF Desktop Integration Excel Download Filter

Use 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.

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
```

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 before the integrated Excel workbook is downloaded. 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.

Table D-5 (Cont.) Properties to Configure HTTP Filter

For this property...	Enter this value...
Description	(Optional) In General Filter tab, enter a description of the filter.

- 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	*.xlsx
Dispatcher Type	No value is required for this property.

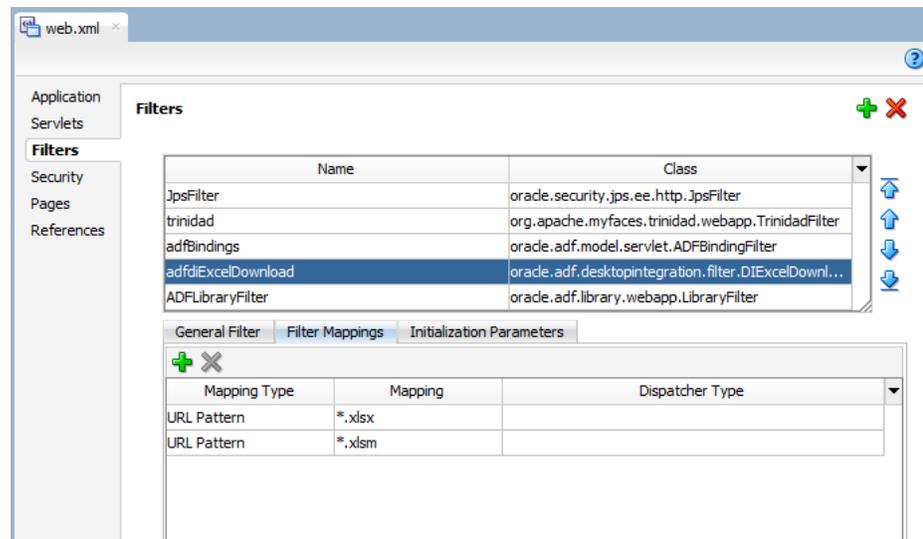
- Add another filter mapping for integrated Excel workbooks that use the macro-enabled workbook format (.xlsm) by entering values as described in [Table D-7](#).

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

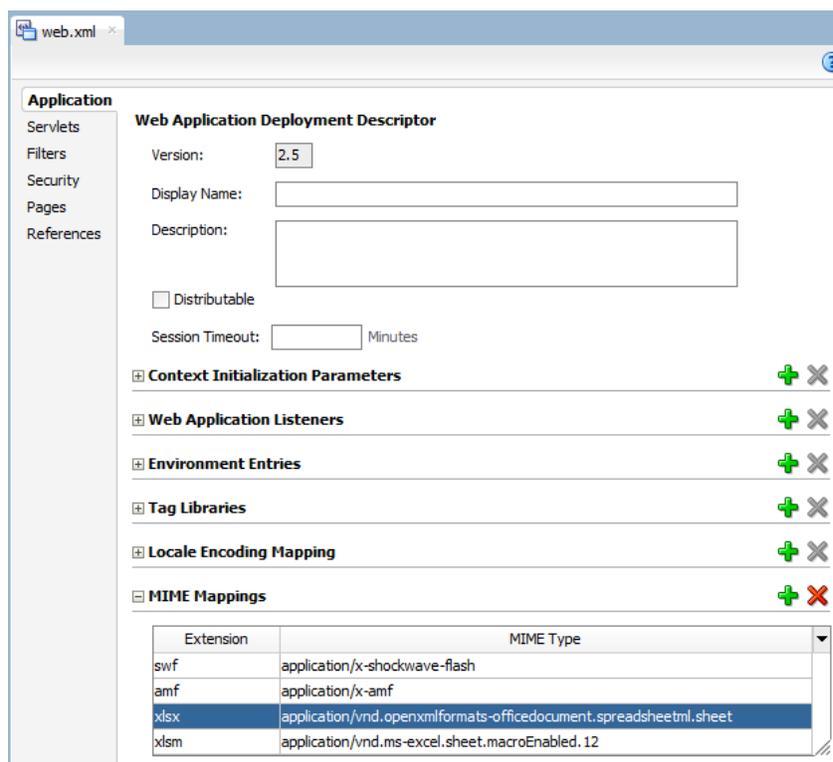
- Add another MIME type for integrated Excel workbooks that use the macro-enabled workbook format (.xlsm) by entering values as described in [Table D-9](#).

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.

ADF Desktop Integration-enabled web applications can also display a system check that verifies the end user's environment when the end user attempts to download an integrated Excel workbook. You can manage the display of this system check by configuring the `SystemCheck.Enabled` initialization parameter for the `adfdiExcelDownload` filter, as described in [How to Manage the Display of the System Check to End Users](#).

Configuring the ADF Library Filter for ADF Desktop Integration

Using a model-driven list picker requires you to configure the filter for ADF Library Web Application Support (`<filter-name>ADFLibraryFilter</filter-name>`) for your web application.

For information about using a model-driven list picker, see [Adding a Model-Driven List Picker to an ADF Table Component](#).

You configure an entry in the deployment descriptor file (`web.xml`) of your Fusion web application so that the application references the ADF Library Filter.

To configure the ADF Library Web Application Support 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 then click the **Add** icon to create a row entry in the Filters table.

Enter the values as described in [Table D-10](#) to configure the ADF Library Web Application Support filter.

Table D-10 Properties to Configure the ADF Library Web Application Support Filter

For this property...	Enter this value...
Name	<code>ADFLibraryFilter</code>
Class	<code>oracle.adf.library.webapp.LibraryFilter</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 by entering values as described in [Table D-11](#).

Table D-11 Properties to Configure Filter Mappings

For this property...	Enter this value...
Mapping Type	URL Pattern
Mapping	/*
Dispatcher Type	No value is required for this property.

4. 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, as demonstrated in [Example D-1](#), so that integrated Excel workbooks can be downloaded from the Fusion web application.

Example D-1 Entries in web.xml File for ADF Library Web Application Support

