



SIEBEL INTERACTIVE DESIGNER ADMINISTRATION GUIDE

VERSION 7.5, REV. A

12-DX88X3

NOVEMBER 2002

Siebel Systems, Inc., 2207 Bridgepointe Parkway, San Mateo, CA 94404
Copyright © 2002 Siebel Systems, Inc.
All rights reserved.
Printed in the United States of America

No part of this publication may be stored in a retrieval system, transmitted, or reproduced in any way, including but not limited to photocopy, photographic, magnetic, or other record, without the prior agreement and written permission of Siebel Systems, Inc.

The full text search capabilities of Siebel eBusiness Applications include technology used under license from Hummingbird Ltd. and are the copyright of Hummingbird Ltd. and/or its licensors.

Siebel, the Siebel logo, TrickleSync, TSQ, Universal Agent, and other Siebel product names referenced herein are trademarks of Siebel Systems, Inc., and may be registered in certain jurisdictions.

Supportsoft™ is a registered trademark of Supportsoft, Inc. Other product names, designations, logos, and symbols may be trademarks or registered trademarks of their respective owners.

U.S. GOVERNMENT RESTRICTED RIGHTS. Programs, Ancillary Programs and Documentation, delivered subject to the Department of Defense Federal Acquisition Regulation Supplement, are “commercial computer software” as set forth in DFARS 227.7202, Commercial Computer Software and Commercial Computer Software Documentation, and as such, any use, duplication and disclosure of the Programs, Ancillary Programs and Documentation shall be subject to the restrictions contained in the applicable Siebel license agreement. All other use, duplication and disclosure of the Programs, Ancillary Programs and Documentation by the U.S. Government shall be subject to the applicable Siebel license agreement and the restrictions contained in subsection (c) of FAR 52.227-19, Commercial Computer Software - Restricted Rights (June 1987), or FAR 52.227-14, Rights in Data—General, including Alternate III (June 1987), as applicable. Contractor/licensor is Siebel Systems, Inc., 2207 Bridgepointe Parkway, San Mateo, CA 94404.

Proprietary Information

Siebel Systems, Inc. considers information included in this documentation and in Siebel eBusiness Applications Online Help to be Confidential Information. Your access to and use of this Confidential Information are subject to the terms and conditions of: (1) the applicable Siebel Systems software license agreement, which has been executed and with which you agree to comply; and (2) the proprietary and restricted rights notices included in this documentation.

Contents

Introduction

How This Guide Is Organized	12
Revision History	13

Chapter 1. Interactive Designer and Interactive Selling Applications

Interactive Designer Overview	16
Siebel eAdvisor and Browser-Based eConfigurator Application Overview .	17
Interactive Designer Application Architecture	18
Siebel Web Client Deployment	20
Siebel Dedicated Web Client Deployment	20
Global Deployment	20
Performance Considerations	22
Siebel eConfigurator Deployment Methods	23

Chapter 2. Building an Application Using Interactive Designer

Siebel eAdvisor	26
Browser-Based eConfigurator	27
Creating an eAdvisor Application	28
Creating a Browser-Based eConfigurator Application	31

Chapter 3. Working in Interactive Designer

Opening Interactive Designer	34
Project Administration Screen	35

Pagesets View	35
Contents List View	36
Project Files View	37
Validation Results View	37
Search View	38
Pageset Administration Screen	39
Feature Table Editor View	39
Feature Table Designer View	41
Configuration Table Editor View	42
Configuration Table Designer View	43
Input UI View	44
Output UI View	44
Pageset Files View	45

Chapter 4. Working with Projects

About Projects	48
Viewing Your Projects	48
Creating a Project	50
Migrating a Project	51
Copying a Project	51
Deleting a Project	52
Editing a Project	52
Performing a Project Data Search	53
Validating a Project	54
Previewing a Project	55
Deploying a Project	57
Exporting and Importing a Project	58
Projects in a Team Environment	60
View Access	60
Team Access	61

Locking Pagesets 61
 Recording the Release Number 62

Chapter 5. Working with Configuration Tables

About Configuration Tables 64
 The Configuration Matching Process 65
 Configuration Column Types 68
 Input Columns 68
 Output Columns 69
 Subtable Columns 70
 Cell Functions 71
 About Referring to Other Table Columns 71
 About Performing Calculations 71
 Nested Cell Functions 72
 Example 73
 Range Functions 74
 Viewing Configuration Tables 75
 Creating Configuration Tables 75
 Creating Configuration Subtables 75
 Designing the Configuration Table 76
 Entering Data in the Configuration Table 79
 Copying a Configuration Table 80
 Deleting a Configuration Table 81
 Editing a Configuration Table 82
 Creating Exception Messages 83
 Creating Cross-Sell and Up-Sell Messages 84

Chapter 6. Working with Pagesets

About Pagesets 88
 About Managing the Display of Pagesets 89

Viewing Your Pagesets	90
Locking Pagesets	91
Creating a Pageset	92
Associating a Pageset with a Product	93
Copying a Pageset	94
Deleting a Pageset	95
Editing a Pageset	95
Validating a Pageset	96
Performing a Pageset Data Search	97

Chapter 7. Working with Feature Tables

About Feature Tables	100
Viewing Feature Tables	103
Creating Feature Tables	103
Creating Linked Feature Tables	109
Copying a Feature Table	110
Deleting a Feature Table	111
Editing a Feature Table	111
Managing Feature Table Columns	112

Chapter 8. Advanced Modeling

About Trigger and Target Feature Tables	116
Creating Trigger and Target Feature Tables	118
Creating Target Tables	118
Creating Trigger Tables	119
Trigger and Target Example	120
Additional Trigger Capabilities	123
Dynamic Defaults	124
Creating Dynamic Defaults	124

Working with Subconfiguration	128
Example	128
About Referencing Feature Tables	129
About Setting Defaults	130
About Accessing Model Variables	132
Performance Considerations	133
Creating Javascript Conditional Statements	134
Example	135
Duplicate Configuration Column Names	135

Chapter 9. Building Your UI

About Building Your UI	138
Contents List	138
Input UI Display Page	139
Output UI Display Page	139
About Creating Input UI Controls	140
Creating Input UI Controls	141
Copying Input UI Controls	145
Deleting Input UI Controls	145
Editing Input UI Controls	146
Generating Your Input UI Display Page	146
About Creating Output UI Controls	147
Creating Output UI Controls	148
Copying Output UI Controls	150
Deleting Output UI Controls	151
Editing Output UI Controls	152
Generating Your Output UI Display Page	153

Chapter 10. Working with the Contents List

About the Contents List156
 Customizing Get Advice Button Functionality 158
About Populating the Default Contents List158
Viewing the Contents Lists158
Creating a Contents List159
Creating Contents List Items160
Copying a Contents List161
Deleting a Contents List162
Editing a Contents List163

Chapter 11. Customizing Your UI

About Customizing Your UI166
Editing the Project UI Files167
About the Project UI Files168
 Modifying the Application UI Definition File 169
About Modifying the Contents List Location170
 In the Application Definition File 170
 In a Pageset UI Definition File 171
About the UI Controls172
 Input UI Display Example 172
 Output UI Controls Example 173
Using UI Templates174

Chapter 12. Referencing Other Siebel Data

About Referencing Other Siebel Data176
About Binding to Siebel Business Components177
 How the Binding Works 177
 Configuration Table Designer Example 178

Adding Access to Additional Business Components	179
About Modeling for Customizable Products	181
Data Evaluation in Feature Tables	181
Data Evaluation in Configuration Tables	182
Evaluation of the Customizable Product Structure	182
Automatic Creation of the Customizable Product Structure	183
Mapping Root Products in the Configuration Table	184
Mapping Root Product Attributes in the Configuration Table	186
Mapping Root Product Attributes in Feature Tables	187
Mapping Child Products in Feature Tables	187
Mapping Attributes of the Child Product in Feature Tables	189
Best Practices	189
Runtime Interaction with Your Shopping Cart or Quote	191
Runtime Interaction with the Server-Based eConfigurator	193
Runtime Access to Your Pricing Information	198
About Publishing Pricing Information in Pagesets	199
Associating a Price List with Your Browser-Based Model	200
About Modifying Display Information in app_config.js	202
About Modifying the Application Configuration File	203
Adding Rules Based Pricing	203

Chapter 13. Working with a Deployed Application

Running in Stand-Alone or Standard Mode	212
Calling Your Application from Another Siebel Application	213
Referencing Pagesets from Customizable Products	214
Invoking the ShowCDA Method from a Button	215
About Passing in Parameters When Invoking the ShowCDA Method	216
Synchronization Setup	220
Modifying the Siebel Synchronize Database Behavior	220
Using the Synchronize CDA Projects Screen	223
About Modifying the Project Synchronization Behavior	224

About Working in Mobile Client Mode225

About Unframing Customer and Partner Applications226

 Limitations of Unframing 226

 Implementing a Customer and Partner Application Without Frames 227

Chapter 14. Working with the Project Files

About the Project Files230

 Interactive Designer Project Structure 230

 The Project Files Tab 231

Viewing the Project Files233

Creating a File Attachment234

Copying a File Attachment235

Deleting a File Attachment236

Editing a File Attachment236

Appendix A. Implementation of Multi-Variable and Cascading Triggers

Installation of the Multi-Variable and Cascading Triggers Module238

Index

Introduction

The purpose of this guide is to describe how to author and maintain Siebel eAdvisor and browser-based eConfigurator applications. These applications are intended for users who need advice and guidance to configure products, create quotes, or purchase products from Siebel Customer and Partner Application Web sites.

Although job titles and duties at your company may differ from those listed in the following table, the audience for this guide consists of employees in these categories:

Model Developers	Those responsible for designing and maintaining product configuration models.
Product Managers	Those responsible for the marketing decisions and overall direction of a product or product family.

All users should have detailed knowledge of their company's products and how they are sold. Before using Interactive Designer, you should also be familiar with the Siebel eBusiness Applications user interface and its basic operations. You should know how to accomplish basic tasks such as navigating between screens and views, selecting from lists, and entering information.

How This Guide Is Organized

This guide is organized to help you build applications using Interactive Designer. The following is a suggested path for working with this guide.

- 1** Read [Chapter 1, “Interactive Designer and Interactive Selling Applications”](#) for an understanding of how Interactive Designer is related to and works with other Siebel applications in Siebel Interactive Selling. This will give you a high level view of the technology you may choose to integrate your application with.
- 2** Read [Chapter 2, “Building an Application Using Interactive Designer.”](#) This chapter helps you determine which application you want to create. It also provides the steps to build that application with references to the relevant sections in this guide.
- 3** Refer to the relevant sections in this guide to complete the necessary steps to develop your application.
- 4** For more information on application development, refer to *Siebel Interactive Designer API Reference*.

For users creating stand-alone applications, first read [“Running in Stand-Alone or Standard Mode” on page 212](#) for an understanding of your development tasks. Except for [Chapter 12, “Referencing Other Siebel Data,”](#) read all of this guide.

NOTE: Your Siebel implementation may not have all the features described in this guide, depending on which software modules you have purchased.

Revision History

Siebel Interactive Designer Administration Guide, Version 7.5, Rev. A

November 2002 Bookshelf

Book Version: Rev A

Topic	Revision
“To create input UI controls” on page 141	Added procedural step 11 to describe how to supply the Map Filename, Map Shape, and Map Coordinates columns for image maps.
“To create output UI controls” on page 148	Added control types in step 5 of the procedure.
“About the Contents List” on page 156	Corrected the description.
“The Project Files Tab” on page 231	Corrected the description.

Introduction

Revision History

Interactive Designer and Interactive Selling Applications

1

This chapter introduces Interactive Designer and describes how it fits into Siebel Interactive Selling applications. This chapter describes an eAdvisor and a browser-based eConfigurator. It also describes the browser-based application architecture and the Interactive Designer administration environment for creating browser-based applications.

Interactive Designer Overview

Interactive Designer is included with any of the following applications:

- Siebel eAdvisor
- Browser-based eConfigurator

Interactive Designer is the administration environment used to author and maintain interactive selling applications. Use Interactive Designer to create projects which are the high-level containers from which all aspects of the browser-based application are referenced. Projects reference the tables you design and edit with product information and additional selling knowledge. You can enter product data in the tables or reference existing business components and their fields in the Siebel database. After you deploy the project, it is referred to as one of two kinds of applications: eAdvisor or browser-based eConfigurator.

Siebel eAdvisor and Browser-Based eConfigurator Application Overview

The primary purpose of applications built with Interactive Designer is to facilitate an organization's sales process by providing customers with targeted product and service information, as well as guidance and advice to assist customers in deciding which products and services are needed.

Interactive Designer applications use a Web browser to present product specifications, prices, images, schematics, applicability guidelines, and other information that customers require to make product selection decisions.

The functionality of any Interactive Designer application includes support for the following user activities:

- Recommending products, services, or a course of action based on user needs
- Selecting and configuring products based on product features and user requirements

Interactive Designer generates the JavaScript, HTML, and image files needed for an eAdvisor or browser-based eConfigurator application. The applications run inside other Siebel applications as part of a Siebel application or as stand-alone applications.

When a user opens an eAdvisor or browser-based eConfigurator application, the application files and the engine that manages them are downloaded from the Web server and individual files containing code, data, and user interface definitions are loaded in the user's browser as needed.

To produce the JavaScript and HTML files, Interactive Designer uses feature and configuration information you enter about your product line. You also enter basic information about how you want your application to appear.

After the application is defined, you can extend your Interactive Designer applications with other solutions, such as HTML editors, to further customize the appearance and behavior of your application.

Interactive Designer Application Architecture

In eAdvisor and browser-based eConfigurator applications, the engine runs on the browser rather than on the server, providing users with quick response time. Because trips to the server are reduced, there is no network latency.

Application data is stored in the Siebel database and Siebel File System. Unlike in server-based applications, in eAdvisor and browser-based eConfigurator applications, files needed for the application are statically published after the application data model and the user interface are developed. In some deployments, you will need to take the additional step of moving your application files to a Web server where the browser can find them. The data accessed from the Siebel business components is also statically published. You can use the Siebel Application Integration functions to dynamically access data in the business components. For more information about integrating your data, see [“Referencing Other Siebel Data” on page 175](#) and *Siebel Interactive Designer API Reference*.

Figure 1 illustrates eAdvisor and browser-based eConfigurator application architecture.

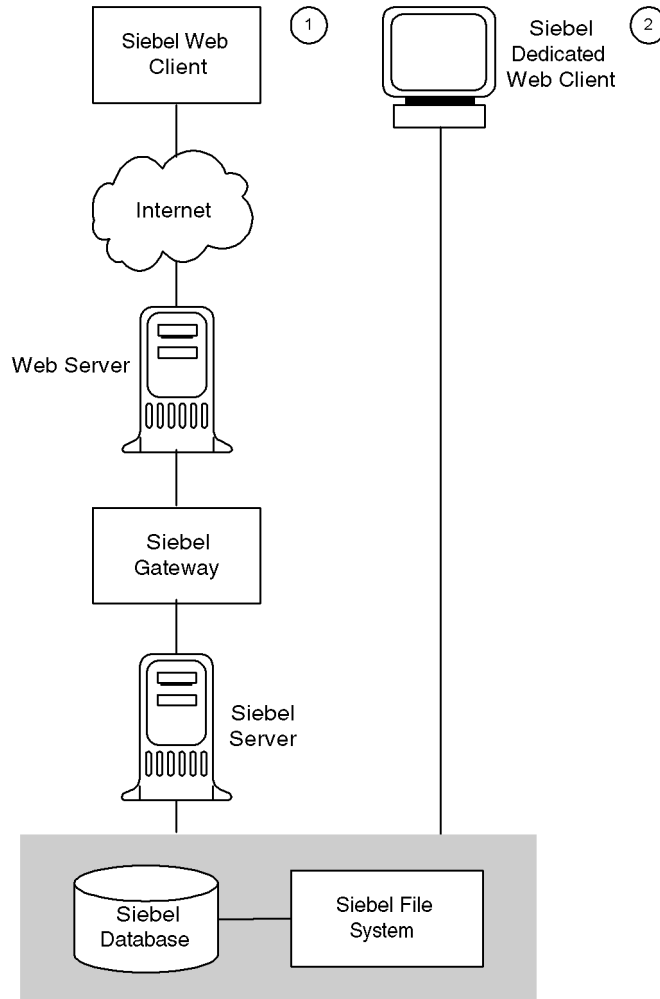


Figure 1. Siebel eAdvisor and Browser-Based eConfigurator Application Architecture

[Figure 1 on page 19](#) shows two types of deployment:

- Siebel Web Client
- Siebel Dedicated Web Client

Siebel Web Client Deployment

In [Figure 1 on page 19](#), the Siebel Web Client deployment path is marked 1. As you create your application in Interactive Designer, your data is saved in the Siebel database and Siebel File System. When you are finished creating your application and are ready to make it available to users, you need to deploy the run-time files to the Web server. You also need to modify the configuration file for the Siebel application you are using to point to the location of these files. For more information, see [“Deploying a Project” on page 57](#).

Siebel Dedicated Web Client Deployment

In [Figure 1 on page 19](#), the Siebel Dedicated Web Client deployment path is marked 2. In this setup, the Web client, Web server, and Siebel server are on the same machine. You work in Interactive Designer on the Dedicated Web Client. You save your data to the Siebel database and Siebel File System. When you are ready to deploy the application, you deploy your run-time files on the client. There is no need to update the Siebel configuration file in this scenario.

Global Deployment

When you are deploying your application for multiple languages, a Siebel server, Siebel Web server, and Siebel Web client are added to your setup for each language. All servers can reference the same Siebel Unicode database. You will need to modify the Siebel configuration file for each language you are using to point to the location of the application files. For more information, see [“Deploying a Project” on page 57](#).

Figure 2 illustrates the global deployment architecture for an eAdvisor or browser-based eConfigurator application.

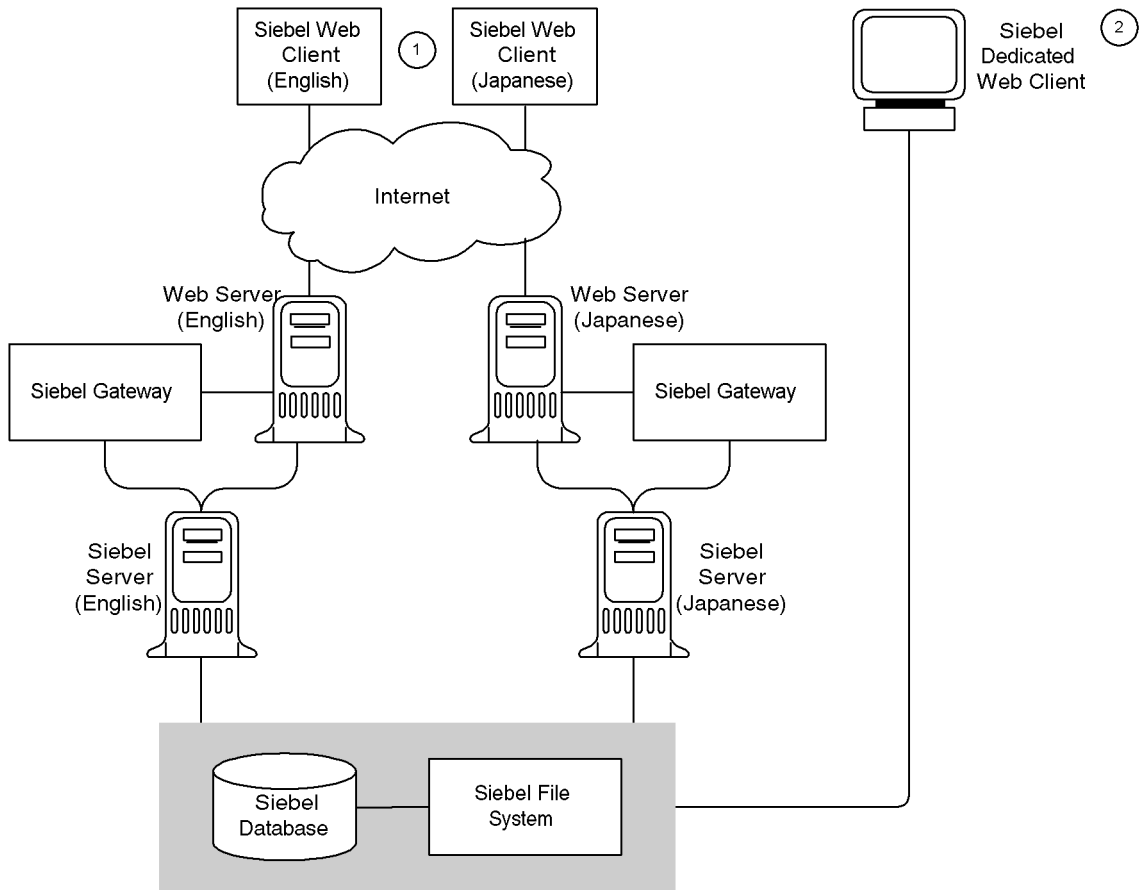


Figure 2. Global Deployment Architecture

Performance Considerations

Use the following guidelines to improve the performance of your application.

Project Size

When determining how you will break up your pagesets across projects, be aware that smaller projects load faster. Consider creating a number of smaller projects and linking to pagesets across projects as needed.

Pageset Size

When determining how you will break up your products across pagesets, be aware that smaller pagesets load faster. The combined file sizes for the _1 and _2 files (in the pg directory) and _00 and _m files (in the ds directory) should not exceed 150K for a pageset. A combined size of 50K or less is preferred.

Table Size

When determining how you will break up your tables, be aware that smaller tables execute faster and are easier to maintain. Consider creating a number of smaller tables, limiting the number of rows to less than 200 per table.

Number of Attached Files

Attached files increase the amount of time some Interactive Designer operations take. Migrate, Import/Export, Deploy, Copy Project, and Preview can take between 0.25 and 0.50 seconds longer per attached file. To improve operation time, keep your attached files to a minimum.

Number of Messages

If the same message is used under a variety of circumstances, consider defining the message once in a separate Feature table, then setting that value from the selected Feature table. This reduces the maintenance (changing the same message in multiple places) and reduces the file size (increasing performance).

Siebel eConfigurator Deployment Methods

Siebel offers two deployment options for eConfigurator applications:

- Browser-based, administered through Interactive Designer

This method uses Configuration tables to describe relationships, and delivers a subset of configuration capabilities directly to the end user's browser, using only JavaScript and HTML.

- Server-based, administered through Customizable Products

This method works by solving simultaneous constraints to make sure the solution is accurate, with all data and constraint processing occurring at the server.

For more information about using server-based eConfigurator, refer to *Product Administration Guide*.

Building an Application Using Interactive Designer

2

This chapter describes each application you can build with Interactive Designer, and outlines the steps to get you started on each application.

Siebel eAdvisor

Use the eAdvisor application to involve customers in an interactive dialogue to understand their needs and recommend appropriate solutions as an initial point in the buying process. Siebel eAdvisor allows customers to base their decision on their needs, without extensive product knowledge.

The eAdvisor applications open from a link in another application. For example, a sales application in which a user can configure and purchase a car might include a Need Help? link. When the user clicks Need Help?, the eAdvisor application opens, displaying a list of questions. For example, if a customer is shopping for a car, eAdvisor might present the following questions:

- What will be the minimum number of passengers in your car?
- What are your storage needs?
- What are your priority considerations (safety, fuel, economy, transmission)?

Based on the user's answers, a product or solution appears that can be added to a Siebel quote or shopping cart, or further configured in eConfigurator. To allow the user to further refine their configuration, an eAdvisor session is often integrated with an eConfigurator application (either browser-based or server-based). The eAdvisor application collects the user's requirements to provide a possible solution. A link to the eConfigurator application is provided for a user to further configure that solution. Users can restore an eAdvisor session from a Siebel quote or shopping cart by clicking the Customize button.

The eAdvisor links are provided in Siebel eCatalog, shopping cart, and quote applications.

For information on creating an eAdvisor application, see [“Creating an eAdvisor Application” on page 28](#).

Browser-Based eConfigurator

The eConfigurator application allows users to configure a product or solution. Users can move back and forth between selection criteria, changing original selections to reach a desirable solution.

The eConfigurator application also acts as a virtual sales consultant. For example, if a user selects a particular product, eConfigurator can recommend a better product for their needs or a product accessory. If a user configures a product, the eConfigurator application can verify that the selection resulted in a valid product.

In addition to the browser-based deployment described in this guide, a server-based deployment of eConfigurator is offered. See [“Siebel eConfigurator Deployment Methods” on page 23](#) to determine which eConfigurator best fits your needs.

For information on creating an eConfigurator application, see [“Creating a Browser-Based eConfigurator Application” on page 31](#).

Creating an eAdvisor Application

Using Interactive Designer, you can create an eAdvisor application for analyzing a user's needs and presenting a solution or configuration.

In eAdvisor, each question is tied to a Feature table. Based on the user's answers to the questions, the Configuration table determines which solution (pageset link) to display. If you tie your eAdvisor application to an eConfigurator application, you can use *dynamic defaults*, in which the UI controls for the configuration display the appropriate default values based on the answers the customer provided in eAdvisor. For more information, see [“Creating Dynamic Defaults” on page 124](#).

Figure 3 shows an example of an eAdvisor application at runtime.

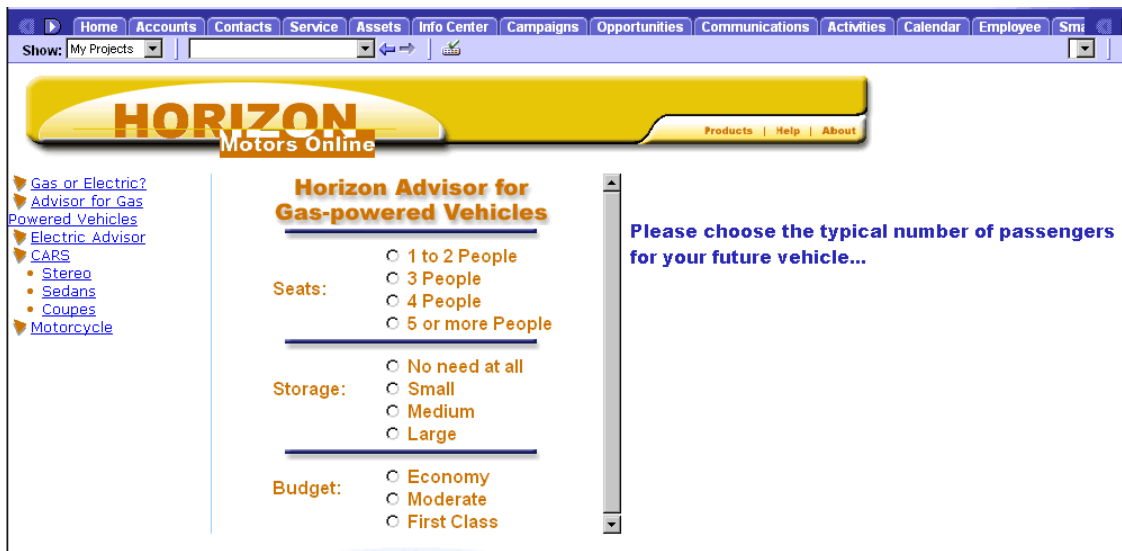


Figure 3. Siebel eAdvisor Application at Runtime

The following workflow describes how to create an eAdvisor application.

- 1 Create a project.

For more information, see [“Creating a Project” on page 50](#).

2 Create a pageset.

For more information, see [“Creating a Pageset” on page 92](#).

3 Design your eAdvisor questions.

Customers should arrive at a solution by answering a few questions (~ 5).

4 Create a Feature table for each question.

For example, a Feature table named PASSENGER might store the choices for the number of passengers. A Feature table named STORAGE could include size requirements and storage examples. For more information, see [“Creating Feature Tables” on page 103](#).

5 Create a Configuration subtable for each recommendation and reference the subtable from a subtable column in the MAIN Configuration table.

For more information, see [“Creating Configuration Tables” on page 75](#).

6 In the Configuration subtable, create an input column for each Feature table (question).**7** Add an output column to the configuration subtable to store pageset IDs for the recommended product or solution.**8** Enter all valid combinations for the eAdvisor questions and the corresponding information for the recommended solution into the configuration subtables.

For information on how to enter configurations and exception and guidance messages in Configuration tables, see [“Creating Configuration Tables” on page 75](#). For information on changing default values in controls based on how a user answered previous questions, see [“Creating Dynamic Defaults” on page 124](#).

9 Create the contents list.

For more information, see [“Creating a Contents List” on page 159](#).

10 Add UI controls for the questions to the Inputs page.

For more information, see [“About Creating Input UI Controls” on page 140](#).

- 11** Add UI controls to the Results page to display information about the recommended solution.

For more information, see [“About Creating Output UI Controls” on page 147](#).

- 12** Add a link labeled “Get My Product” to the Inputs page and a UI control to the Results page to display the configured solution when the user clicks on the link.

- 13** On the Results page, add a link to the recommended solution so that users can further configure it using browser-based technology. The following is an example of a link:

```
<A>
<SCRIPT>
document.write(ISS.BuildTarget("LINK",window,"REC_PROD",true));
</SCRIPT>
Click here to further configure your vehicle.
</A>
```

- 14** Preview the project.

For more information, see [“Previewing a Project” on page 55](#).

- 15** Deploy the application.

For more information, see [“Deploying a Project” on page 57](#).

- 16** Link the application to quotes, the shopping cart, and other Siebel products.

For more information, see [“Working with a Deployed Application” on page 211](#).

Creating a Browser-Based eConfigurator Application

Using Interactive Designer, you can create an eConfigurator application that allows users to configure a product or solution.

An eConfigurator application displays UI controls populated with data contained in Feature tables. As users make selections, rules stored in the Configuration table determine which selections are valid. Guidance messages stored in the Configuration tables are displayed based on the user's selections. For example, if a user selects a red luxury sedan, and the luxury sedan is not available in red, the guidance message appears: "The luxury sedan is available in only black or silver. Please select one of these colors or select a different model." In the Outputs page, you can display the current product or solution based on the user's selections.

Figure 4 shows an example of an eConfigurator application at runtime.



Figure 4. An eConfigurator Application at Runtime

The following workflow describes how to create an eConfigurator application.

- 1 Create a project.

For more information, see “Creating a Project” on page 50.

2 Create a pageset.

For more information, see [“Creating a Pageset” on page 92](#).

3 Create Feature tables.

For more information, see [“Creating Feature Tables” on page 103](#).

4 Add an output column to the MAIN Configuration table or to a subtable, to store product IDs for your products or solutions.

For more information, see [“Creating Configuration Tables” on page 75](#).

5 Add a link UI control to the Results page of the pageset and reference the column that stores product IDs.

6 In the Configuration table, enter all valid and invalid combinations of your features.

7 Create the contents list.

For more information, see [“Creating a Contents List” on page 159](#).

8 Add UI controls for the features to the Inputs page.

For more information, see [“About Creating Input UI Controls” on page 140](#).

9 Add UI controls to the Results page to display information about the product or solution that results from the user’s selections.

For more information, see [“About Creating Output UI Controls” on page 147](#).

10 Preview the project.

For more information, see [“Previewing a Project” on page 55](#).

11 Deploy the application.

For more information, see [“Deploying a Project” on page 57](#).

12 Link the application to quotes, the shopping cart, and other Siebel products.

For more information, see [“Working with a Deployed Application” on page 211](#).

Working in Interactive Designer

3

This chapter provides an example and high-level description of each of the Interactive Designer views. Detailed instructions for working in these views are described throughout the rest of this guide.

Opening Interactive Designer

Interactive Designer is a collection of administrative screens in which you work to create an eAdvisor or browser-based eConfigurator application.

To open Interactive Designer

- 1** From the View menu, choose Site Map.
- 2** Click Interactive Designer.
- 3** Click one of the following:
 - All Pagesets
 - All Projects
 - My Pagesets
 - My Projects

Once you are in Interactive Designer, you can move between these views by using the Show drop-down list.

Project Administration Screen

Use the Project Administration screen to create, view, modify, and delete your application projects. A project is the highest level container. From this screen, you can access your contents list, Project Files, validation and search functionality, as well as the Catalog Wizard (licensed separately). You can also click the Pagesets tab to create, delete, and view the pagesets. Pagesets, contents lists, and catalogs are always associated with a project.

For more information, see [Chapter 4, “Working with Projects.”](#)

Pagesets View

In addition to viewing your pagesets from the Pagesets Administration screen, explained in [“Pageset Administration Screen” on page 39](#), you can view the pagesets associated with the selected project from the Pagesets view.

Contents List View

Use the Contents List view (Figure 5) to create, view, and modify your contents lists. The contents list is a collapsible list used to navigate an application. Click a contents list item in runtime to load the pageset associated with that item.

For more information, see Chapter 10, “Working with the Contents List.”

The screenshot displays two parts of the 'Contents List View' interface. The top part is a table with columns: Name, Contents List ID, Notes, Last Updated, and Last Updated By. The bottom part is an 'Editor' view with columns: Sequence, Level, Label, Link To, and Image Location.

Contents Lists					
More Info		Pagesets		Project Files	
Validation Results		Project Search		1 - 1 of 1	
Query	Edit				
Name	Contents List ID	Notes	Last Updated	Last Updated By	
prodlstdata	prodlstdata		07/24/2001 6:36:33	SADMIN	

Editor				
New		Save		1 - 7 of 7+
Sequence	Level	Label	Link To	Image Location
1	1	Gas or Electric?	FL_ADVISOR	
2	1	Advisor for Gas Pow	GAS_ADVISOR	
3	1	Electric Advisor	ELECTRIC_ADVISOR	
4	0			

Figure 5. The Contents List View

Project Files View

When a project is created, Interactive Designer automatically adds files for that project to the Siebel File System. Use the Project Files view, shown in [Figure 6](#), to add or modify the files that belong to a project.

For information on the files Interactive Designer creates, refer to the Browser-Based Application File Reference chapter in *Siebel Interactive Designer API Reference*.

For information on working in this view, see [Chapter 14](#), “Working with the Project Files.”

Attachment Name	Type	Directory	Modified Date/Time	Notes
home	htm		07/24/2001 6:36:43	
kernel	htm		07/08/2001 2:50:56	
onl_boot	htm		07/08/2001 2:50:56	
back_ui	htm	cs	07/24/2001 6:37:42	
codeset_config	js	cs	07/24/2001 6:37:42	
complexProductCod	htm	cs	07/24/2001 6:37:43	
config	js	cs	07/24/2001 6:37:43	

Figure 6. The Project Files View

Validation Results View

Use the Validation Results view ([Figure 7](#)) to make sure your project model has no errors. When you validate your project, the Validation Results tab displays any errors. Click on an error to find the exact area in the project where the error needs to be fixed.

For more information, see “Validating a Project” on page 54.

Message	Pageset	Table	Column	Sequence
'MAIN' (row '20') : er COUPE		MAIN		20
'MAIN' (row '20', col COUPE		MAIN	MODEL_COLOR	20
'MAIN' (row '20', col COUPE		MAIN	MODEL_TRANS	20
'MAIN' (row '20', col COUPE		MAIN	VALID_SHIP	20
'MAIN' (row '20', col COUPE		MAIN	UPSELL_OPT4	20

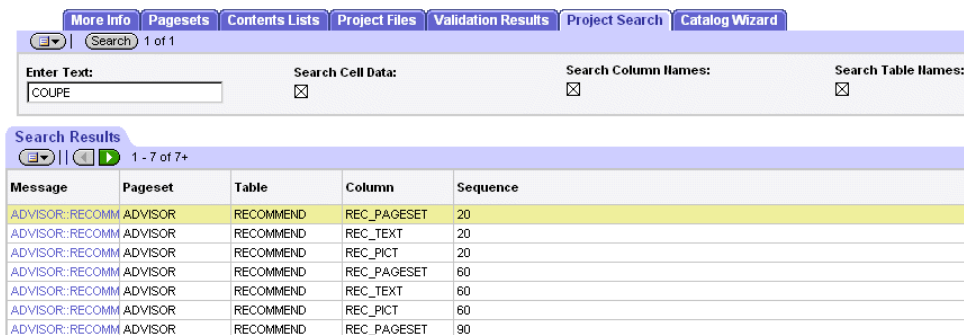
Figure 7. The Validation Results View

Search View

In a large project, use the Search feature to locate the text you are looking for. In the Search view, shown in [Figure 8](#), you can search the project data for a particular string of text. The Search tab displays a list of matches for your search entry. Clicking on a particular match takes you to the project area containing that text.

Search automatically performs a floating search. A floating search is a search that assumes wild cards around the search text. For example, in a search for “ar,” all text containing “ar” is returned.

For more information, see [“Performing a Project Data Search”](#) on page 53.



The screenshot shows the Search View interface. At the top, there are tabs for 'More Info', 'Pagesets', 'Contents Lists', 'Project Files', 'Validation Results', 'Project Search', and 'Catalog Wizard'. The 'Project Search' tab is active. Below the tabs, there is a search bar with the text 'COUPE' entered. To the right of the search bar, there are three checkboxes: 'Search Cell Data', 'Search Column Names', and 'Search Table Names', all of which are checked. Below the search bar, there is a 'Search Results' section with a table of results. The table has five columns: 'Message', 'Pageset', 'Table', 'Column', and 'Sequence'. The first row is highlighted in yellow.