```
<filter>
  <filter-name>adfdiExcelDownload</filter-name>
  <filter-class>oracle.adf.desktopintegration.filter.DIExcelDownloadFilter</filter-class>
</filter>
....
<filter>
  <filter-name>ADFLibraryFilter</filter-name>
  <filter-class>oracle.adf.library.webapp.LibraryFilter</filter-class>
</filter>
.....
```

Examples in a Deployment Descriptor File

Shows extracts from the `web.xml` file of a Fusion web application with ADF Desktop Integration.

For information ordering of filters, see [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>
  <filter-name>ADFLibraryFilter</filter-name>
  <filter-class>oracle.adf.library.webapp.LibraryFilter</filter-class>
  <init-param>
    <param-name>include-extension-list</param-name>
    <param-value>png,jpg,jpeg,gif,js,css,htm,html,xlsx,xlsm</param-value>
  </init-param>
</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>
. . .
<filter-mapping>
  <filter-name>ADFLibraryFilter</filter-name>
  <url-pattern>/*</url-pattern>
  <dispatcher>FORWARD</dispatcher>
  <dispatcher>REQUEST</dispatcher>
</filter-mapping>
. . .
<servlet>
  <servlet-name>adfdiRemote</servlet-name>
  <servlet-class>oracle.adf.desktopintegration.servlet.DIRemoteServlet</servlet-class>
</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>
. . .

```

E

Java Data Types Supported By ADF Desktop Integration

Lists the Java data types that an ADF Desktop Integration project supports. This appendix includes the following sections:

- [Primitive Java Types](#)
- [Object Java Types](#)

Note:

Using data types not listed in this appendix will generate errors at runtime.

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`

F

Using the ADF Desktop Integration Model API

Use the ADF Desktop Integration Model API 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 API is for the use cases where you want to allow uploading ADF Table data even when there are no rows available in a tree binding.

This appendix includes the following sections:

- [About the Temporary Row Object](#)
- [About ADF Desktop Integration Model API](#)

About the Temporary Row Object

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.

Each ADF Table component is bound to a tree binding defined within a page definition. Each tree 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.

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.

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 information about the classes and methods available in the API, see *Java API Reference for Oracle ADF Desktop Integration*.

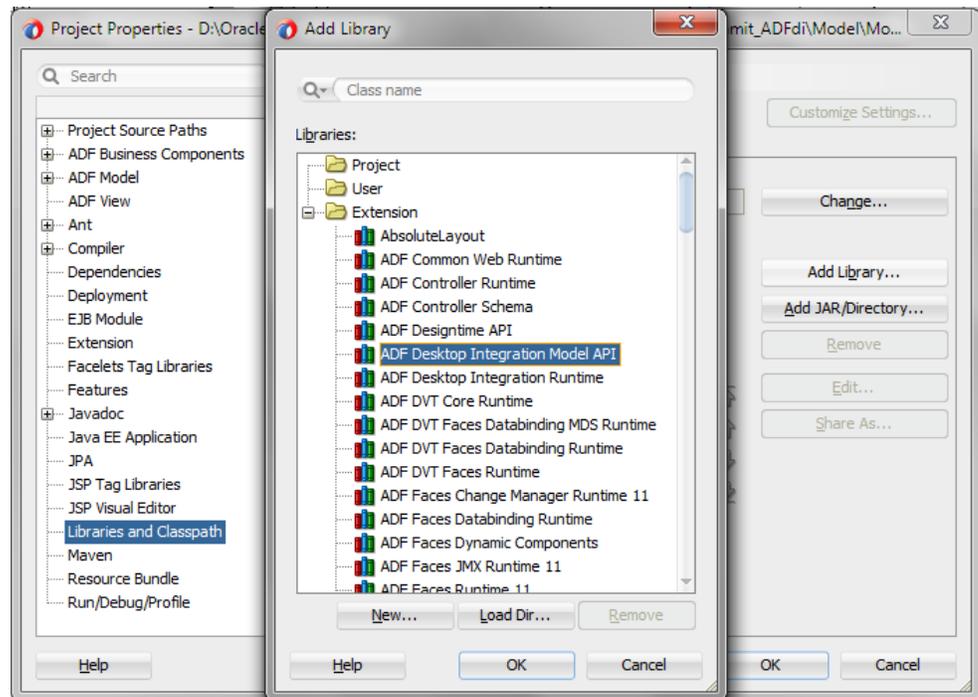
How to Add ADF Desktop Integration Model API Library to Your JDeveloper Project

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 F-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

End User Actions

Describes the actions end users perform while using a Fusion web application and integrated Excel workbook, such as installing ADF Desktop Integration, importing data from non-integrated Excel workbooks, making changes in the workbook at runtime, and handling time zone conversion of date-time values in the workbook. The audience for this appendix are end users of integrated Excel workbooks at runtime. References to "you" in this appendix are directed to these end users.

This appendix includes the following sections:

- [Installing, Upgrading, and Removing ADF Desktop Integration](#)
- [Removing Personal Information](#)
- [Limitations of an Integrated Excel Workbook at Runtime](#)
- [Using an Integrated Excel Workbook](#)
- [Handling Time Zone Conversion](#)
- [Providing Diagnostic and Logging Information to Technical Support](#)

Installing, Upgrading, and Removing ADF Desktop Integration

You can install ADF Desktop Integration by downloading the installer file from the Fusion web application.

Your system administrator can make this installer available as described in [Installing and Upgrading ADF Desktop Integration](#). When the installer runs, it verifies whether the required software is installed on the your system. For information about the required software, see [Required Oracle ADF Modules and Third-Party Software](#).