Message	Pageset	Table	Column	Sequence
ADVISOR:RECOMM	ADVISOR	RECOMMEND	REC_PAGESET	20
ADVISOR:RECOMM	ADVISOR	RECOMMEND	REC_TEXT	20
ADVISOR:RECOMM	ADVISOR	RECOMMEND	REC_PICT	20
ADVISOR:RECOMM	ADVISOR	RECOMMEND	REC_PAGESET	60
ADVISOR:RECOMM	ADVISOR	RECOMMEND	REC_TEXT	60
ADVISOR:RECOMM	ADVISOR	RECOMMEND	REC_PICT	60
ADVISOR:RECOMM	ADVISOR	RECOMMEND	REC_PAGESET	90

Figure 8. The Search View

Pageset Administration Screen

Use the Pageset Administration screen to create, view, modify, and delete pagesets. A pageset contains all the feature and configuration data for a product, along with information on how that data should appear inside the completed application.

From the Pageset Administration view, you can create and access:

- Feature and Configuration tables
- Input and Output UI
- Files that make up the pageset
- Validation and search functionality

For more information, see [Chapter 6, “Working with Pagesets.”](#)

Feature Table Editor View

The Feature Tables tab provides two views which can be selected from the Show drop-down list:

- **Editor.** Edits the Feature table data.
- **Designer.** Designs the Feature table.

Use the Feature Table Editor view ([Figure 9](#)) to enter information about the features in your pageset. Each Feature table represents a single feature, and contains all possible values for that feature. Data you enter in Feature tables populates the UI controls in your input UI.

For more information, see [“Creating Feature Tables”](#) on page 103.

Show drop-down list

Name	Table Type	Linked To Table	Notes	Last Updated	Last Updated By
BUDGET	Standard			07/24/2001 6:34:54	SADMIN
SEATS	Standard			07/24/2001 6:34:52	SADMIN
STORAGE	Standard			07/24/2001 6:34:53	SADMIN

Row Type	Sequence	CODE	DESC	DEFAULT
DATA	10	NO	No frills	
DATA	20	MOD	Moderate	
DATA	30	FULL	Fully Loaded	
DATA	40	B	BLANK SO FAR	default

Figure 9. Feature Table Editor View

Feature Table Designer View

The Feature Tables tab provides two views which can be selected from the Show drop-down list:

- **Editor.** Edits the Feature table data.
- **Designer.** Designs the Feature table.

Use the Feature Table Designer (shown in [Figure 10](#)) to add columns to your Feature tables. You can associate these columns with Siebel business component fields or Siebel classification system attributes.

For more information, see [“Creating Feature Tables” on page 103](#).

Show drop-down list

The screenshot displays the 'Feature Tables' configuration screen. At the top, there are several tabs: 'More Info', 'Feature Tables', 'Configuration Tables', 'Input UI', 'Output UI', 'Pageset Files', 'Validation Results', and 'Pageset Search'. The 'Feature Tables' tab is active. Below the tabs, there is a 'Show:' dropdown menu currently set to 'Designer'. To the right of the dropdown are buttons for 'Query' and 'Edit', and a page indicator '1 - 3 of 3'. Below this is a table with the following data:

Name	Table Type	Linked To Table	Notes	Last Updated	Last Updated By
BUDGET	Standard			07/24/2001 6:34:54	SADMIN
SEATS	Standard			07/24/2001 6:34:52	SADMIN
STORAGE	Standard			07/24/2001 6:34:53	SADMIN

Below the table, there is a 'Designer' tab. The 'Show:' dropdown is also set to 'Designer'. To the right are buttons for 'New' and 'Save', and a page indicator '1 - 3 of 3'. Below this is a table for designing columns with the following data:

Sequence	Column Name	Business Compo	Field Name	Shared	Class	Attribute	Target Table
1	CODE						
2	DESC						
3	DEFAULT						

Figure 10. The Feature Table Designer View

Configuration Table Editor View

The Configuration Tables tab provides two views which can be selected from the Show drop-down list:

- **Editor.** Edits the selected Configuration table data.
- **Designer.** Designs the selected Configuration table.

Use the Configuration Tables Editor view (Figure 11) to enter all combinations of Feature table data, and to determine which combinations are valid. In the Configuration tables, you will also write exception messages that appear when a user selects an invalid combination of data.

For more information, see “Creating Configuration Tables” on page 75.

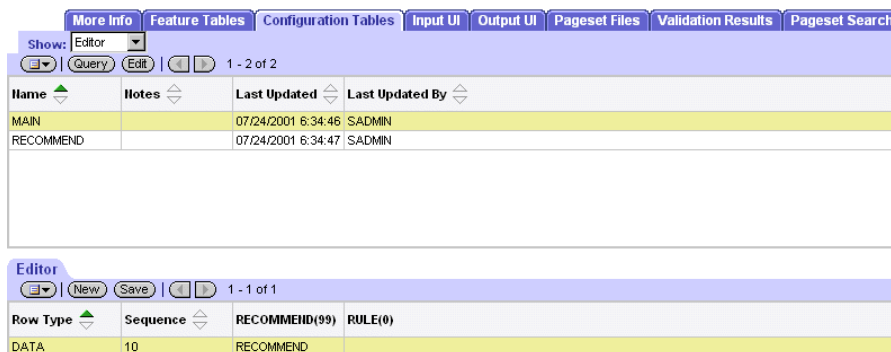


Figure 11. The Configuration Table Editor View

Configuration Table Designer View

The Configuration Tables tab provides two views which can be selected from the Show drop-down list:

- **Editor.** Edits the selected Configuration table data.
- **Designer.** Designs the selected Configuration table.

Use the Configuration Table Designer view (shown in [Figure 12](#)) to add columns to your Configuration tables. You can associate these columns with Siebel business component fields or Siebel classification system attributes.

For more information, see [“Creating Configuration Tables” on page 75](#).

More Info Feature Tables Configuration Tables Input UI Output UI Pageset Files Validation Results Pageset Search							
Show: Designer							
Query Edit 1 - 2 of 2							
Name	Notes	Last Updated	Last Updated By				
MAIN		07/24/2001 6:34:46	SADMIN				
RECOMMEND		07/24/2001 6:34:47	SADMIN				

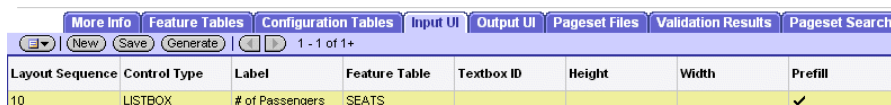
Designer								
New Save 1 - 2 of 2								
Sequence	Column Name	Column Type	Business Compo	Field Name	Shared	Class	Attribute	Notes
1	RECOMMEND	99						
2	RULE	0						

Figure 12. The Configuration Table Designer View

Input UI View

Use the Input UI view, shown in [Figure 13](#), to determine the types of UI controls that display your Feature table values. Your Input UI control options are Check Box, Get Text, List Box, and Radio. Use the Generate button to add your changes to the Input UI in the HTML file (pageset_1.htm) associated with the Input UI frame.

For more information, see [“Creating Input UI Controls” on page 141](#).



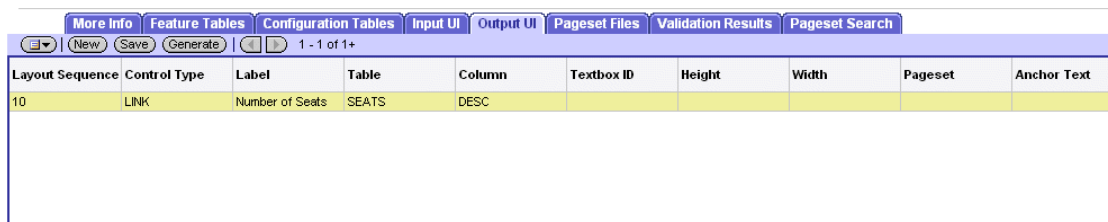
Layout Sequence	Control Type	Label	Feature Table	Textbox ID	Height	Width	Prefill
10	LISTBOX	# of Passengers	SEATS				<input checked="" type="checkbox"/>

Figure 13. The Input UI View

Output UI View

Use the Output UI view ([Figure 14](#)) to determine the types of output UI controls that display in the Results page. Your output UI control options are Detail, Link, Pict, and Text. These UI controls are populated by values in the Configuration and Feature table columns. Use the Generate button to add your changes to the Output UI in the HTML file (pageset_2.htm) associated with the Output UI frame.

For more information, see [“Creating Output UI Controls” on page 148](#).



Layout Sequence	Control Type	Label	Table	Column	Textbox ID	Height	Width	Pageset	Anchor Text
10	LINK	Number of Seats	SEATS	DESC					

Figure 14. The Output UI View

Pageset Files View

Use the Pageset Files view (Figure 15) to open the files that belong to a pageset.

When you create a new pageset, the following four files are automatically added to the Siebel File System (Directory|File):

- ds|pageset_x.js
- pg|pageset_1.htm
- pg|pageset_2.htm
- pg|pageset_i.htm

You can also access these files from the Project Files tab of the project containing the pageset you want to work with.

You can associate additional files with the pageset by clicking New. A Files picklist will open, from which you can select and add a file, such as image files and product feature documents, to the Pageset files tab.

For information about these directories and files, refer to the Browser-Based Application File Reference chapter in *Siebel Interactive Designer API Reference*.

Attachment Name	Type	Directory	Modified Date/Tim	Notes
pricing	htm	cs	07/24/2001 6:37:47	
home	htm	cs	07/24/2001 6:36:43	

Figure 15. The Pageset Files View

Working with Projects

4

This chapter describes how to work with projects in Interactive Designer, from creating and editing a project to validating and deploying it. It also describes best practices for working on team projects.

About Projects

A project is the top-level container for all the information you work with in Interactive Designer that makes up the application. Use the Interactive Designer project to create pagesets and determine the structure and content of the contents list.

If the project data changes, update the appropriate tables in Interactive Designer and redeploy the project to update the associated application. You can edit the contents list on the Contents List tab. In Pagesets view, the Configuration Tables tab, the Feature Tables tab, and the Pageset Files tab display a list of the tables and files that define the information for all pagesets.

After Interactive Designer generates application files from the project data, you can use a text or HTML editor to add application functions that further customize the appearance and behavior of the application.

Viewing Your Projects

Interactive Designer provides two views of projects:

- My Projects

Displays a list of projects for everyone in your group. For example, if your eBusiness position (which is associated with your login) is set to Sales Manager, select My Projects to see the projects for all sales managers. You can edit only the projects that are created by your team or your team has access to.

- All Projects

Provides a view of all projects for all positions. While you will be able to view all of the projects, you can edit only the projects that belong to your team.

NOTE: Positions are set and modified in the Application Administration tab in Siebel eBusiness Applications. For more information, refer to *Siebel Applications Administration Guide*.

To view your projects

- Select My Projects from the Show drop-down list.

To view a particular project

- 1** In the Projects screen, select a project row.
- 2** In the More Info tab, you can view the following details for the project:
 - Name for the project
 - Project directory in which the project lives
 - Notes about the project
 - Associated price list, if any
 - Project team
 - Date on which the project was last updated
 - User to last update the project

Select the other project tabs to view the associated pagesets, contents list, files, and validation results. Use the search tab to perform a text search. See the related topics in this book for more information on the Project tabs.

Creating a Project

Use the following procedure to create a project.

To create a project

- 1** From the Show drop-down list, select My Projects.
- 2** From the Projects View applet menu, choose New Record.

The New Project applet appears.

- 3** Enter the name of the project.

This is the name that appears in the Projects list.

- 4** Enter the directory name, no spaces, in which to save the project files.
- 5** Select a UI Template from the UI Template field.

- 6** Click Project Team to open the Project Team picklist and select the teams who can access the project.

The project appears to all members of the selected teams when they select My Projects. This field is automatically selected for you, but you can modify it as described in this step.

- 7** Click Price List to open the Price List picklist and select the price list you want to use for the project.

From the Price List picklist, you can select from all available price lists. If your project references products from the Siebel product master list, you may use an price list. See *Pricing Administration Guide* for information on setting up pricing lists.

- 8** Enter the release number for the project and any notes. This step is optional.
- 9** Click Save.

Migrating a Project

Select Migrate from the applet menu to update 4.0 projects to 7.0. If you are working with a 3.x project, you must migrate it to 4.0 before migrating to 7.0.

For more information, refer to *Siebel Interactive Selling Applications Upgrade Guide*.

Copying a Project

To use the same data in an existing project, you can copy a project. All dependent pagesets, contents lists, Feature tables, Configuration tables, and data will be copied along with the project.

To copy a project

1 Select a project.

2 From the Projects View applet menu, choose Copy Record.

A New Project screen appears, displaying the data for the project you are copying.

3 Enter a new project name and directory.

These values must be unique throughout the application.

4 Click Save.

The new record appears in the Project list.

5 Deploy the new project.

The new project contents are an exact copy of the project you copied.

Deleting a Project

Use the following procedure to delete a project.

To delete a project

- 1** In the Projects screen, select the project you want to delete.
- 2** From the Projects View applet menu, choose Delete Record.

A dialog appears confirming that you want to delete the record.

- 3** Click OK.

The project is deleted from the Projects list and all pagesets associated with the project are deleted. Any files that have been deployed will remain in their deployed directories and must be deleted manually.

NOTE: Until you move to a new project, you can undo the delete operation.

Editing a Project

Use the following procedure to edit a project.

To edit a project

- 1** In the Projects screen, select the project you want to edit.
- 2** In the More Info tab, edit the information and click Save.

Performing a Project Data Search

Perform the Interactive Designer data searches on the selected project using either the:

- Project Search tab
- Query button

Project Search queries all the data in the project. Project Search automatically performs a floating search. A floating search is a search that assumes wild cards around the search text. For example, in a search for “ar,” all text containing “ar” is returned.

Query searches the data shown in the Project list.

To perform a search using the Search tab

- 1** Select the Project Search tab.
- 2** Enter the text you want to find.
- 3** Specify what to search: Cell Data, Column Names, or Table Names.
Select all three to search all three options.
- 4** From the View applet menu, choose Search.
All matches for your search appear in the Search Results applet.
- 5** Click a search result to open the view, pageset, and table that contains the search result.

To perform a search using the Query button

- 1** On the tab in which you want to search for data, select New Query.
A Query screen appears that lists all the fields for that tab.
- 2** Enter data in the field you want to search by.
- 3** Click Go.

The Query returns the results for your search. All other records will not appear until you perform another search.

Validating a Project

Validate your data model regularly as you work and before you deploy a project. If you have validation errors, you can not deploy the application.

To validate a project

- 1** In the Projects screen, select the project you want to validate.
- 2** From the Projects View applet menu, choose Validate.
- 3** A list of validation errors appear in the Validation Results tab. The error identifies the pageset, table, column, sequence of the error, and message describing the error. Errors are also generated for contents lists. Click an error to view it.

When you click an error, Interactive Designer opens the correct screen and tab and selects the record containing the error.

- 4** Fix the errors and return to the Validation Results tab to select Validate again.

When all errors are fixed, the Validation Results tab contains no error records.

Previewing a Project

To make sure that the edits you make to the Interactive Designer project affect your application, preview the application in a browser.

NOTE: To preview a stand-alone application, open the home.htm file in a browser.

To preview a project

- 1 From the Tools menu of your browser, choose Internet Options and click Settings.
- 2 Select Every Visit to the Page. This is a one-time setting.
- 3 In the Projects screen, select the project you want to preview.
- 4 Validate the project.

For more information, see [“Validating a Project” on page 54](#).

NOTE: Complete the following two steps if you are working in a zero footprint Web client development environment.

- 5 In your Siebel application .cfg file, set the WebClientSiteDir variable to the directory you want your files published to. This directory must specify the Siebel Web Engine, the public folder, and the language folder, as in SWEAPP\public\ < lang > .

The files will be saved to this folder under the ISSRUN\CDAPROJECTS directory. For example, if the WebClientSiteDir variable is set to eapps\public\enu, then for the project “QuickTour” whose project location is set to “QuickTour,” the preview files are saved to eapps\public\enu\issrun\cdaprojects\QuickTour.

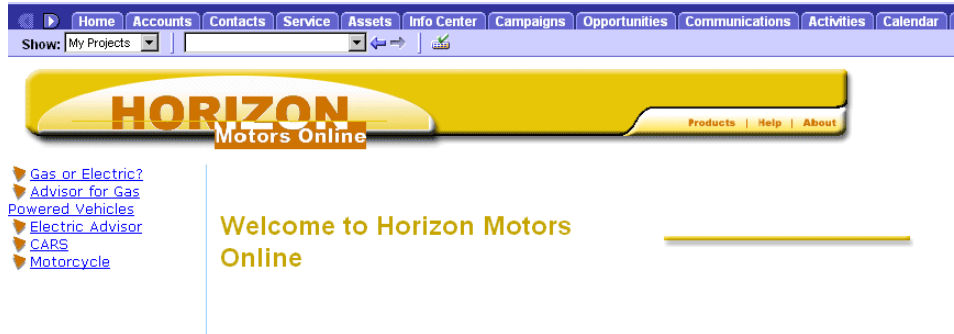
- 6 If you are deploying in multiple languages, repeat [Step 3](#) for each .cfg file in each language folder.

The .cfg files live under the Siebel Root\bin\ < lang > directory. Open each language directory, for example, ENU for English, and set the WebClientSiteDir variable in the .cfg file to point to the appropriate Siebel Web Engine for that language’s deployment.

- 7 From the Projects View applet menu, choose Preview.

The application appears as it will at runtime.

The following figure shows an example of a previewed application.



Deploying a Project

After previewing and validating your application, use the Deploy button to deploy the files to the appropriate environment.

To deploy your application

- 1 Open the project you want to deploy.
- 2 Unlock all pagesets for the project.

NOTE: Complete the following two steps if you did not already complete them to preview your application.

- 3 In your Siebel application .cfg file, set the WebClientSiteDir variable to the directory you want your files published to. This directory must specify the Siebel Web Engine, the public folder, and the language folder, as in SWEAPP\public\ < lang > .

The files will be saved to this folder under the ISSRUN\CDAPROJECTS directory. For example, if the WebClientSiteDir variable is set to eapps\public\enu, then for the project “QuickTour” whose project location is set to “QuickTour,” its runtime files are deployed to eapps\public\enu\issrun\cdaprojects\QuickTour.

- 4 If you are deploying in multiple languages, repeat [Step 3](#) for each .cfg file in each language folder.

The .cfg files live under the Siebel Root\bin\ < lang > directory. Open each language directory, for example, ENU for English, and set the WebClientSiteDir variable in the .cfg file to point to the appropriate Siebel Web Engine for that language’s deployment.

- 5 From the Project View applet menu, choose Deploy.

The project version number is increased by an increment of one and all files in the Project Files view are saved to the project directory (the directory you specified when you created the project) under the directory you specified in the WebClientSiteDir variable in your Siebel application .cfg file.

Exporting and Importing a Project

Use Export Project and Import Project to move project data from one database server to another. When you use Export Project, an .xml file is created. This .xml file contains all the data and graphics for the project.