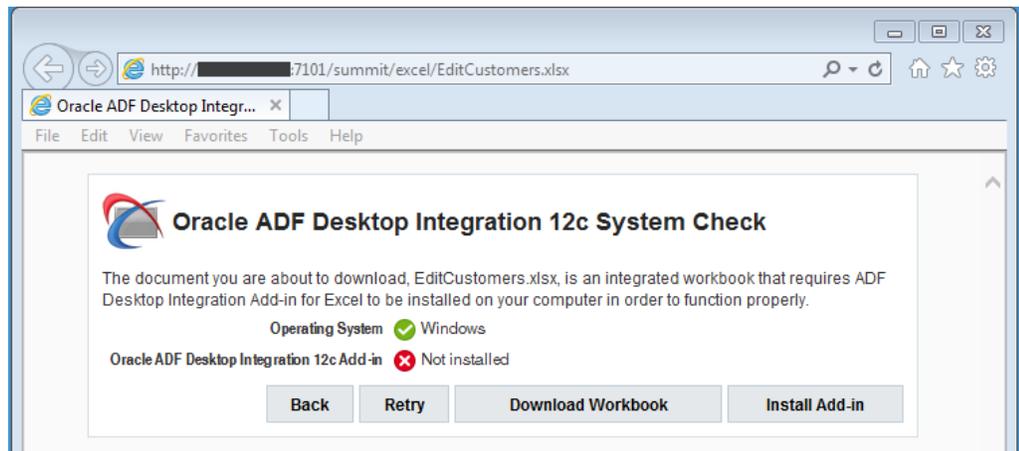
 **Note:**

You do not require JDeveloper to install ADF Desktop Integration.

How to Install ADF Desktop Integration on Your System

The first time that you attempt to download an integrated Excel workbook from a Fusion web application, ADF Desktop Integration runs a system check to verify that your system meets the requirements to run integrated Excel workbooks. You can install ADF Desktop Integration by clicking the **Install Add-in** button, as shown in [Figure G-1](#).

Figure G-1 Installing ADF Desktop Integration



If you do not see the system check because, for example, your system administrator has disabled it you can still install ADF Desktop Integration by downloading the installer from the Fusion web application. If you cannot locate the URL, ask your system administrator. The URL has the following format:

```
http://<hostname>:<portnumber>/<context-root>/adfdiRemoteServlet?excel-addin-installer
```

Assume, for example, that a system administrator at your company has deployed the Summit sample application for ADF Desktop Integration to an application server named `acme-corp-intranet` using port 7101. In this scenario, you enter the following URL in your web browser to download the installer file:

```
http://acme-corp-intranet:7101/summit/adfdiRemoteServlet?excel-addin-installer
```

To install ADF Desktop Integration:

1. After downloading the `adfdi-excel-addin-installer.exe` file, run it.
2. In the installation wizard, click **Install**.

Follow the instructions that appear in the dialog boxes to successfully install the required components. If you encounter an error during the installation process, an error message with a description appears, and installation is rolled back. For more details, check the `adfdi-installer-log.txt` error log file in the `temp` directory of the user profile.

3. Click **Close**.

How to Remove ADF Desktop Integration

Use the Microsoft Windows Control Panel to remove 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 **Oracle ADF Desktop Integration 12c Add-in for Excel** in the Uninstall or change a program window, and click **Uninstall**.

 **Note:**

The specific steps may vary depending on the version of Windows used. Please refer to the Windows documentation for more details.