Before exporting and importing a project, make sure that Siebel product or business component data accessed from the application is available in both databases. For example, if you added additional business components to the project you are importing, you will need to define the LOVs that display those business components before importing the project. References to data are maintained when exporting and importing, but you must make sure the data referred to exists before importing.

To export a project

- 1** Open Interactive Designer on the server containing the project you want to move.
- 2** In the Projects view, select a project.
- 3** From the Projects view applet menu, choose Export Project.

The Export Project dialog appears.

- 4** Click Export Project.

The File Download dialog box appears. Depending on the size of the project, it could take several minutes before this dialog appears.

- 5** Select Save File to Disk and click OK.
- 6** Navigate to the directory in which you want to save the project XML file.

You can save this file on your hard drive or on the network.

- 7** Click OK.

An .xml file is created and saved to the specified directory.

- 8** Click Save.

NOTE: If you are importing a project that you have previously imported to the same location, rename the project before importing. You cannot overwrite a project when you import.

To import a project

- 1** Open Interactive Designer on the server you will move the project to.
- 2** Open the Projects view.
- 3** From the Projects view applet menu, select Import.

The Import Project dialog appears.

- 4** Click Browse.
- 5** In the Choose File dialog, navigate to the project XML file and click Open.
- 6** Click Import Project.

The project is added to the Project list. All project data, including pagesets, tables, files, and contents lists, are available in the project. All pagesets are unlocked.

The Last Updated field displays the date and time the project was imported. The Last Updated by field displays the user ID for the user who imported the project.

Projects in a Team Environment

The Siebel environment allows you to work on your projects in a team environment. It provides a central repository for your project data, as well as distributed Web access to your projects. It also provides other Siebel views in which different individuals or groups can maintain price lists and product information.

Additionally, Interactive Designer provides the following features to support working on projects in a team environment:

- Controlling view access
- Controlling team access
- Locking pagesets
- Recording Version Number

View Access

Use the Responsibilities view in Siebel Application Administration to set the visibility to the Interactive Designer views, such as the Table Editor views. Visibility is based on the logged on user's ID (by associating the logon ID with a particular responsibility). This allows you to control what a user can do through the Interactive Designer user interface.

For more information on setting up view access, refer to *Siebel Applications Administration Guide*.

Team Access

Use Interactive Designer to set up the Pageset Team and Project Team groups so that, based on the logged on user's primary position, you can control what projects and pagesets appear in My Projects and My Pagesets.

To set up team access

- When you create a project or pageset, the primary position becomes the owner of the project or pageset and any person with this primary position can view and edit the project or pageset. You can add other positions to the team by clicking the Project Team or Pageset Team button on the More Info tab which opens a picklist. The project or pageset will appear to all members of the selected teams when they select My Projects.

Locking Pagesets

To avoid problems with concurrent updates to projects, you must lock a pageset before editing it. Only the user who locked the pageset has permission to make edits.

To lock a pageset

- 1 Navigate to My Pagesets.
- 2 In the Pagesets tab, select a pageset.
- 3 In the More Info tab, check the Locked check box.
- 4 Click Save.

Recording the Release Number

Each time you deploy a project, an incremental release number is created. Record this number to track your versions of a project.

To view a project release number

- Navigate to My Projects.

In the Projects tab, the Release Number column displays the version number for a project.

Working with Configuration Tables

5

This chapter describes how to work with Configuration tables. It describes basic functions, such as creating, designing, and editing Configuration tables. It also describes advanced use of Configuration tables, such as creating exception messages and cross-sells, as well as creating links back to a pageset from a cross-sell. Additionally, this chapter describes how the configuration matching process works.

About Configuration Tables

Use Configuration tables to define valid and invalid combinations of all the features you entered for the pageset.

Each time you make a selection in the Input UI display page, the application engine checks to see if the feature values are valid or invalid. A portion of the data in a pageset defines all of the possible combinations of feature values. Each combination represents a valid configuration or an exception.

When the selected features are compatible and represent a product that is available for sale, the selection set represents a valid configuration. When a combination of features does not make a valid configuration, it generates an exception. When this situation occurs, the application displays an exception message that guides users to make choices that match a valid configuration.

Each pageset must have at least one Configuration table called MAIN, which Interactive Designer creates for each new pageset. If you have large amounts of complex configuration information, you can create additional Configuration subtables and point to them from the MAIN Configuration table.

The Configuration Matching Process

When users make a feature selection in one of the input UI controls in an application, the engine searches through the configuration data of the pageset to determine whether the feature selection represents, in combination with all the other feature selections, a valid product configuration.

The engine searches through product configuration information in the following order:

- 1 The engine runs through the Configuration Data area of the MAIN Configuration table, comparing user selections in the application to the feature codes defined in input columns of configuration rows.

In all Configuration tables, the matching process evaluates DATA rows before evaluating EXCEPTION rows. The current input UI control selections are compared against valid configurations before they are compared against invalid configurations.

Additionally, because Configuration table cells are read from left to right, when you create configuration rows you must place cells in the order in which you want them evaluated. The RULE column, for instance, should always be the last column on the right, so that all feature codes are evaluated before the exception message appears.

The MAIN Configuration table ultimately defines all valid feature configurations. If the engine finds a matching configuration row in the DATA Row Types of the MAIN table, the configuration is valid. If not, the configuration is invalid.

While searching Configuration Data in the MAIN Configuration table, the engine may refer to subtables that contain more configuration definitions for the pageset. In order for a configuration row in the MAIN Configuration table to be a valid match, all the subtables it points to must also contain matches.

- 2** If the engine finds a match from among the input columns in the MAIN Configuration table (a row in which all the input columns contain feature codes that match the feature selections in the application) it checks the rest of the configuration row for any subtable columns.

If the engine does not find a subtable column, the configuration is considered valid and the configuration matching process ends. If the engine does find a subtable column in the row, it pauses and evaluates the configuration rows defined in the subtable that the subtable column references.

As with feature codes, when you create configuration rows you must place subtable columns in the order, from left to right, in which you want the Configuration subtables they represent to be evaluated.

If the Configuration subtable contains subtable columns that point to further subtables (sub-subtables), the engine pauses and evaluates the configuration rows defined in the sub-subtables. The engine then returns to the original subtable and continues evaluating from the point at which it paused.

If the engine does not find a valid configuration that matches the current input UI control selections, then it evaluates exception data in the subtable. The engine then returns to the Configuration table from which it came and evaluates the data in that table.

- 3** If the engine does not find a configuration that matches the current input UI control selections in the valid configurations defined in the MAIN Configuration table, or in any of the valid or invalid configurations defined in Configuration subtables, it compares the user-selected configuration to the invalid configurations (exception data).

When the engine reaches a matching row in the EXCEPTION Row Types, it displays the text in the associated output RULE column as an exception message in the application.

- 4 The engine checks to see if the pageset includes a Configuration table named OL_CONDITIONS. If not, the configuration matching process is complete. Otherwise, the engine continues on to evaluate the OL_CONDITIONS Configuration table.

An OL_CONDITIONS Configuration table is not required in any pageset, and each pageset can contain only one OL_CONDITIONS table.

The application engine recognizes the name OL_CONDITIONS and begins the configuration matching process in this Configuration table, if one is found. An OL_CONDITIONS Configuration table exists only to evaluate the current input UI control selections in the application against JavaScript-defined conditions.

The OL_CONDITIONS table must contain only two columns: an input column named TEST, and a output RULE column. The cells in each row of the TEST column contain a JavaScript expression that returns true or false. The RULE column for each row contains the exception message that appears if the TEST expression returns true.

Each row in the OL_CONDITIONS table is evaluated in order of SEQUENCE number, from smallest to greatest, against the current user-selected configuration.

If all of the TEST expressions in the OL_CONDITIONS table return false, the engine considers the user-selected configuration valid and ends the configuration matching process. If a TEST expression returns true, the exception message defined in the associated RULE column appears in the application.

For more information on the OL_CONDITIONS Configuration table, see [“Creating Javascript Conditional Statements” on page 134](#).

Configuration Column Types

Like Feature tables, Configuration tables ask for a name and an optional description of the table columns you are creating. Unlike Feature tables, Configuration tables require that you specify a column type: output, input or subtable. The type of column in a Configuration table represents the purpose that the data entered into it serves in defining a product configuration. After you define all columns, choose the Editor tab. In the Editor view, you can enter row and cell data for the table.

Input Columns

Each input column, sometimes called a key column, in a Configuration table corresponds to a Feature table. The cells in input columns must contain the code values of items in the corresponding Feature table. Each row in a Configuration table represents a configuration, either valid or invalid, of the Feature code values listed in its input column cells.

When you use a Feature table name as an input column name in a Configuration table, in the Table Editor view you will fill in the code values of the rows in the corresponding Feature table.

For example, if one Feature table in a pageset defines the exterior colors of a car, and another Feature table defines the interior colors, in the Configuration Table Editor view you would enter the exterior and interior color codes from the Feature tables in the input columns for EXT_COLOR and INT_COLOR. You would specify which exterior colors are available with each interior color by matching each exterior color value with each interior color value, row by row. The following example determines which models are available with which exterior and interior colors. This example uses range functions to simplify the data entry process. See [“Range Functions” on page 74](#) for more information.

In [Figure 16](#), the highlighted row shows that the Model GXE is available with a red exterior and interior.

Row Type	Sequence	MODEL(1)	COLOR_EXT(1)	COLOR_INT(1)	RULE(0)
DATA	10	XE	(=BE,VH,BK,GR)	(=BE,VH,BK,GR)	
DATA	20	LE	(=BE,VH,BK,GR,BL)	(=BE,VH,BK,GR)	
DATA	30	LE	BL	BL	
DATA	40	GLE	(=BE,VH,BK,GR,BL)	(=BE,VH,BK,GR,BL)	
DATA	50	GLE	BL	BL	
DATA	60	GXE	(=BE,VH,BK,GR,BL)	(=BE,VH,BK,GR,BL)	
DATA	70	GXE	RE	RE	
DATA	80	GXE	BL	BL	
EXCEPTION	10	XE	(=BL,RE)	(=*)	Sorry, the Basic Model is not ...
EXCEPTION	20	XE	(=*)	(=BL,RE)	Sorry, the Basic Model is not ...
EXCEPTION	30	LE	RE	(=*)	Sorry, the Limited Edition is ...
EXCEPTION	40	LE	(=*)	RE	Sorry, the Basic Model is not ...
EXCEPTION	50	LE	(=BL)	BL	Sorry, the ...
EXCEPTION	60	GLE	RE	(=*)	Sorry, the Special Edition is ...
EXCEPTION	70	GLE	(=*)	RE	Sorry, the Special Edition is ...
EXCEPTION	80	GLE	(=BL)	BL	Sorry, the ...
EXCEPTION	90	GXE	(=BL)	BL	Sorry, the ...
EXCEPTION	100	GXE	(=RE)	RE	Sorry, the r...
EXCEPTION	110				TEST for MODEL_COLOR

Figure 16. Configuration Editor Displaying Input Columns

The valid configurations (the combinations of Feature code values that represent available products) appear in the Configuration table list as DATA Row Types. The invalid configurations, the unavailable feature combinations, appear in the Configuration table list as EXCEPTION Row Types.

Output Columns

Output columns contain data that is not evaluated during the configuration matching process. Output column data contains information about a configuration without being part of the configuration definition itself. Use an output column to define items and events that exist as a result of a user selecting a configuration in the application. An item or event can be a variable that is defined, text that is displayed, or an image. Output columns hold the recommended solution for the user-selected choices.

Every Configuration table must contain at least one output column, called RULE. Interactive Designer automatically adds this column whenever you create a new Configuration table. The RULE column defines the text, called an exception message, that appears in the application to guide users when they select an invalid configuration.

In the Configuration Table Editor, cells in the RULE column are usually empty for the DATA Row Types and filled in for the EXCEPTION Row Types. This is because the Exception Row Types define invalid product configurations. Cells in other output columns, such as a column that defines a variable, may contain data only in the DATA Row Sets, because their existence is dependent on a valid configuration instead.

Subtable Columns

Subtable columns point from a Configuration table to a Configuration subtable. The difference between a Configuration table and a Configuration subtable is the order in which the configurations defined in the table are evaluated.

If the engine reaches a subtable column during the configuration matching process, it pauses and evaluates the configurations defined in the subtable. If the engine does not find a match in the subtable, it returns to the original Configuration table and continues to evaluate defined configurations from the point at which it exited. For more information, see [“The Configuration Matching Process” on page 65](#).

Cell Functions

When you are working in Configuration tables, you can use javascript expressions in the table cells to refer to other table columns, perform calculations, and simplify the amount of data you enter into a cell. When a cell with a function is referenced, the calculation is performed and the application displays the result of the calculation.

When using cell functions, use the following rules:

- Create an output column for cell functions
- Use only javascript expressions
- Enclose all cell functions in parentheses

For example, (CAR.PRICE-DISCOUNT)

About Referring to Other Table Columns

Refer to columns in Feature tables by using the expression TABLE.COLUMN. For example, to refer to the column PRICE in the COLOR Feature table, use the expression (COLOR.PRICE).

Refer to Configuration tables by using the expression COLUMN. For example, to refer to the column SPEED in the MODEM Configuration table, use the expression (SPEED).

In cell functions, you can reference an unlimited number of columns.

About Performing Calculations

Use standard javascript syntax to perform calculations in a table cell.

For example, you can calculate the total quantity of car alarms and stereo systems by using the following calculation: ALARM.QTY + STEREO.QTY. This example adds the value in QTY column of the ALARM table to the value in the QTY column of the STEREO table and displays the total in the application that referenced it.

Nested Cell Functions

You can nest cell functions, but be aware of how the nested expression is evaluated. Cell functions are evaluated from left to right and bottom to top, as in [Figure 17](#).

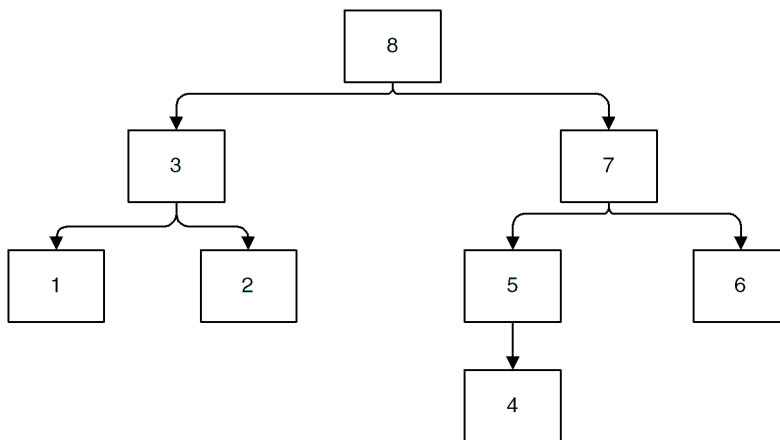


Figure 17. Order of Evaluation for Nested Cell Functions

The numbers in this chart represents the order in which functions are evaluated. A larger numbered table can reference values in a smaller numbered table.

For example, table 1 has a column RESTRAINT_PRICE with cell function (LEASH.PRICE + HARNESS.PRICE).

In this case, table 8 (MAIN) could have a column TOTAL_PRICE with (RESTRAINT_PRICE + PET.PRICE).

You cannot reverse the cell functions and have the second cell function in table 1, and the first in table 8 due to the order of evaluation.

Because tables are evaluated from left to right, you also need to be careful about which cells you reference. When referring from a Configuration table to a Feature table, you can only refer to cell functions that are in your table to the left, or to a literal anywhere in the Feature table. In Configuration tables, you can reference literals anywhere in any Configuration or Feature table.

Example

In this example, you will add a cell function that calculates the customer's spending capacity based on two selections: salary, and cash available for a down payment.

To add the cell function

- 1** Open the MAIN Configuration table in Design view.
- 2** Create a New row.
- 3** Name the column SPEND_MAX and enter Output for the type of column.
- 4** Save the row.
- 5** In Edit view, enter the following formula into all valid configuration rows of the new SPEND_MAX(Output) column: $((\text{SALARY.AMT} * .75 * .05 * 5) + \text{CASH.AMT})$.
- 6** Click Save.

Range Functions

Use range functions in input columns to simplify data entry and create smaller models for faster application load time. For example, instead of creating three separate rows in a Configuration table to create a rule that displays an exception message when a user selects the colors red, blue, and green, you can create one row and enter the range function (= Red,Blue,Green).

NOTE: For rules involving more than one code value, the syntax (= bm*) cannot be used; doing so will cause an error when generating HTML files in eAdvisor. Instead, use the syntax (= bm1,bm2,bm3) or (= *).

Table 1 shows examples of range functions:

Table 1. Examples of Range Functions

Range Function	Description
(= Red,Blue)	Anything equal to Red or Blue
(= 1-3)	Anything that numerically matches, numbers only
(= *)	All
(! = Red,Blue)	Anything not equal to Red and not equal to Blue

Viewing Configuration Tables

Use the following procedure to view a Configuration table.

To view a Configuration table

- 1 Select the pageset containing the Configuration table you want to view.
- 2 Click the Configuration Tables tab to view the list of Configuration tables.
- 3 Select a Configuration table from the list to view its details in the Editor applet.

Creating Configuration Tables

Creating a Configuration table requires three steps:

- Opening the Main Configuration table or create a new configuration subtable.
- Designing the Configuration table.
- Entering data in the Configuration table.

These steps are explained in detail in the following sections.

Creating Configuration Subtables

The Main Configuration table is included in your project by default. Use the following procedure to create Configuration tables in addition to the Main Configuration table. You can use a Configuration subtable for a clear subset of configuration data in your project. Use Configuration subtables to normalize the data and make the data model small, or for organizational purposes.

To create a Configuration subtable

- 1 From the Pagesets screen, select the Configuration Tables tab.
- 2 From the applet menu, choose New Record.

The new Configuration Table applet appears.

- 3 Enter a name for the Configuration table, and any notes.

- 4 Click Save.
- 5 Select a Configuration table and choose the Designer tab.
- 6 In the Designer, click New.
- 7 In the Sequence text field, enter a sequence number.
- 8 In the Column Name text field, enter the name of the Configuration subtable.
- 9 From the Column Type drop-down list, select Subtable, and click Save.
- 10 Enter the table data with the name of the subtable.

For example:

Table 1	Price	Subtable
0	0	SUBTABLE 1
1	100	SUBTABLE 1

Designing the Configuration Table

When you design a Configuration table, you create columns that reference each Feature table. Use these columns to choose configurations for your features in edit view.

To design the Configuration table

- 1 Open the Configuration table you want to design and select the Designer tab.
- 2 In the Designer applet, click New.

A new row appears.

- 3 Enter the sequence number for the configurations. This number determines the order in which the columns appear in the Configuration editor.

4 From the picklist, select a name for the column.

For example, if you want to include the COLOR Feature table values in your Configuration table, select the column name COLOR.

For output columns:

- If you are mapping to a business component rather than a Feature table, select the business component field from the picklist.
- If you are mapping to a class rather than a Feature table, select the class attribute from the picklist.

5 Select the column type.

For basic modeling, enter:

- Input to point to a Feature table. The data for this column comes from Feature tables that populate input UI controls.
- Output to enter a rule to display an exception message or output information, such as a price or eAdvisor text, or perform another action when a particular configuration is selected. The data for this column can come from Feature tables, new data the user enters, or business components.
- Subtable to point to a Configuration subtable.

For more information on column types, see [“Configuration Column Types” on page 68](#).

6 Select a business component. This step is optional.

NOTE: If you select a business component and field, you cannot select a class and attribute. These options are mutually exclusive. You cannot use business components for subtable columns or the RULE column.

This column maps to the selected column (business component) in the Siebel database. For more information on business components, refer to *Siebel Tools Reference*.

- 7 If you selected a business component in [Step 6](#), click Field Name to open a picklist and select a field name from the business component. This step is optional.
- 8 Select a class. This step is optional.

NOTE: If you select a class and attribute, you cannot select a business component and field. These options are mutually exclusive.

This column maps to the selected Siebel class. For more information about classes and attributes, refer to *Siebel Tools Reference*.

- 9 If you selected a class in [Step 8](#), select an attribute.
- 10 Check the Shared field to reference one row ID in a business component. This step is optional.

All columns in the Table Designer that reference the same business component and have the Shared field selected are populated with data when a value is selected for any one of them in the Table Editor.

For example, if you add the following four columns to your Configuration table and they all reference the Contact business component:

- Name
- Phone
- Organization
- Address

Then when you switch to Edit View and select a value for the Organization column, values for Name, Phone, and Address are automatically filled in.

- 11 Enter any notes and click Save.

Entering Data in the Configuration Table

After you have designed your Configuration table, enter all possible combinations of data and determine which combinations are valid, which are invalid, and what messages or recommendations to display when each combination is selected.

To enter data in the Configuration table

1 In the Configuration Tables tab, select the Configuration table you want to enter data in and select the Editor tab.

2 In the Editor applet, click New.

A new row appears.

3 Click Row Type to open a picklist and choose one of the following:

- Data for valid configuration rows.
- Exception for invalid configuration rows.

4 If you selected Exception in [Step 3](#), enter an exception message in the Rule column.

For example, “The Luxury sedan is not available with a red exterior. Please choose black or silver.” This is a required step.

If you selected Data in [Step 3](#), create a guidance message for cross and up-sell messages or eAdvisor questions. For more information, see [“Creating Cross-Sell and Up-Sell Messages” on page 84](#) or [“Creating an eAdvisor Application” on page 28](#).

5 Enter the sequence number. This is an optional step.

This determines the order in which the new row will appear in the Configuration table editor. This number is automatically generated, but you can override it by editing the number.

NOTE: A data row and an exception row may have the same sequence number because data rows are sequenced separately from exception rows. Data rows and exception rows must have unique sequence numbers within their types, but a data row may have the same sequence number as an exception row.

- 6** For Configuration table rows associated with:
 - A business component, click the select button to open a picklist and select a business component field. Click OK to accept your selection and return to the Configuration table editor.
 - An input column, enter a Feature table code value in the text field.
 - A class, enter an attribute in the text field.
- 7** If you have a PRICE column, enter price information for each configuration.

For more information, see [“Runtime Access to Your Pricing Information” on page 198](#).
- 8** Click Save.

Copying a Configuration Table

To use the same data in an existing Configuration table in a new Configuration table, you can copy a Configuration table. All table data and column definitions will be copied to the new table.

To copy a Configuration table

- 1** Select a Configuration table.
- 2** From the applet menu, choose Copy Record.

A New Configuration Table screen appears, displaying the data for the Configuration table you are copying.
- 3** Enter a new Configuration table name.

This value must be unique within the pageset.
- 4** Click Save.

The new record appears in the Configuration Table list.

To copy a Configuration table column

- 1 Select a Configuration table column.
- 2 From the applet menu, choose Copy Record.

A New Configuration Table Column screen appears, displaying data for the column you are copying.

- 3 Click Save.

Only the Configuration table column definition is copied.

To copy a Configuration table record

- 1 Select a Configuration table column record.
- 2 From the applet menu, choose Copy Record.

A New Configuration Table Column Record screen appears, displaying data for the record you are copying.

Deleting a Configuration Table

Use the following procedure to delete a Configuration table.

NOTE: You cannot delete the MAIN Configuration table.

To delete a Configuration table

- 1 Select a pageset.
- 2 In the Configuration Tables tab, select the Configuration table you want to delete and click Delete.
- 3 Validate the associated project to verify that deleting the Configuration table did not cause problems in the project.

Editing a Configuration Table

Use the following procedure to edit a Configuration table.

To edit columns in the Configuration table

- 1** Select a pageset.
- 2** In the Configuration Tables tab, select a Configuration table.
- 3** Select the Designer tab.
- 4** Select the row you want to edit in the Designer applet.
- 5** Make changes directly in the row and click Save.

To edit rows in the Configuration table

- 1** Select a pageset.
- 2** In the Configuration Tables tab, select a Configuration table.
- 3** Select the Editor tab.
- 4** Select the configuration row you want to edit in the Editor applet.
- 5** Make changes directly in the row and click Save.

Creating Exception Messages

Create exception messages to help customers purchase the right product or service. If a customer chooses an invalid combination, the application will display an exception guidance message. Using exception messages that clearly describe what is not available assists the user in determining what choice or choices need to be made.

[Table 2](#) lists some examples of informative exception messages.

Table 2. Examples of Informative Exception Messages

Less Informative	More Informative
“This combination is not available.”	“The XYZ network card is not compatible with the LMNO motherboard. Please select another card.”
“Model needs more RAM.”	“The ACME Laptop requires 256MB RAM.”
“That color is not available.”	“The ACME Tower Case is not available in teal. Please choose from our current selection of black, tan, and dark blue.”

Use exception messages to create up-sell and cross-sell messages as well. For more information, see [“Creating Cross-Sell and Up-Sell Messages” on page 84](#).

To create an exception message

- 1 In the Configuration Tables tab, select a Configuration table.
- 2 In the editor, select a configuration row.
- 3 In the Row Type column, click the button to open a picklist.
- 4 In the picklist, select EXCEPTION and click OK.
- 5 Enter values in input columns to determine what combination of data will create the exception message.

For example, if you have a column for PASSENGERS and a column for MODEL, you could enter 4 for PASSENGERS and sport for MODEL and create the Exception message, “The sport model does not have room for four passengers. Please select another model.”

- 6 In the Rule column, enter the exception message.

NOTE: It is recommended that every main Configuration table have at least one Exception row to catch all of the exceptions. While it is recommended that you create a separate Exception message for each Exception situation, you can catch all exceptions using the last exception row. Enter text that indicates which table the fall through came from. For example: MODEL_TRANSMISSION FALL THROUGH or MAIN FALL THROUGH.

Creating Cross-Sell and Up-Sell Messages

An eConfigurator application can act as a virtual sales assistant. If a user configures a product, the Sales application lets the user know that the selection resulted in a valid product. If a user selects a particular product, eConfigurator can also point the user to a better product for their needs or a product accessory. You provide this guidance by adding exception messages to your Configuration tables.

Cross-sells are accessories for a product or any special offers.

Up-sells are similar products, usually an upgrade to the currently configured product.

There are two steps for adding cross-sell and up-sell messages.

- Adding a cross-sell or up-sell message to your data model.

You will add the message to an output non-RULE column in the appropriate configuration rows.

- Adding a cross-sell or up-sell link.

You will add a LINK control to your Output UI. You will edit the file that displays your UI to provide a location where the message will appear followed by a link to the recommended cross-sell or up-sell items.

The procedures for each step are described in the following section. They describe adding an up-sell message, but the steps are the same for both cross-sell and up-sell messages. When adding a cross-sell message, substitute *up-sell* with *cross-sell* in the instructions that follow.

For information on creating cross-sells and up-sells that link to the server-based eConfigurator, see [“Example” on page 196](#) in [“Runtime Interaction with the Server-Based eConfigurator.”](#)

To add a cross-sell or up-sell message to your data model

- 1** In the Configuration Table designer, add two new rows above the RULE row.
- 2** Name one of the rows UPSELL_LINK.

This column requires the pageset ID of the linked pageset.
- 3** Name the other row “UPSELL_MSG.”

This column will include the up-sell message.
- 4** Enter a value of 0 in the Column Type field of both rows.
- 5** Select the Configuration table you want to add the up-sell message to and select the Editor tab.
- 6** In the UPSELL_LINK column, enter the pageset ID of the linked pageset in each row of valid configuration data for which an up-sell link is applicable.
- 7** In the UPSELL_MSG column, enter the text and any HTML formatting tags for the up-sell message.

To add a cross-sell or up-sell link

- 1** In the Output UI tab, create a new LINK control.
- 2** Choose a table and column.
- 3** In the Output UI record, check the Pageset column to indicate that the value in the column is a Pageset ID and not a URL.

Working with Configuration Tables

Creating Cross-Sell and Up-Sell Messages

Working with Pagesets

6

This chapter describes how to work with pagesets, including pageset creation, editing, and validation. It also describes how to search for pageset data.

About Pagesets

A pageset contains all the feature and configuration data for a product. A pageset also contains instructions for how that data should appear inside the completed application. Use a pageset to organize the product data and layout files for each selling category.

When you create a pageset, the following files are created and stored in the Siebel File System (Directory|File):

- ds|pageset_x.js
- pg|pageset_1.htm
- pg|pageset_2.htm
- pg|pageset_i.htm

For information about these directories and files, see the File Reference chapter in *Siebel Interactive Designer API Reference*.

About Managing the Display of Pagesets

When a user clicks Get Advice, the Advisor List appears and displays eAdvisor pagesets. You can control which pagesets display in this list by doing one of the following:

- By choosing “In-Development” in the Display Category column when defining the pageset record. This prevents the pageset from displaying in the Advisor List for all users. Use this method for pagesets or projects that are actively being developed or modified.
- By writing personalization rules that refer to the Display Name, Display Category, or Display Description. Write the personalization rules against the ISSCDA Personalized Contents List View and the ISSCDA Personalized Contents List Applet. For more information on defining personalization rules, see *Personalization Administration Guide*. Use this method to restrict visibility of pagesets based on user type.
- By leaving the Display Name blank when defining the pageset record. This prevents the pageset from displaying in the Advisor List for all users. Use this method for projects that contain pagesets that you do not want users to see.

When you import projects created prior to Siebel 7.5, or when you migrate projects from previous versions of Interactive Designer, *In-Development* is assigned as the value for Display Category, Display Name, and Display Description.

In previous releases, when the user clicked Get Advice, the Advisor List displayed the pagesets in the default project. If you want to display the default project rather than controlling pageset display as described above, locate the ISSCDA RT UI Service in Siebel Tools, and set the ShowPersonalListApplet user property to N. Then recompile the application .srf.

Viewing Your Pagesets

Interactive Designer provides two views of pagesets:

- **My Pagesets**

My Pagesets displays a list of pagesets for everyone in your group, or people that share your position in Siebel Applications. For example, if your eBusiness role is set to Sales Manager, by selecting My Pagesets you will see the pagesets for all sales managers.

- **All Pagesets**

All Pagesets provides a view of all pagesets for all positions.

NOTE: Use the Responsibilities view in Siebel Application Administration to set the visibility to the Interactive Designer views, such as the Table Editor views, based on the logged on user's ID (by associating the logon ID with a particular responsibility). This allows you to control what a user can do using the Interactive Designer user interface. For more information on setting up view access, refer to *Siebel Applications Administration Guide*.

To view your pagesets

- Select My Pagesets from the Show drop-down list.

NOTE: My Pagesets shows all of the pagesets created by people that share your position in Siebel Applications, so you will likely see more pagesets than those you have created.

To view the pagesets for a particular project

- In the Projects screen, click the Pagesets tab.

To view a particular pageset

- In the Pagesets screen, select a pageset row.

The More Info tab displays the pageset details.

NOTE: If you have a large list of pagesets, you can search for the pageset using the Siebel Search functionality. For more information, see [“Performing a Pageset Data Search” on page 97](#).

Locking Pagesets

Use the Lock feature to make sure that only one user is working on a pageset at a time.

For more information on useful features for working in a team environment, see [“Projects in a Team Environment” on page 60](#).

To lock a pageset

- 1 Navigate to My Pagesets.
- 2 In the Pagesets screen, select the pageset you want to lock.
- 3 In the More Info tab, check Locked.
- 4 Click Save.

Until you unlock the pageset, other users can view the pageset, but cannot edit it.

To unlock a pageset

- 1 Select My Pagesets from the navigation drop-down list at the top of the screen.
- 2 In the Pagesets screen, select the pageset you want to unlock.
- 3 In the More Info tab, deselect Locked.
- 4 Click Save.

Other users can now edit the pageset.

Creating a Pageset

Use the following procedure to create a pageset.

To create a pageset

- 1** From a project, click the Pagesets tab (or select My Pagesets from the Show drop-down list).
- 2** In the Pagesets tab (or Pagesets screen), choose New Record from the menu.
The New Pageset applet appears.
- 3** Enter a name that describes the pageset, for example, ACME Sedans.
The Pageset applet lists the pageset by this name.
- 4** Enter a pageset ID, for example, sedans.
All browser-based code uses the pageset ID to reference the pageset.
- 5** Click Pageset team to open a dialog in which you can select your position (pageset team).
This determines who can access the pageset from the My Pagesets list.
- 6** In the Display Category field, click the select button and choose a display category.
If you choose *In-Development*, this prevents display of the pageset in the Advisor List. An entry in this field is optional.
- 7** Enter a name in the Display Name field.
This name displays in the Advisor List. If you leave this field blank, the pageset does not display in the Advisor List.
- 8** Enter a description in the Display Description field.
This description displays in the Description field in the Advisor List. An entry in this field is optional.

- 9 If you are in the Pagesets screen, click Project to open a dialog and select the Project the pageset will be associated with.

NOTE: Each pageset is physically associated with one project, but can be referenced by other projects.

- 10 Enter any notes.
- 11 Click Save.

Associating a Pageset with a Product

In Siebel Product Administration, you can associate a product with a pageset.

To associate a pageset with a product

- 1 Navigate to Siebel Product Administration.
- 2 In the Products applet, select the product you want to associate with a pageset.
- 3 Click the Pageset field.
A Pageset list appears.
- 4 Select a pageset.
- 5 Click Save.

When a user selects the product in a quote or catalog and clicks Customize, the associated pageset appears.

Copying a Pageset

To use the same data from an existing pageset for a new pageset, you can copy a pageset. All table data, Input UI data, and Output UI data will be copied along with the pageset.

To copy a pageset

- 1 Select and lock a pageset.
- 2 From the applet menu, choose Copy Record.

A New Pageset screen appears, displaying the data for the pageset you are copying.

- 3 Enter a new pageset name and ID.

These values must be unique throughout the project. If you copy the pageset to a different project, you do not have to change the pageset name and ID.

- 4 Click Save.

The new record appears in the Pagesets list. The pageset files are created with file names that match the new pageset ID.

- 5 Deploy the pageset.

Because pageset files are only links to project files, you cannot copy them in the Pageset Files view. To copy a pageset file, use the following method.

To copy a file and associate it with a pageset

- 1 Make a copy of the file in the Project Files view.
- 2 In the Pageset Files view, click New.
- 3 In the Files dialog box, select the file.
- 4 Click Add.

To use the same data from an existing project in a new project, you can copy a project. All dependent tables and files will be copied with the pageset.

Deleting a Pageset

NOTE: Before deleting a pageset, make sure the pageset is not referenced from anywhere else in your application or other applications.

To delete a pageset

- 1** From a project, click the Pagesets tab (or select My Pagesets from the Show drop-down list).
- 2** Select the pageset you want to delete.
- 3** From the applet menu, choose Delete Record.
A dialog appears confirming that you want to delete the record.
- 4** Click OK.

Editing a Pageset

Use the following procedure to edit a pageset.

To edit a pageset

- 1** From the pagesets list in the Pagesets screen, select the pageset you want to edit.
- 2** Lock the pageset.
- 3** In the More Info tab, edit the pageset fields and click Save.

To edit the associated configuration, feature, input UI, output UI, and file information, click the relevant tabs. Lock the pageset before making edits to the associated information.

Validating a Pageset

Use the Validate feature to verify the referential integrity between feature and Configuration tables. Validate your Pageset data model regularly as you work and before you deploy a project.

To validate a pageset

- 1** In the Pagesets screen, select the pageset you want to validate.
- 2** Click Validate.

A list of validation errors appear in the Validation Results tab. The error identifies the pageset, table, column, and sequence of the error. A message describes the error.

- 3** Click an error to view it.

When you click an error, Interactive Designer opens the correct screen and tab and selects the record containing the error.

- 4** Fix the errors and return to the Validation Results tab to validate the pageset again.

When all errors are fixed, the Validation Results tab will show no error records.

Performing a Pageset Data Search

You can perform Interactive Designer data searches on the selected pageset in two ways:

- Use the Pageset Search tab
- Use the Query button

Pageset Search searches all the data within a pageset. Pageset Search automatically performs a floating search. A floating search is a search that assumes wild cards around the search text. For example, in a search for *ar*, all text containing *ar* is returned.

Query queries the data in the Pageset list.

To perform a search using the Search tab

- 1** Select the Pageset Search tab.
- 2** Enter the text you want to find.
- 3** Specify what to search: Cell Data, Column Names, or Table Names.

Select all three to search all three options.

- 4** Click Search.

All matches for your search appear in the Search Results applet.

- 5** Click a search result to open the view, pageset, and table that contains the search result.

To perform a search using the Query button

- 1** On the tab in which you want to search for data, click Query.

A Query screen appears that lists all the fields for that tab.

- 2** Enter data in the field you want to search by.

- 3** Click Go.

The Query returns the results for your search. All other records will not appear until you perform another search.

Working with Feature Tables

7

This chapter describes how to work with Feature tables. It describes basic functions, such as creating, designing, and editing Feature tables. It also describes more advanced ways of using Feature tables, such as linked Feature tables.

About Feature Tables

Use Feature tables to define the features and feature values of a product. For example, color is a feature and red, green, and blue are values for that feature. Each pageset in an Interactive Designer project includes one or more Feature tables for each feature related to that product.

Information in Feature tables can be used to populate the UI controls that appear on Display pages in your application. Use Configuration tables to define valid and invalid combinations of feature values. For more information, see [“Creating Configuration Tables” on page 75](#).

There are four types of Feature tables:

- Standard
- Linked
- Trigger
- Target

Standard Feature Tables

Use standard Feature tables to define your features.

Linked Feature Tables

Use Linked Tables when Feature table information needed for multiple UI controls is identical. For example, if you have an Interior Color Feature table for a Car pageset and you want to create an Exterior Color Feature table that uses the same set of color values, you can create the Exterior Color Feature table as a linked table. You cannot edit a linked table. To make changes to the table design or table data, edit the original table (in this example, the Interior Color Feature table).