How to Upgrade ADF Desktop Integration On a Local System

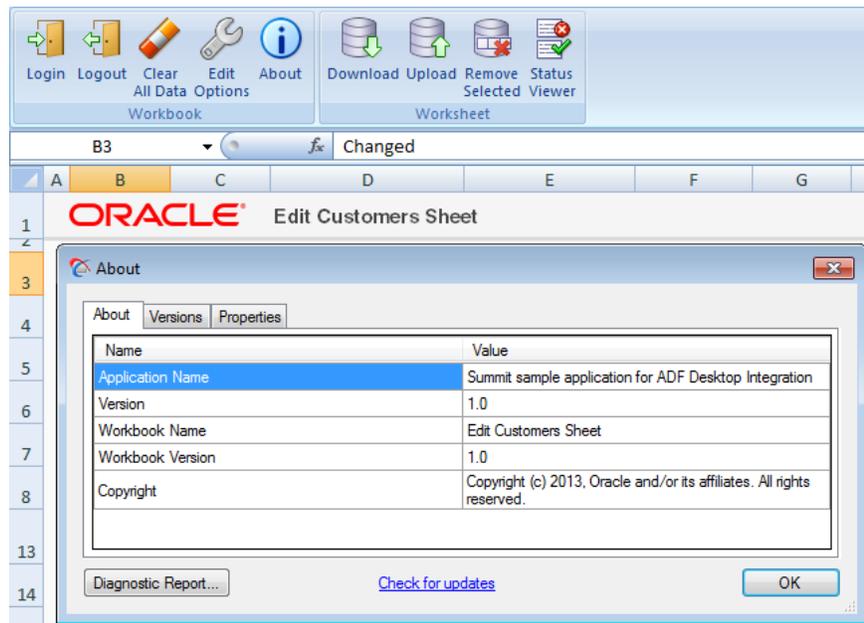
When you establish a connection with the Fusion web application from the runtime integrated Excel workbook, ADF Desktop Integration verifies whether a new version of the add-in is available on the server. If a new version is available, a message appears asking you to upgrade to the newer client version. You can skip this upgrade for the current session and choose to be reminded at a later time that you select from the Remind Me Later dropdown list (for example, Next Week).

 **Note:**

Using the newest client version available from the server is highly recommended to avoid unexpected behavior or errors in integrated Excel workbooks.

You can check for a newer version of the client at any time. To do this, you establish a session with the Fusion web application and then click the **Check for updates** link in the About dialog of the integrated Excel workbook, as shown in [Figure G-2](#). A dialog then appears that shows you the current client and server versions. The dialog also allows you to install a newer client version, if available, from the server.

Figure G-2 Check for Updates Link



For information about the upgrade process, see [Verifying the Client Version of ADF Desktop Integration](#).

Removing Personal Information

If the Fusion web application that the application developer integrated 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 you access the integrated Excel workbook.

You can remove this information using Microsoft Internet Explorer. You 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.

Limitations of an Integrated Excel Workbook at Runtime

There are some known limitations on changing ADF Desktop Integration components at runtime.

- Avoid this operation because using Excel cut-insert operations on worksheet columns that render in an ADF Table component may produce unexpected results during subsequent interaction with the component.
- Avoid moving or deleting columns that render in an ADF Table component's group of columns as this may produce unexpected results and/or affect the grouping of columns.

Additional known limitations:

- Excel's Conditional Formatting feature cannot be used effectively with ADF Desktop Integration table components.
- You should not sort tables containing dependent lists of values.
- You cannot use Microsoft Excel's Undo or Redo commands to undo or redo changes made while using ADF Desktop Integration.
- Excel's Track Changes and Share Workbook features are not compatible with ADF Desktop Integration. You cannot use these Excel features with integrated Excel workbooks.
- If you see a message while viewing a web page in a popup dialog that your session or page has timed out or expired, close the popup dialog without completing the action and then retry the action.

Using an Integrated Excel Workbook

Describes common actions to be aware of if you are new to the ADF Desktop Integration technology and integrated Excel workbooks.

- 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.
- 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 invoke this double-click action by displaying a context menu from where you select **Invoke Action**. To display the context menu, right-click in the cell or, alternatively, select the cell and press Shift + F10 on your keyboard.
- 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.
- When you try to close the integrated Excel workbook, Microsoft Excel prompts you with a dialog to save the workbook even if you have not modified it after opening it. This behavior is expected because ADF Desktop Integration modifies an integrated Excel workbook each time you open it.

Some common actions, such as inserting or deleting a row, and sorting data in ADF Table, are described in the subsequent sections.

How to Insert or Paste Rows in an ADF Table Component

To insert rows in the middle of an ADF Table component, insert a full row or rows in the worksheet, and add data in all mandatory columns. For information, see [Inserting Data in an ADF Table Component](#).

Data that you manage in another Excel workbook (for example, a non-integrated Excel workbook) can also be pasted into an ADF Table component.