Trigger and Target Feature Tables

Trigger and Target Feature tables are used to determine what values appear in a UI control based on a selection in another UI control. Based on the user's selection in a UI control tied to the Trigger table, different subsets of information (values a user can select) are displayed in the UI control tied to the Target table. The range of values in the target table are dynamically filtered based on the value selected from the trigger table. For more about Trigger and Target Feature tables, see [“About Trigger and Target Feature Tables” on page 116](#).

Feature Table Views

You can work with Feature tables in several views:

- Designer
- Editor
- Feature Table Column Manager

These views are tabs in the pageset view.

Designer View

In Designer view, you design the Feature table in which you will enter your feature data. When you select the Designer tab, the Feature Table Designer opens.

Each Feature table must contain at least three columns: CODE, DESC, and DEFAULT. When you create a Feature table in Interactive Designer, these columns are automatically created.

The following table describes the three columns.

Column	Description
CODE	Abbreviated value for the feature defined in the DESC column. Contains unique values (within the column) that identify the rows in the table.
DESC	Full-length text description of the feature value. This value represents the text that appears in input UI controls on display pages in the application.
DEFAULT	Defines the initial display value for an input UI control that takes its content from the Feature table. Type “default” in the row that represents the default feature value and leave all other cells in the column blank.

In the Table Designer view, you can add additional columns between the DESC and DEFAULT columns to represent other aspects of a feature, such as price or a part number. There can be as many columns to a table as there are values to a feature.

Editor View

After you have designed a Feature table, enter data in the table by switching to the Feature Table Editor view. In the Feature Editor view, you enter the row and cell data for Feature table columns. When you select the Editor tab, the Feature Table Editor opens.

Additionally, the Row type column provides a picklist for Target tables. (See [“Creating Trigger and Target Feature Tables” on page 118](#) for information on creating Trigger and Target Feature tables.) For all other Feature table types, use DATA, which is selected by default. The Sequence column determines the order in which the rows are published.

For more information on creating, designing, and populating Feature tables, see [“Creating Feature Tables” on page 103](#).

Feature Table Column Manager

If you have features that are common to several products you can use the Feature Table Column Manager to assign these features to products. In the Feature Table Column Manager, Feature table names display as columns. Products or other items you want to add display as rows.

You assign a feature to a product by placing a check mark in the desired Feature table column. This automatically adds the product or item as a column to the Feature table.

For more information on populating Feature tables using the Feature Table Column Manager, see [“Managing Feature Table Columns” on page 112](#).

Viewing Feature Tables

Use the following procedure to view a Feature table.

To view your Feature tables

- 1** Select the pageset containing the Feature tables you want to view.
- 2** Click the Feature Tables tab to view a list of Feature tables.
- 3** Select a Feature table from the list to view its details in the Editor applet.

NOTE: If you do not remember which pageset you defined the Feature table in, use Project Search to find the Feature table.

Creating Feature Tables

Creating a Feature table requires the following steps:

- Dividing your data into a set of features and selecting a default value for each.
- Creating a new Feature table.
- Designing the Feature table.
- Entering data in the Feature table.

To divide your products into features

- 1** Break down your product or situation in terms of selectable features.

For example, if your product is a sports car, make a list for each configurable feature. If the sports car comes with manual transmission, the user does not need to make a selection. Therefore, transmission is not a feature.

However, if the user has a choice in exterior color, you will need a Feature table to hold the possible values for exterior color. For this feature, you might name the Feature table EXT_COLOR.

These feature values may also serve as answers to questions in an eAdvisor application. For example, in response to the question, “What exterior color would you like your car to have?” a user can select from a drop-down list of exterior colors.

- 2** Determine the possible values for each feature.

For the exterior color feature, determine what colors are offered. Eventually you will populate the Feature table with these values, for example Red, Black, and Silver.

- 3** Determine the default value for each feature.

For example, you may decide that black will be the most commonly selected value for the exterior color feature.

NOTE: You can also use Trigger and Target Feature tables to determine default values based on previous user selections. For more information, see [“Creating Trigger and Target Feature Tables” on page 118](#).

Your compiled list of features and values for Sports Car might look like this:

Sports Car Features

- Exterior Color
 - Black (default)
 - Silver
 - Red

- Interior Color
 - Black (default)
 - Tan
- Sun Roof
 - No (default)
 - Yes
- Model
 - Basic (default)
 - Turbo

You will use this information when creating your Feature tables in the next steps.

To create a Feature table

- 1** From the navigation drop-down list, select My Pagesets.
- 2** Select a pageset.
- 3** Select the Feature Tables tab and, from the menu, choose New Record.
A new Feature Table window appears.
- 4** Enter a name, without spaces, for the Feature table.
The name will automatically convert to all capital letters. The name must start with a letter and can contain the letters A-Z, the numbers 0-9, and an underscore character (as in EXTERIOR_COLOR).
- 5** Select the type of table from the drop down list. The following are the table type values:
 - Linked
For more information, see [“Creating Linked Feature Tables” on page 109](#).
 - Standard

- Target

For more information, see [“Creating Trigger and Target Feature Tables” on page 118.](#)

- Trigger

For more information, see [“Creating Trigger and Target Feature Tables” on page 118.](#)

6 If you are creating a linked table, click the Linked To Table button to open a picklist from which you can select the table to link to.

7 Enter any notes. This step is optional.

8 Click Save.

To design the Feature table

1 Select the Designer tab.

The Feature Table designer opens.

2 In the Feature Table designer, enter sequence numbers. This is an optional step.

The sequence numbers determine the order in which the feature columns will appear in the Feature editor. These numbers are automatically generated, but you can overwrite them.

3 Enter a column name for each feature.

For example, for the CARS Feature table, you might enter columns for MODEL, COLOR, and PRICE.

- 4 Select a business component. This step is optional.

If the data you want to display is already available in the Siebel database, you can access the business component and field and select that data.

This column maps to the selected column (business component) in the Siebel database. For more information on using business components, refer to *Siebel Tools Reference*.

NOTE: If you select a business component and field, you cannot select a class and attribute. These options are mutually exclusive.

- 5 If you selected a business component in [Step 4](#), click Field Name Select to open a picklist. Select a field name from the business component you selected.
- 6 Check the Shared field to reference one row ID in a business component. This step is optional.

All columns in the Table Designer that reference the same business component and have the Shared field selected are populated with data when a value is selected for any one of them.

For example, you add the columns Name, Phone, Organization, and Address to your Feature table, and they all reference the Contact business component. Then when you switch to the Editor View and select a value for the Organization column, values for Name, Phone, and Address are filled in.

- 7 Select a class. This step is optional.

NOTE: If you select a class and attribute, you cannot select a business component and field. These options are mutually exclusive.

This column maps to the selected class in the Siebel database. For more information on classes and attributes, see *Product Administration Guide*.

- 8 If you selected a class in [Step 7](#), select an attribute for the class.

- 9 If the Feature table is a trigger table, you may need to choose a target table.

For more information, see [“Creating Trigger and Target Feature Tables” on page 118.](#)

- 10 Enter any notes and click Save.

To enter information in the Feature table

- 1 Select the Editor tab.

- 2 In the Feature Table editor, enter the sequence of the features as they will appear in the associated UI control. This is an optional step.

The sequence numbers are automatically generated. You can override them by entering a different number.

- 3 Enter a code value for the feature.

For example, for Black you might use the value BK and for Blue you might use the value BL. Make code values short so that they will be easy to identify in the Configuration table cells, but meaningful so that you do not need to reopen the Feature table to remember what the values stand for.

- 4 Enter a full description of the feature.

- 5 Fill in information for any other columns you have added.

- 6 Select the value you want to appear by default in the associated UI control and enter DEFAULT in the DEFAULT column.

- 7 If you have a PRICE column, enter additional costs for particular features.

For example, if a red car costs \$500 more than a black car, in the PRICE column for the RED row, you would enter 500. Later, in the PRICE column of the configuration table, you can add this column to the base price column. For more information, see [“Runtime Access to Your Pricing Information” on page 198.](#)

- 8 Click Save.

Creating Linked Feature Tables

Use Linked Tables when Feature table information needed for multiple UI controls is identical. For example, if you have an Interior Color Feature table for a Car pageset and you want to create an Exterior Color Feature table that uses the same set of color values, you can create the Exterior Color Feature table as a linked table.

You cannot edit a linked table. To make changes to the table design or table data, edit the original table. In this example, the Interior Color Feature table.

To create a linked Feature table

- 1** From the navigation drop-down list, select My Pagesets.
- 2** In the Pagesets screen, select the Feature Tables tab and click New.
A new Feature Table window appears.
- 3** Enter a name, without spaces, for the Feature table.
- 4** From the Type drop-down list, select Linked.
The Linked To Table button is enabled.
- 5** Click Linked To Table to open a picklist and select the table whose data you want to use.
- 6** Select a table and click OK.
- 7** Click Save.

The linked table appears in the Feature Tables list.

Copying a Feature Table

To use the same data in an existing Feature table for a new table, you can copy a Feature table. All table data and column definitions will be copied to the new table.

To copy a Feature table

- 1 Select a Feature table.
- 2 From the applet menu, choose Copy Record.

A New Feature Table screen appears, displaying the data for the Feature table you are copying.

- 3 Enter a new Feature table name.

This value must be unique within the pageset.

- 4 Click Save.

The new record appears in the Feature Table list.

To copy a Feature table column

- 1 Select a Feature table column.
- 2 From the applet menu, choose Copy Record.

A New Feature Table Column screen appears, displaying data for the column you are copying.

- 3 Click Save.

Only the Feature table column definition is copied. To copy the column data, use the following procedure.

To copy a Feature table record

- 1 Select a Feature table column record.
- 2 From the applet menu, choose Copy Record.

A New Feature Table Column Record screen appears, displaying data for the record you are copying.

- 3 Click Save.

Deleting a Feature Table

Use the following procedure to delete a Feature table.

To delete a Feature table

- 1 In the Pagesets screen, select a pageset.
- 2 In the Feature Tables tab, select the Feature table you want to delete.
- 3 From the applet menu, choose Delete Record.

NOTE: When deleting a Feature table, data that references the Feature table is not automatically deleted (for example, UI controls and Configuration tables). After deleting a Feature table, validate the project to make sure there are no validation errors.

Editing a Feature Table

Editing a Feature table requires two steps:

- Designing the Feature table.
- Editing the Feature table.

Procedures for these steps are described in the following section.

To design the Feature table

- 1 In the Pagesets screen, select a pageset.
- 2 In the Feature Tables tab, select a Feature table.
- 3 Select the Designer tab.
- 4 In the Designer applet, select the row you want to edit.
- 5 Make changes directly in the row and click Save.

To edit rows in the Feature table

- 1** In the Pagesets screen, select a pageset.
- 2** In the Feature Tables tab, select a Feature table.
- 3** Select the Editor tab.
- 4** Select the feature row you want to edit in the Editor applet.
- 5** Make changes directly in the row and click Save.

Managing Feature Table Columns

Feature table columns are maintained and managed by the Feature Table Column Manager. If you have features that are common to several products you can use the Feature Table Column Manager to assign these features to products.

Before using this feature, you must define Feature tables.

In the following procedure, products are used as the item added to the feature table. The procedure applies to any of the items you can add to a feature table, such as recommendations and instructions. You are not limited to products.

To manage Feature table columns

- 1** In the Pagesets screen, select a pageset.
- 2** Select the Feature Table Column Manager tab.

The Feature Table Column Manager appears. The columns display the names of the Feature tables you have created.

- 3** To add a product to the table, click New and enter the name of the desired product.
- 4** To assign a feature to a product, click in the desired column.

A check mark appears to indicate the feature has been assigned to the product. This adds the product to the Feature table.

- 5** Repeat these steps until you have assigned the desired features to products.

- 6** Click the Feature Tables tab.

The Feature Tables view displays.

- 7** Select the desired Feature table, and then click the Designer tab.
- 8** Verify that all the products you assigned in the Feature Table Column Manager display correctly.

This chapter describes advanced techniques for working with your Interactive Designer data model. These techniques include:

- Adding Trigger and Target controls to determine what sets of values are dislodged in a UI control based on a previous selection.
- Adding dynamic default UI controls that display logical values for the user given previous selections.
- Using Subconfiguration tables to create parent and child pagesets.
- Creating JavaScript conditional statements.

About Trigger and Target Feature Tables

Use Trigger and Target Feature tables when you want the values to appear in a UI control to change depending on a selection in another UI control. For example, if you want to display a set of values in either inches or centimeters, use a UI control tied to a Trigger table to allow a user to select Metric or Imperial. Based on that selection, a drop-down list tied to a Target table displays a set of values for Centimeters or a set of values for Inches. This relationship is shown in [Figure 18](#).

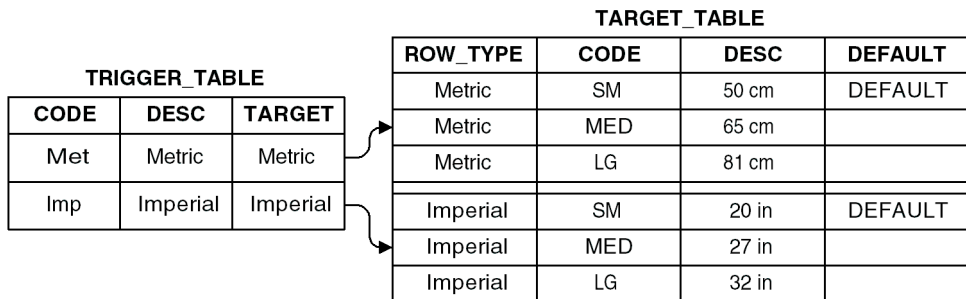


Figure 18. Trigger and Target Features Tables

Target tables allow you to create multiple sets of data (row types) for the same table. The Target table is a Feature table whose rows are defined by one or more row types. A row type is one particular set of values that will appear if the user selects a certain value from the trigger table.

The Trigger table defines a relationship between its Code values and the named row types in the Target table. The field values in a Trigger table column are used to dynamically set the active row set for the Target table.

In the Trigger and Target example of a drop-down list that displays metric and imperial measurements, the UI control tied to the Trigger table gives the user the option of selecting Metric or Imperial. The UI control tied to the Target table displays one of two sets of values:

- If the user selects Metric, the Trigger table references the Metric Row Type in the Target table and displays the metric values 50 cm, 65 cm, 81 cm.
- If the user selects Imperial, the Trigger table references the Imperial Row Type in the Target table and displays the imperial values 20 inch, 27 inch, 32 inch.

In the Target table, there are two row types: Metric and Imperial. You create these row types in the Target Feature table editor. By default, the Feature table designer creates a row type named table_DATA, where table is the name of the table. The CODE values for both sets are the same (for example, SM, MED, LG). The Description column contain the values for each set of data. Data in the Configuration table references only the single set of CODE values, so no extra work is required in the Configuration table to make reference to multiple row types in the Target table.

To determine which row type appears, the Trigger table contains a trigger column that references the Target table. This column contains the names of the row types: Metric and Imperial.

NOTE: A browser-based application offers unconstrained selections, that is, users can move freely from UI control to UI control, making selections from a complete list of options, and can go back and change selections without restriction. Trigger and Target Feature tables should be used only in cases where they best serve the user.

Creating Trigger and Target Feature Tables

Creating trigger and target feature tables requires the following steps:

- Creating a target feature table.
- Creating row types for the target feature table.
- Populating the target feature table.

See [“Editing a Feature Table” on page 111](#) for instructions on completing this step.

- Creating a trigger feature table and tying it to the target table.
- Tying each table to a UI Control.

Each of these steps is explained in the following procedures.

Creating Target Tables

Create the target tables first.

To create a target table

- 1** In the Pagesets screen, select the Feature Tables tab.
- 2** Choose New Record from the Feature table applet menu.

A new Feature Table window appears.

- 3** Enter a name, without spaces, for the Feature table.
- 4** From the Table Type drop-down list, select Target.
- 5** Click Save.

To create new Row Types for the Target table

- 1 Select a Target table.
- 2 Select the Editor tab.
- 3 In the Editor, click New.
A new row appears in the editor.
- 4 Click the Row Type picklist icon to open the Feature Table Rowset picklist.
- 5 Select a Rowset or click New to open a dialog in which you can create a new rowset name.
- 6 Click OK.

Creating Trigger Tables

The four types of feature tables are described in [“About Feature Tables” on page 100](#).

To create a trigger table

- 1 In the Pagesets screen, select the Feature Tables tab.
- 2 From the Feature Table applet menu, choose New Record.
A new Feature Table window appears.
- 3 Enter a name, without spaces, for the Feature table.
- 4 From the Table Type drop-down list, select Trigger.
- 5 Click Save.
- 6 Select the Designer tab.
- 7 From the applet menu, choose New Record.
- 8 Enter a column name.

This column will link to the Target table.

- 9 In the YEAR column, click Target Table to open the Target Table picklist and select the Target table you created in the previous procedure. Switch to Edit mode.
- 10 Follow the steps in [“To enter information in the Feature table” on page 108](#).

NOTE: In the column tied to the Target table, enter row types.

To tie the Trigger and Target Feature tables to UI controls

- Follow the steps in the procedure [“About Creating Input UI Controls” on page 140](#) to create a UI control to display the trigger table values and a UI control to display the target table values.

NOTE: Any UI control may be used as a trigger, but only list boxes can be the target of that trigger.

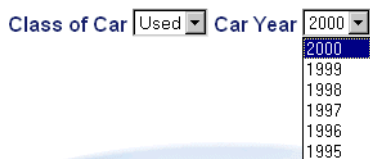
Trigger and Target Example

In this example, you will add Trigger and Target Feature tables that populate a drop-down list with appropriate year values, based on whether a user wants to purchase a new or used car.

At runtime, if the user selects New for Class of Car, they have one option in the Car Year drop-down list.



If the user selects Used, they can choose from a list of year values.



To create the Trigger and Target example:

- Create a Target table named YEAR.
- Create a Trigger table named CLASS and link it to the target table.
- Create UI controls that link to these feature tables.

Each of these steps is explained in the following section.

Creating the YEAR Target Table

Creating the YEAR target table requires three steps:

- Creating a target table named YEAR.
- Creating row types for the target table.
- Populating the target table with values.

Each of these steps is explained in the following procedures.

To create the YEAR target table

- 1** In the Pagesets screen, select the Feature Tables tab.
- 2** Choose New Record from the applet menu.
A new Feature Table window appears.
- 3** Enter a name, without spaces, for the Feature table.
- 4** From the Table Type drop-down list, select Target.
- 5** Click Save.

To create the row types for the YEAR Target table

- 1** In the Feature Tables tab, select the YEAR Target table.
- 2** Select the Editor tab.
- 3** In the Editor, click New.
A new row appears in the editor.

4 Click the Row Type picklist icon to open the Feature Table Rowset picklist.

5 Click New.

The New Rowset dialog appears.

6 In the Name text box, enter YEAR_NEW.

7 Click Save.

8 Repeat this procedure and create a second new rowset named YEAR_USED.

To populate the YEAR target table

1 In the Feature table editor, create a row with values for the YEAR_NEW rowset.

2 In the Feature table editor, create a row with values for the YEAR_USED rowset.

Creating the CLASS Trigger Table

Use the following procedure to create the CLASS Trigger Table.

To create the CLASS Trigger Table

1 In the Pagesets screen, select the Feature Tables tab.

2 From the Feature Table applet menu, choose New Record.

A new Feature Table window appears.

3 In the Name text box, enter CLASS.

From the Type drop-down list, select Trigger.

4 Click Save.

5 Select the Designer tab.

6 From the applet menu, choose New Record.

7 In the Column Name text box, enter YEAR.

In the Designer tab, the Target Table button is enabled.

8 Click the Target Table button to open the Target Table picklist.

- 9 Select the YEAR target table and click OK.
- 10 Follow the steps in [“To enter information in the Feature table” on page 108.](#)

NOTE: In the column tied to the Target table, you can only enter row types.

Creating UI Controls to Display the Trigger and Target Values

Use the following procedure to create UI controls to display the Trigger and Target values.

To tie the Trigger and Target Feature tables to UI controls

- 1 In the Input UI tab, click New.
- 2 Enter values in the new row.
- 3 Click Save.
- 4 Click New again.
- 5 Enter the following values for the second row.
- 6 Click Save.

Additional Trigger Capabilities

As mentioned in [“Trigger and Target Feature Tables” on page 101](#), a change in a trigger may cause a change in the set of options presented in one or more target choices. Two variations on the supported trigger and target functionality are multi-variable triggers and cascading triggers. For these variations, see [Appendix A, “Implementation of Multi-Variable and Cascading Triggers.”](#)

Dynamic Defaults

When Interactive Designer generates an Input UI page, each UI control in the page automatically defaults to the value specified as default in the corresponding Feature table. Default settings in the pageset can be overridden with the dynamic default functionality. This is the equivalent of changing which line in a Feature table is the default.

Using dynamic defaults, you can link to a recommended solution from a Results page and override the default settings for the UI controls on the new Display page. You can set the UI control values to new defaults based on the responses to eAdvisor questions. In the car example, if the user specified that they were interested in a two-seater model, in the new Input UI page for the recommended product you can automatically set a UI control displaying number of passengers to 2.

Creating Dynamic Defaults

Use the following steps to create dynamic defaults.

- Determine what UI Controls (and the Feature tables that populate them) need to be changed on the new Input UI page to reflect the choices made at the Needs Analysis level.
- In a Configuration table, create an output column to store pageset IDs or URLs of recommended products.
- In the same table, create an output column to store the defaults to change on the recommended Inputs page.
- Create a link target to reference the recommended product column and dynamic default column.

The following procedure is one way to create dynamic defaults.

NOTE: Dynamic Defaults requires the creation of a second pageset. Refer to the *LoadPageset* and *LoadPagesetWithDynDefObj* sections in *Siebel Interactive Designer API Reference*.

To create dynamic defaults

- 1 In a Configuration table, create a DYNDEFS column.

A Pagesetname column is not necessary because the PagesetName will be part of the string in the DYNDEFS column.

- 2 In the DYNDEFS cells, enter the pagesetID and desired Feature table values in the format:

```
PagesetID', '1stFTName=value, 2ndFTName=value, nthFTName=value
```

- 3 In the Output UI control:

- a Select the Control Type LINK.
- b In Label, enter the text you want to appear before the link.
- c In Table and Column, reference the Configuration table and DYNDEFS column.
- d Select Pageset.
- e In Anchor Text, enter the text that will appear as the link.

Notice the single quote, comma, single quote in-between *PagesetID* and *1stFTName* in [Step 2](#). Because the wrapper will put the outside quotation marks around the string,

```
document.write(ISS.BuildTarget("LINK",window,"DYNDEFS",true));
```

becomes

```
document.write(ISS.BuildTarget("LINK",window,"PagesetID', '1stFTName=value,2ndFTName=value",true));
```

NOTE: When used in Feature tables, DYNDEF is reserved and refers to a subconfiguration child's dynamic defaults. In Configuration tables, it is not reserved, but you should not use DYNDEF in more than one Configuration table. The column names in Configuration tables are global, so you will only end up with one DYNDEF output even if you have multiple instances of the column. Instead of using DYNDEF in Configuration tables, use more descriptive names, such as UPSELL_DYNDEF.

Example of Dynamic Default

In this example, a needs analysis page is used to recommend a particular computer, including the model, processor, and memory.

To create the dynamic defaults example

- 1 Add a drop-down list UI control to the needs analysis page.
- 2 Create a Feature table to populate the drop-down list with the values:
 - Laptops
 - Desktops
 - Work stations
- 3 In a Configuration table that references the Feature table you created in [Step 2](#), add an output column named REC_PROD to store the pageset ID of the recommended product.
- 4 In the same Configuration table, add an output column named DYNDEFS to store the defaults to change on the linked-to display page.
- 5 Fill the REC_PROD and DYNDEFS columns as shown in the following table.

The REC_PROD column stores the pageset IDs of the recommended products.

The DYNDEFS column includes references to the Feature table names used by the UI controls and the new CODE value to serve as the default value. The equal sign (=) separates Feature table names and CODE values. A comma (,) separates multiple dynamic defaults.

For example, for the laptops pageset ID, the first value under DYNDEFS references the MODEL Feature table in the next pageset that is going to be displayed, and sets the new default value to 1000.

REC_PROD(0)	DYNDEFS(0)
laptops	MODEL = XR50,PROC = 366,MEM = 32
laptops	MODEL = XR80,PROC = 400,MEM = 64

REC_PROD(0)	DYNDEFS(0)
desktops	MODEL = XP50,PROC = 366,MEM = 32
workstations	MODEL = XP80,PROC = 366,MEM = 64

- 6 Create a Link Output target on the display page of the Needs Analysis pageset that links to the recommended Inputs page.

The Output UI tab does not generate the DYNDEFS variable. For this reason, you will need to edit the code in the Pageset_2.htm file to add it. Use the syntax:

```
<SCRIPT>
document.write( ISS.BuildTarget( "LINK",window, "REC_PROD",true, "DY
NDEFS" ) );
</SCRIPT>
Click here to further configure your computer.
</A>
```

This overrides the default settings for the pageset being loaded by referencing the DYNDEFS column.

For more information on using the BuildTarget function to create Link Output targets, refer to the Pageset Functions chapter in *Siebel Interactive Designer API Reference*.

The Result of the Example

If the user's selections match the recommended solution described in the Configuration table, a link will appear to further configure the computer. When a shopper clicks the link, a display page for the pageset recommended in the REC_PROD column will appear. Also, the MODEL UI control will be set to the CODE value indicated in the DYNDEFS column. For example, the default MODEL description is 1000, but if the user is recommended the Laptop 2000, the UI control will display 2000.

Working with Subconfiguration

Use subconfiguration to create a master pageset that is comprised of many separate pagesets. Subconfiguration allows for a pageset to be created once and reused in many places within the application. Using subconfiguration allows you to:

- Provide conditional messaging information
- Separate data into smaller size pagesets
- Provide users with simultaneous multiple configurations

Example

In the following example, there are three complete pagesets, including data and UI:

- Computer

The computer pageset has selections for processor speed and hard-drive size.

- PCI-1

The PCI-1 pageset allows the user to configure Ethernet cards.

- PCI-2

The PCI-2 pageset allows the user to configure SCSI cards.

To link these pagesets together in a subconfigured mode, use the following procedure.

To create a subconfiguration

- 1** Create pagesets complete with data and UI.
- 2** Determine which pageset is the PARENT, for example Computer.
- 3** Determine which pagesets are the CHILDREN, for example PCI-1 and PCI-2.
- 4** Create a Feature table in the PARENT pageset, for example PCI_CARD.
- 5** Inside the Feature table, create a column called CHILD.

CHILD is a reserved name.

- 6** In the CHILD column, enter the name of the CHILD pagesets, for example PCI-1 and PCI-2. The following table shows the complete data to enter into the Feature table.

CODE	DESC	CHILD	DEFAULT
0	None	null	DEFAULT
PCI-1	Ethernet Card	PCI-1	
PCI-2	SCSI Card	PCI-2	

- 7** In the Inputs page for the Computer pageset, create a BuildWidget UI control that calls its values from the PCI_CARD Feature table. For information on BuildWidget, see *Siebel Interactive Designer API Reference*.

When the user selects Ethernet card, the PCI-1 pageset is instantiated as the CHILD. If the user changes their mind and selects SCSI card, the PCI-2 pageset is instantiated and replaces the PCI-1 pageset Feature table values with PCI-2 values.

- 8** To link the pagesets, you can either:
- Use the Subconfiguration and Optional Subconfiguration links in the BuildTarget API. For more information about using these links, refer to the Pageset Functions chapter in *Siebel Interactive Designer API Reference*.
 - Use references to Feature tables.

About Referencing Feature Tables

To reference a pageset in a subconfigured data model, you can reference the Feature table that has the CHILD column that instantiates the pageset. In the example above, you would use PCI_CARD:FEATURE_TABLE_NAME.COLUMN_NAME to reference any of the Feature tables in the PCI-1 or PCI-2 pagesets.

Example

This example uses a subconfigured data model where RACK is the name of the top level, SERVER1 is the next level, and SOFTWARE is the third level.

From the Rack Pageset

- To reference variables within SERVER1, use `SERVER1:TABLE_NAME.COLUMN_NAME`
- To reference variables within SOFTWARE, use `SERVER1:SOFTWARE:TABLE_NAME.COLUMN_NAME`

From the SERVER1 Pageset

- To reference variables within the RACK, use `TOP:TABLE_NAME.COLUMN_NAME`
or `PARENT:TABLE_NAME.COLUMN_NAME`
- To reference variables within SOFTWARE, use `SOFTWARE:TABLE_NAME.COLUMN_NAME`

From the SOFTWARE Pageset

- To reference variable within the RACK, use `TOP:TABLE_NAME.COLUMN_NAME`
or `PARENT:PARENT:TABLE_NAME.COLUMN_NAME`
- To reference variables within SERVER1, use `TOP:SERVER1:TABLE_NAME.COLUMN_NAME`
or `PARENT:TABLE_NAME.COLUMN_NAME`

About Setting Defaults

In a Subconfiguration data model, defaults are defined in the following places:

- An external pageset
- The DYNDEF column of a Feature table that instantiates a CHILD
- The Feature tables within Interactive Designer

In an External Pageset

Each of the Feature tables that need to get defaulted in the Subconfiguration data model must be explicitly referenced (including the TOP items). Using the example above with RACK, SERVER1, and SOFTWARE, the syntax would be as follows:

```
FTN = FEATURE_TABLE_NAME
```

```
CV = CODE VALUE
```

```
TOP:FTN=CV , TOP:SERVER1:FTN=CV , TOP:SERVER1:SOFTWARE:FTN=CV
```

In the DYNDEF Column

You can set defaults in the DYNDEF column of a Feature table that instantiates a CHILD. See [Table 3](#) for sample values.

Table 3. PCI_CARD

Code	Desc	Child	Dyndef	Default
0	None	null		DEFAULT
PCI1	Ethernet Card	PCI-1	SPEED = 4	
PCI2	SCSI Card	PCI-2	SIZE = 2	

DYNDEF, like CHILD, is a reserved column name for Feature tables. In this column, you can specify which defaults will load into the CHILD. A CHILD pageset will only get defaulted once. The use case is when the user selects the Ethernet Card, opens the PCI-1 pageset and changes SPEED from 4 to 3. Now the user changes the PCI_CARD UI control to SCSI Card, then back to Ethernet Card. Which value should SPEED have? The engine will retain the user's selection, and therefore not re-default the pageset in any way.

The same pageset can get defaulted different ways depending on how you instantiate the CHILD. If the computer example had two PCI Card slots, you could use the data as shown in [Table 4](#) and [Table 5](#):

Table 4. PCI_CARD1

Code	Desc	Child	Dyndef	Default
0	None	null		DEFAULT
PCI1	Ethernet Card	PCI-1	SPEED = 4	
PCI2	SCSI Card	PCI-2	SIZE = 2	

Table 5. PCI_CARD2

Code	Desc	Child	Dyndef	Default
0	None	null		DEFAULT
PCI1	Ethernet Card	PCI-1	SPEED = 2	
PCI2	SCSI Card	PCI-2	SIZE = 1	

About Accessing Model Variables

Having stored your data in different places within the same user session, it is important to understand which variables are accessible at which points.

In Configuration Tables

When creating Configuration tables, you can use any Feature table within the pageset as well as any Feature tables in the CHILD and below. This is the way to enforce different rules on the same CHILD pageset depending on where it gets instantiated.

If you are using a CHILD variable as an input column in a Configuration table in the PARENT, be aware of the row matching algorithm. If the CHILD is not instantiated (which it is not upon initial load of the PARENT pageset), the value of the input column variable will be null. If you do not account for this, you could have some trouble getting a valid match in your Configuration table. Using range values like (= *) or (! = 3,4) will match even if the input column variable is null. If you want to make sure the CHILD is not instantiated, you could type the word “null” into the cell of the Configuration table to try to get a match.

In OL_CONDITIONS and Cell Functions

OL_CONDITIONS and Cell Functions provide access to any variable on any pageset. Be careful with this operation, because the order of execution will have significant impact on your cell functions. As with any operation in the pagesets, make sure that the variables you are setting and the variables you are calculating are getting performed in a clear, predictable order based on the engine’s order of execution. A cell function in a CHILD pageset that references an output variable in the PARENT will not work. The cell function will calculate before the output variable is set, causing the application to be out of sync.

Performance Considerations

When designing a subconfigured data model, use the following guidelines for optimum performance:

- To increase the speed of initial load, start the system with as few children instantiated as possible.
- To increase the speed of click to click time, limit the UI control selections to only instantiate one CHILD at a time.

Creating Javascript Conditional Statements

Create an OL_CONDITIONS Configuration table to evaluate the current input UI control selections in the application against javascript conditional statements. These statements are evaluated and, based on the results, a message can be presented to the user.

For information on how the OL_CONDITIONS Configuration table is evaluated, see [“The Configuration Matching Process” on page 65](#).

To create an OL_CONDITIONS Configuration table

- 1** Create a Configuration table.

See [“Creating Configuration Tables” on page 75](#) for more information.

- 2** Create an input column named TEST.

- 3** Create an output column named RULE.

- 4** In the TEST column, enter a JavaScript conditional statement that returns true or false.

The conditional statement may refer to column values from Feature tables or Configuration tables using the same syntax used in cell functions.

- 5** In the RULE column, enter the message that appears if the TEST expression returns true.

Each row in the OL_CONDITIONS table is evaluated in order of SEQUENCE number, from smallest to greatest, against the current user-selected configuration.

If all of the TEST expressions in the OL_CONDITIONS table return false, the engine considers the user-selected configuration valid and ends the configuration matching process. If a TEST expression returns true, the exception message defined in the associated RULE column appears in the application.

Example

The following table shows an example of a JavaScript conditional statement in the TEST column and the message in the RULE column that will appear if the statement is true.

TEST	RULE
(APPLES.QTY + ORANGES.QTY < BANANAS.QTY)	Your smoothie needs more bananas.

Duplicate Configuration Column Names

If two Configuration column names have the same name, the latter column name in the evaluation cycle will overwrite the value from the first column. In case analysis, you can use this method to evaluate one subtable or another based on input values. Since only one of the tables will get evaluated, each of the options should have the same outputs, and you need to duplicate column names. Only one of the columns will get evaluated for any particular set of selections.

This chapter describes how to use the Input UI and Output UI tabs in Interactive Designer to build the UI for your application. It describes how to select the UI controls that will appear on your display pages, how to generate the necessary code, and the parts of the resulting display pages.

About Building Your UI

While you can customize the application UI in an HTML editor, Interactive Designer offers the capability to build a UI layout that consists of three areas (see [Figure 19](#)):

- Contents List
- Input UI display page
- Output UI display page

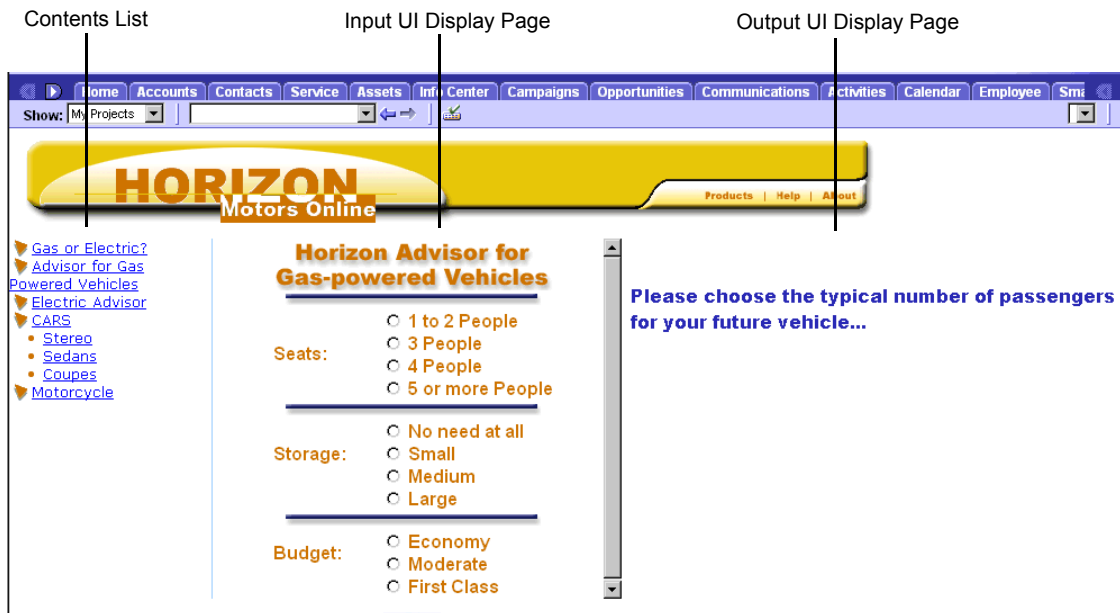


Figure 19. The UI Layout of a Browser-Based Application

Contents List

The contents list is a collapsible list that allows you to navigate an application. Clicking an item in the list loads the display pages associated with the item by linking to the appropriate pageset. By default, the contents list appears to the left of the display pages, as shown in [Figure 19](#). You can edit the position of the contents list in an HTML editor.

For information on how to create a contents list, see [“Working with the Contents List” on page 155](#).

Input UI Display Page

Use the Input UI tab in Interactive Designer to create the HTML file that is loaded into the Input UI Display area. This file is named `pagesetName_1.htm` (referred to as the Input UI display page in this guide), and displays Input UI controls, such as checkboxes. This page presents product features and allows users to make feature selections. The UI controls you create in the Input UI display page reference a Feature table that provides their values. The GETTEXT UI control is an exception. The GETTEXT UI control specifies a text box ID instead of a Feature table.