To paste data from another worksheet into ADF Table component rows:

1. Arrange the data in the Excel workbook from which you plan to copy the data to match the layout of the ADF Table component in the integrated Excel workbook.

For example, if the first column in the ADF Table component where you want to enter data is Column D, make Column D the first column where you arrange data in the Excel workbook. Also, make sure to provide data for all mandatory columns that the ADF Table component specifies.
2. In the Excel workbook, copy the rows of data.
3. To paste the copied rows into the middle of an ADF Table component:
 - a. Select the entire row above which you want to paste the data from the Excel workbook.
 - b. With the row selected, right-click and choose **Insert Copied Cells**.
 - c. In the Insert Paste dialog that appears, select **Shift cells down**.
4. To paste the copied rows after the last row of an ADF Table component:
 - a. Select the entire row above which you want to paste the data from the Excel workbook.
 - b. With the row selected, right-click and choose **Insert Copied Cells**.

To insert a row in an ADF Table component between the header and last row:

1. In the ADF Table component, 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.

To insert rows in an ADF Table component after the last row:

1. Type data in an empty row immediately after the last row in the ADF Table component.

The ADF Table component automatically converts the edited row to a row in the ADF Table component.

 **Note:**

- If the ADF Table has no data rows, the first row under the column header row acts as a placeholder data row.
- You cannot enter data directly under the table's data rows if ADF Desktop Integration worksheet protection is enabled (`Protection.Mode` property set to `Automatic`), as described in [Using Worksheet Protection](#). You can disable this protection for individual cells or rows by clearing the **Locked** checkbox in the Protection page of Excel's Format Cells dialog that you access from the Format Cells context menu.

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.

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 information about row flagging, see [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.

Handling Time Zone Conversion

Integrated Excel workbooks can be configured 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 you 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, assume you are in Arizona (GMT -07:00) and you download data from the server to the integrated Excel workbook, edit the date-time data in the workbook, save the data, but do not upload it. Later, you travel to Seoul and change the time zone preference of your computer to GMT +09:00. When you open the workbook after changing to the Seoul time zone, you receive 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).

Providing Diagnostic and Logging Information to Technical Support

ADF Desktop Integration provides a number of features and tools to assist you in generating information that may resolve or assist in the resolution of technical issues you encounter while using an integrated Excel workbook.

These include:

- Client Health Check tool that determines if your environment is configured correctly and, in some cases, offers you the opportunity to fix problems it identifies. It also produces a client health check report that you can save to a location you choose.
- Logging information that ADF Desktop Integration automatically captures as you use integrated Excel workbooks.
- Diagnostic reports and a verbose log report file that you can generate from an integrated Excel Workbook.

The client health check report, default log files, verbose log file and diagnostic report may contain information that may assist your Technical Support department in resolving issues you encounter. Before you send these reports to a third party, review to remove any sensitive information that you do not want to share.

Using the Client Health Check Tool

Use the Client Health Check tool to determine if your environment is configured correctly to use integrated Excel workbooks and ADF Desktop Integration. The tool is

an executable (.EXE) that you download from an ADF Desktop Integration-enabled Fusion web application. If you cannot locate the download URL, ask your system administrator. The URL has the following format:

```
<protocol>://<hostname>:<portnumber>/<context-root>/adfdiRemoteServlet?  
excel-addin-health-check
```

Assume, for example, that a system administrator at your company has deployed the Summit sample application for ADF Desktop Integration to an application server named `acme-corp-intranet` using port 7101. In this scenario, you enter the following URL in your web browser to download the Client Health Check tool:

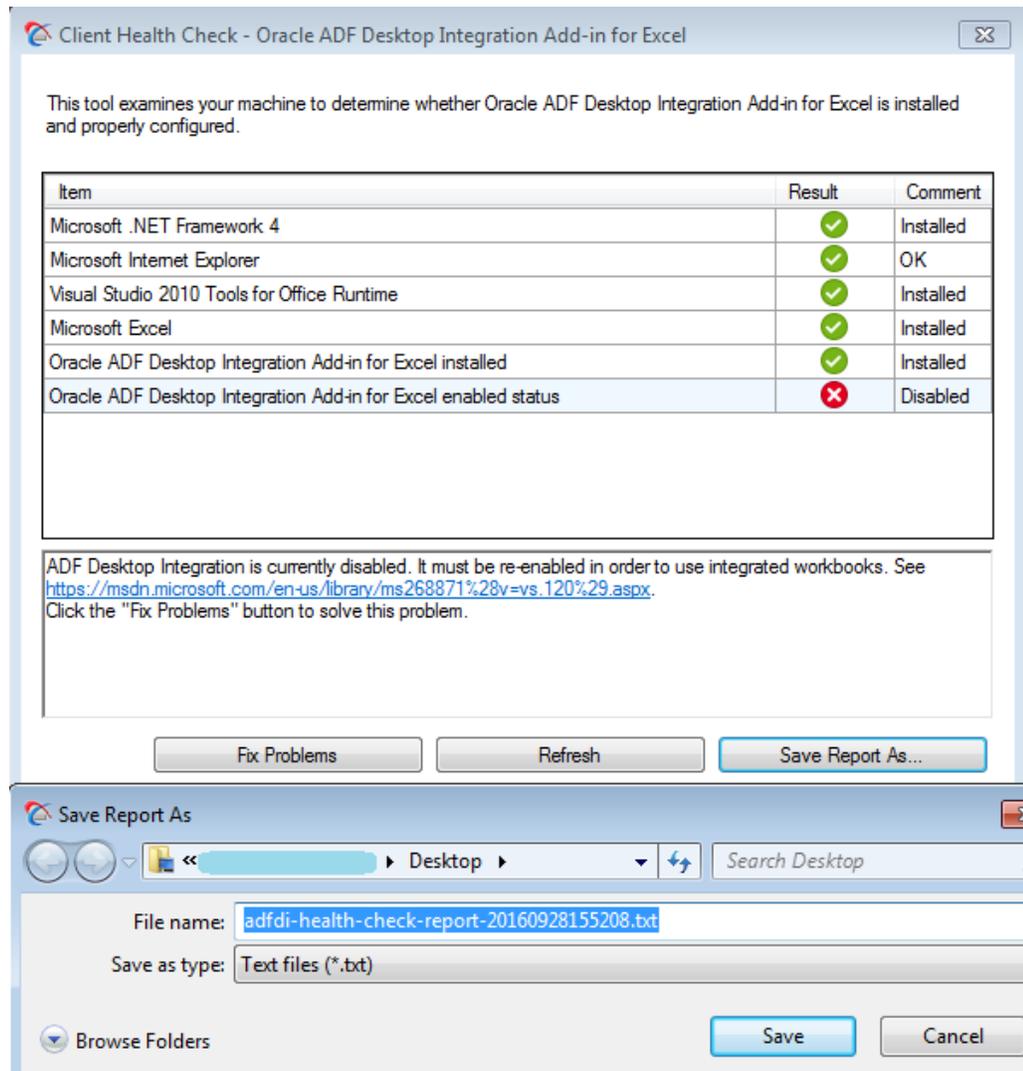
```
http://acme-corp-intranet:7101/summit/adfdiRemoteServlet?excel-addin-  
health-check
```

You can also download the tool by clicking the **Run client health check tool** link that appears in `<protocol>://<hostname>:<portnumber>/<context-root>/adfdiRemoteServlet`.

After you download the `ClientHealthCheck.exe` file, run it to generate a report for your environment.

[Figure G-3](#) shows a Client Health Check tool where the end user can click **Fix Problems** to enable the add-in that is currently disabled plus save a report that the Client Health Check tool generates.

Figure G-3 Client Health Check Tool



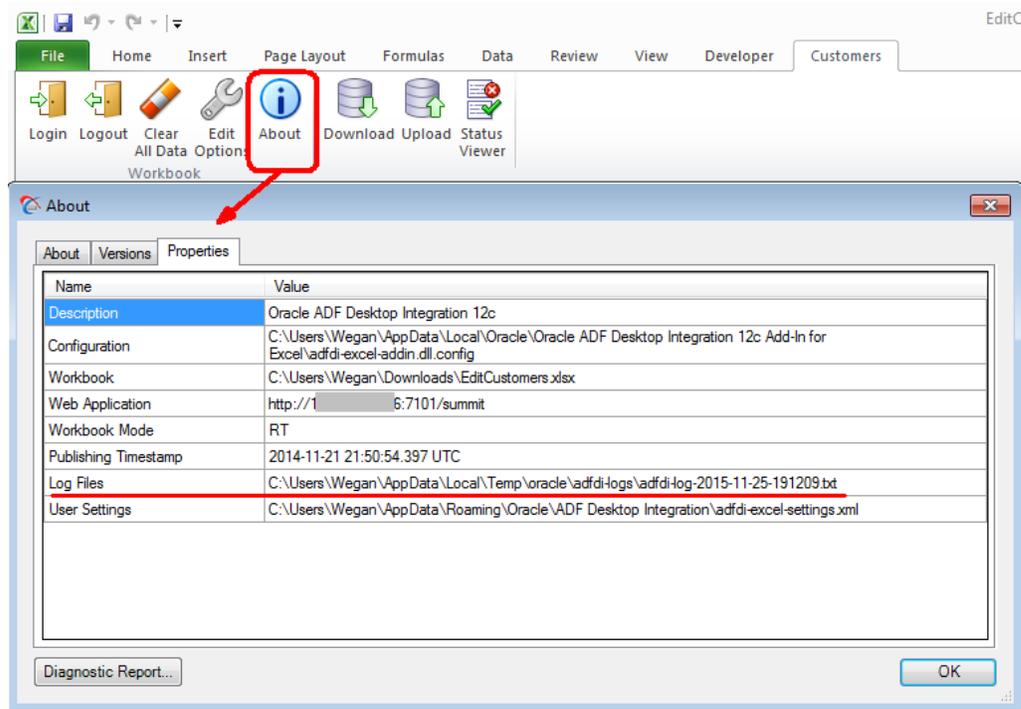
Locating the Log Files that Integrated Excel Workbooks Automatically Generate

By default, ADF Desktop Integration creates a log file in a directory on your machine each time that you open and use an integrated Excel workbook. This log file captures information about the integrated Excel workbook, such as the filename of the workbook, and the actions that you complete using the workbook. ADF Desktop Integration uses the convention `adfdi-log-timestamp.txt` when naming these log files where *timestamp* is the time at which you open the integrated Excel workbook. The following list shows a number of log files that ADF Desktop Integration created in a directory:

```
adfdi-log-2015-11-25-191209.txt
adfdi-log-2015-11-25-214526.txt
adfdi-log-2015-11-25-185635.txt
adfdi-log-2015-11-24-140541.txt
adfdi-log-2015-11-24-103238.txt
adfdi-log-2015-11-23-232952.txt
```

To determine which directory ADF Desktop Integration uses to store log file information on your machine, see the Log Files property that you can view from the About dialog of an integrated Excel workbook, as shown in [Figure G-4](#).

Figure G-4 Directory Location of Log Files



Generating Diagnostic Report and Verbose Log File

The diagnostic report is a text file (.TXT) that contains various pieces of information, such as your Microsoft Windows version and ADF Desktop Integration add-in version. The verbose log file is an XML file that captures more detailed information than the default log files about the actions that you perform, and the events that occur while you use the integrated Excel workbook. If you encounter an issue, enable logging and redo your task to reproduce the issue. ADF Desktop Integration logs the information to the log file that you created when you enabled logging. Once you exit Microsoft Excel, ADF Desktop Integration stops verbose logging until you enable it once again using the **Enable Logging** menu option shown in [Figure G-5](#).

You generate the diagnostic report and enable logging from the same menu. The location of this menu depends on the version of Microsoft Excel that you use. [Figure G-5](#) shows the location of this menu entry in Microsoft Excel 2013.

Figure G-5 Menu to Enable Logging and Save Diagnostic Report