Selecting an input UI control value on the Input UI display page shows the result as an output UI control on the Output UI display page. For example, in [Figure 19 on page 138](#), the Input UI display page contains the selection for exterior color (red), and the Output UI display page shows an image of a red car.

For information on how to create the Input UI display page, see [“Generating Your Input UI Display Page” on page 146](#).

Output UI Display Page

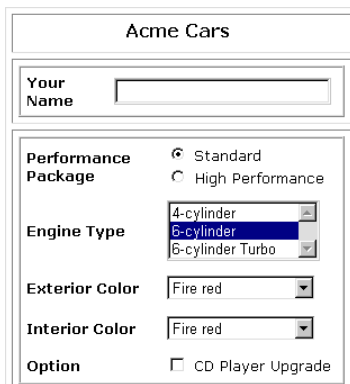
Use the Output UI tab in Interactive Designer to create the HTML file that is loaded into the Output UI Display area. This file is named `pagesetName_2.htm` (referred to as the Output UI display page in this guide) and displays the following items:

- Output UI controls, such as images representing a selected choice, display the recommendations to a valid feature selection made with an input UI control.
- Exception messages, usually text warnings, guide users to a valid product configuration when they choose feature selections that do not match an available product. You can also use exception messages to cross-sell and up-sell. For more information, see [“Creating Cross-Sell and Up-Sell Messages” on page 84](#).

For information on creating an Output UI Display page, see [“Generating Your Output UI Display Page” on page 153](#).

About Creating Input UI Controls

An Input UI control appears in the Input UI display page and allows users to select features. [Figure 20](#) displays the Input UI controls.



The screenshot shows a form titled "Acme Cars" with the following controls:

- Your Name:** A text input field.
- Performance Package:** Radio buttons for "Standard" (selected) and "High Performance".
- Engine Type:** A list box with options "4-cylinder", "6-cylinder" (highlighted), and "6-cylinder Turbo".
- Exterior Color:** A dropdown menu with "Fire red" selected.
- Interior Color:** A dropdown menu with "Fire red" selected.
- Option:** A checkbox for "CD Player Upgrade".

Figure 20. Example of Input UI Controls

The Feature tables contain the choices that appear in the input UI control, for example, the various exterior color options in the car example. The UI control row in the Input UI tab defines the control type, for example, a list box. It also controls where on the page it appears, based on the sequence number you specified in the UI control row in the Input UI tab.

The controls on an Inputs page are created using Interactive Designer functions that are generated from the information you enter in Interactive Designer Feature tables. Therefore, unlike standard HTML controls, the user interface and feature data are separate, and you can make changes to one without affecting the other.

Creating Input UI Controls

Use the following procedure to create input UI controls.

To create input UI controls

- 1 Create a Feature table whose Description values will populate the control.

For more information, see [“Creating Feature Tables” on page 103](#).

- 2 In the Pageset screen, select a pageset.

- 3 In the Input UI tab, click New.

A new row appears.

- 4 Enter the sequence number in which the control will be laid out on the page.

Enter a number relative to the numbers you entered for other controls to determine the sequence in which the control appears on the Input portion of the page. Enter numbers such as 10, 20, 30 and so forth. This allows you to later position a control between existing controls.

- 5 From the drop-down list, select the control type. They are:

- Check box

Allows a user to select one or more features by checking one or more of these controls. For check boxes, your Feature table should contain only two rows. The first row must have a CODE value of F, FALSE, or 0 to indicate the unchecked state. The second row must have a CODE value of T, TRUE, or 1 to indicate the checked state. The DESC value from the TRUE row will be used as the label for the check box in the UI.

The following example references four Feature tables with two rows each.

Options

- Convenience Package
- Security Package
- Leather Trim Package
- Sporty Package

- Get Text

Allows a user to enter a text string.

However a customized license plate will be standard with every Electric Roadster. Enter your new License plate:

- List Box

Allows a user to select from a drop-down list of values.

Class of Car

- Radio

Allows a user to select one feature from two or more features by clicking one of these controls.

Budget: Economy
 Moderate
 First Class

- Image Maps

Allows a user to select from an image.

6 Enter a label for the control. For example, for a list box of colors, enter the label “Select Interior Color.”

7 Click Feature Table to open a picklist and select the Feature table whose Description values will populate the control.

8 For textbox controls, enter a name.

Because text boxes do not map to Feature tables, a name is required to reference this control.

- 9** Enter a height and width for the control if applicable.

The number you enter in Height specifies the number of visible rows for the list box. The default value, 1, creates a drop-down list.

The number you enter in Width specifies the number of characters visible in the default browser font.

- 10** Select or deselect the Prefill check box to determine the width of a list box.

If Prefill is selected, the value you entered for Width is used to determine the width of the list box (if the specified width is shorter than the longest item in the list, the box uses the width for the longest item in the list instead). If you deselect Prefill, the list uses the width for the longest item in the list.

- 11** Supply the Map Filename, Map Shape, and Map Coordinates columns for image maps.

The feature table specified in the Input UI record must contain three columns that hold the values for the Map Filename, Map Shape, and Map Coordinates. The column names (IMAGE, SHAPE, COORDS, for example) in the feature table are entered in the corresponding fields. These three columns populate the following HTML constructs:

- Map Filename populates the filename specified in the `` tag used in the image map. CA.jpg is an example of a map filename. This image must reside in the PG directory for the project.
- Map Shape populates the parameter for the SHAPE attribute in the `<AREA>` HTML tag. POLY is an example of a map shape. Refer to an HTML reference book for all the SHAPE attributes.
- Map Coordinates populates the parameter for the COORDS attribute in the `<AREA>` HTML tag, for example, "9,67,37,76,30".

- 12** Click Save.

- 13 Click Generate to add the new UI controls to the Input UI display page.

NOTE: It is recommended that you build your UI using the Input UI and Output UI tabs in Interactive Designer. Once the functionality in your application is set, it is recommended that you customize your UI in a third party HTML editor. If you do use a third party HTML editor, any custom code you write between the BEGIN GENERATE HTML and END GENERATE HTML comments are deleted in the pg|pageset_1.htm file. Be sure to write any custom code outside of this area.

Copying Input UI Controls

Use the following procedure to copy an Input UI control.

To copy an Input UI control

- 1 Select an Input UI control.
- 2 From the applet menu, choose Copy Record.

A New Input UI Control screen appears, displaying the data for the Input UI control you are copying.

- 3 Enter a new number for the layout sequence.
- 4 Click Save.

The new record appears in the Input UI Control list.

- 5 Click Generate.

Deleting Input UI Controls

Use the following procedure to delete an input UI control.

To delete an Input UI control

- 1 Select a pageset.
- 2 In the Input UI tab, select the Input UI control you want to delete.
- 3 From the applet menu, choose Delete Record.
- 4 Click Generate.

Editing Input UI Controls

Use the following procedure to edit an input UI control.

To edit an Input UI control

- 1** Select a pageset.
- 2** In the Input UI tab, select the Input UI control you want to edit.
- 3** Make edits directly to fields in the row.
- 4** Click Save.
- 5** Click Generate.

Generating Your Input UI Display Page

To update your changes to the UI layout, you need to generate the UI files. Previewing or deploying your application will not automatically update UI changes.

To generate your input UI display page

- 1** From the navigation drop-down list at the top of the screen, select My Pagesets.
- 2** Select the pageset you want to generate your input UI for and click the Input UI tab.
- 3** Create UI controls.
- 4** Click Generate to add the new UI controls to the Input UI display page.

NOTE: It is recommended that you build your UI using the Input UI and Output UI tabs in Interactive Designer. Once the functionality in your application is set, it is recommended that you customize your UI in a third party HTML editor. If you do use a third party HTML editor, any custom code you write between the BEGIN GENERATE HTML and END GENERATE HTML comments is deleted in the pg|pageset_1.htm file. Be sure to write any custom code outside of this area.

About Creating Output UI Controls

Output UI controls appear on an Output UI display page whenever the selected values of all the input UI controls match a *valid configuration*. Output UI controls can be text, links, or images. [Figure 21](#) is an example of output UI controls.

Hi

This car matches your selections:



Your Current Selections:

Standard performance package
6-cylinder engine
Fire red exterior color
Fire red interior color
Stereo w/o CD player

Base Price:	\$ 15499
Upgrades:	\$ 1099
Total Price:	\$ 16598

Figure 21. Output UI Controls

The Output UI display page also displays *exception messages* when a user's selections do not match a valid configuration. These messages are designed to guide you to selections that match a valid configuration.

Creating Output UI Controls

Use the following procedure to create output UI controls.

To create output UI controls

- 1** Create the Feature or Configuration table containing the values that will populate the control.

For more information, see [“Creating Feature Tables” on page 103](#) and [“Creating Configuration Tables” on page 75](#).

- 2** In the Pageset screen, select a pageset.
- 3** In the Output UI tab, click New.

A new row appears.

- 4** Enter a sequence number for the layout of the control on the page.

Enter a number relative to the numbers you entered for other controls to determine the sequence in which the control appears on the Output portion of the page. Enter numbers such as 10, 20, 30 and so forth. This allows you to later position a control between existing controls.

- 5** From the drop-down list, select the control type.

- **ADDTOCART**

Adds an item to a shopping cart, quote, or order.

- **CONFIGURE**

Passes the product to the Siebel server-based eConfigurator.

- **DETAIL**

Displays a product detail view or HTML page. Select the table and column that contains the string to load the product or HTML page.

- **LINK**

Displays a link to a pageset or URL.

- OPT_SUBCONFIG_LINK
Creates a BuildTarget call with an OPT_SUBCONFIG_LINK parameter defined.
 - PICT
Displays an image.
 - PRICE
Displays the price of a selected product for the current user.
 - SUBONFIG_LINK
Creates a BuildTarget call with a SUBCONFIG_LINK parameter defined.
 - TEXT
Displays a text string.
- 6 Enter a label for the control.
For example, to display the user-selected color, enter Color.
 - 7 Click Table to open a picklist and select the table whose values will populate the control.
 - 8 Click Column to open a picklist and select the column, from the table you selected in [Step 7](#), whose values will populate the control.
For example, if you are creating a LINK UI control that will display a URL, choose the table and column containing the URL values.
-
- NOTE:** The Output UI can be populated with both Feature and Configuration table columns.
-
- 9 For textbox controls, you can enter a table and column combination or a name.
Because text boxes do not map to Feature or Configuration tables, a name is required for Output UI (BuildTarget) calls.
 - 10 If applicable, enter a height and width for the control.

- 11** If you are creating a Link control, specify the type of link by selecting or deselecting Pageset.

Select Pageset if the Link control links to a pageset. Deselect Pageset if the Link control links to a URL.
- 12** For Anchor Text, enter the text that will appear as a link. For example: Click here to see this product.
- 13** Click Save.
- 14** Click Generate to add the new UI controls to the Output UI display page.

NOTE: It is recommended that you build your UI using the Input UI and Output UI tabs in Interactive Designer. Once the functionality in your application is set, it is recommended that you customize your UI in a third party HTML editor. If you do use a third party HTML editor, any custom code you write between the BEGIN GENERATE HTML and END GENERATE HTML comments are deleted in the pg|pageset_1.htm file. Be sure to write any custom code outside of this area.

Copying Output UI Controls

Use the following procedure to copy an Output UI control.

To copy an Output UI control

- 1** Select an Output UI control.
- 2** From the applet menu, choose Copy Record.

A New Output UI control screen appears, displaying the data for the Output UI control you are copying.

- 3** Enter a new number for the layout sequence.
- 4** Click Save.

The new record appears in the Output UI control list.

- 5** Click Generate.

Deleting Output UI Controls

Use the following procedure to delete an output UI control.

To delete an output UI control

- 1** Select the pageset.
- 2** In the Output UI tab, select the output UI control you want to delete.
- 3** From the applet menu, choose Delete Record.
- 4** Click Generate.

Editing Output UI Controls

Use the following procedure to edit an output UI control.

To edit an output UI control

- 1** Select the pageset.
- 2** In the Output UI tab, select the output UI control you want to edit.
- 3** Make edits directly to fields in the row.
- 4** Click Save.
- 5** Click Generate.

Generating Your Output UI Display Page

To update your changes to the Output UI layout, you need to generate the UI files. Previewing or deploying your application will not automatically update UI changes.

To generate your UI display pages

- 1** From the navigation drop-down list at the top of the screen, select My Pagesets.
- 2** Select the pageset you want to generate your Input or Output UI for and click either the Input UI or Output UI tab.
- 3** Create new UI controls.
- 4** Click Generate to add new UI controls to the Output UI display page.

NOTE: It is recommended that you build your UI using the Input UI and Output UI tabs in Interactive Designer. Once the functionality in your application is set, it is recommended that you customize your UI in a third party HTML editor. If you do use a third party HTML editor, any custom code you write between the BEGIN GENERATE HTML and END GENERATE HTML comments is deleted in the pg|pageset_1.htm file. Be sure to write any custom code outside of this area.

Working with the Contents List **10**

This chapter describes how to work with the contents list. It describes the structure of the contents list and also describes basic functions, such as creating and modifying your contents list.

About the Contents List

The contents list links to specific pagesets in your eAdvisor application that you can configure from within the application. The two types of contents lists are called the *Personalize Contents List* and the *Standard Contents List*.

■ Personalize Content List

Clicking Get Advice displays the Personalize Content List, which shows all of the available eAdvisor applications in a preconfigured system. Administrators can personalize the list of eAdvisor applications that specific users and user groups see when they click Get Advice by writing personalization rules and tagging specific pagesets. For more information, see *Siebel Personalization Administration Guide*. For more information on display of pagesets, see [“About Managing the Display of Pagesets” on page 89](#).

■ Standard Contents List

You can manage the display of the contents list in the Application UI Definition file or a Pageset UI Definition file. For more information, see [“Customizing Your UI” on page 165](#). You can use the contents list table to define the hierarchy and links of the contents list entries in the application. To define other HTML properties, such as background color or the images used for bullets, use the cascading style sheet, located at `\pg\onlink.css`, that defines the look and feel of the contents list.

You can use the default project (DefaultProject) and contents list (prodlistdata) as a high-level index from which your application references a number of projects and pagesets. For example, you may use the default project to contain only the default contents list, which acts as a root list that links to other projects and contents lists.

The default project and contents list files are located in the `ISSRUN\CDAPROJECTS\ENU` directory on the server on which you installed Interactive Designer. Whether or not you use them, do not delete the default project and its contents list.

You can manage the display of the contents list in the Application UI Definition file or a Pageset UI Definition file. For more information, see [“Customizing Your UI” on page 165](#). You can use the contents list table to define the hierarchy and links of the contents list entries in the application. To define other HTML properties, such as background color or the images used for bullets, use the cascading style sheet, located at `\pg\onlink.css`, that defines the look and feel of the contents list.

You can also control which pagesets from the content list display when the user clicks Get Advice. To do this, see [“About Managing the Display of Pagesets” on page 89](#).

You can use the default project (DefaultProject) and contents list (prodlistdata) as a high-level index from which your application references a number of projects and pagesets. For example, you may use the default project to contain only the default contents list, which acts as a root list that links to other projects and contents lists. The default project and contents list files are located in the `ISSRUN\CDAPROJECTS\ENU` directory on the server on which you installed Interactive Designer. Whether or not you use them, do not delete the default project and its contents list.

Customizing Get Advice Button Functionality

Use the following procedure to reference a project and contents list other than Default Project and its contents list when an application is called from a Get Advice button.

To customize the Get Advice button

- 1 Edit the button in Siebel Tools.
- 2 Change the Project value to the name of the project you want to use.
- 3 Delete the Pageset and Dyndefs values.

For more information, refer to *Siebel Tools Reference*.

About Populating the Default Contents List

When you create a new project or pagesets you need to:

- Update the contents list of the project to contain the pagesets in that project.
- Deploy the project.
- Add a link from the Default Project's contents list to either the new project or new pagesets in the project.
- Deploy the modified DefaultProject.

Viewing the Contents Lists

Use the following procedure to view the contents list.

To view a contents list

- 1 From the Show drop-down list, select My Projects or All Projects.
- 2 In the Projects screen, click the Contents Lists tab.
- 3 From the list, select the contents list you want to view.

Creating a Contents List

You can use the default contents list, `prodlistdata`, or create a new contents list.

To create a contents list

- 1** On the Contents Lists tab of the Projects screen, choose New Record from the applet menu.

A new Contents List form appears.

- 2** Enter a name, contents list ID, and any notes about the contents list.

The contents list ID is the ID used in the code to reference the contents list.

- 3** Click Save.

- 4** Access the new contents list file by using the `ShowContentsList` function.

See *Siebel Interactive Designer API Reference* for more information.

Creating Contents List Items

Use the following procedure to create contents list items.

To enter contents list items

- 1** Select the contents list and, in the Editor tab, click New.

A new row appears.

- 2** Enter the sequence in which the item will appear in the contents list.

- 3** Enter a level for the contents list item.

The level determines the location of the item in the contents list hierarchy. Number 1 represents the outermost level, the number 2 a level indented below that, and so on. Use the number 0 to create a blank entry (vertical space) between items.

- 4** Enter a label for the contents list item.

The label defines the text of the item in the contents list of the browser-based application. You can include standard HTML in the Label cell to format the text of the item.

- 5** In the Link To field, enter the pageset to open when a user clicks the contents list item.

Use the syntax Project|Pageset ID. If the pageset belongs to the current project, use the syntax Pageset ID. You can also enter a URL to link to a URL.

- 6** Enter the image location.

Use image location only for level 1 entries in a contents list table. Image location points to an image that appears in the Image Location area frame when the mouse pointer hovers over the entry in the contents list. Use a path relative to the pg directory when entering the image file name.

- 7** Click Save.

Copying a Contents List

To use the same data in an existing contents list in a new contents list, you can copy a contents list including all the contents list data records.

To copy a contents list

- 1 Select a contents list.
- 2 From the applet menu, choose Copy Record.

A New Contents List screen appears, displaying the data for the contents list you are copying.

- 3 Enter a new contents list name and ID.

These values must be unique throughout the project.

- 4 Click Save.

The new record appears in the Contents List list.

To copy a contents list item, use the following procedure.

To copy a contents list item

- 1 Select a contents list record.
- 2 From the applet menu, choose Copy Record.
- 3 A New Contents List Record screen appears, displaying the data for the record you are copying.
- 4 Click Save.

The new record appears in the Contents List Items list.

Deleting a Contents List

Use the following procedure to delete a contents list and contents list items.

To delete a contents list

- 1** Select a project.
- 2** On the Contents List tab, select a contents list.
- 3** From the applet menu, choose Delete Record.

A dialog box appears confirming that you want to delete the record.

- 4** Click OK.

To delete a contents list item

- 1** Select a project.
- 2** On the Contents List tab, select a contents list.
- 3** In the Editor applet, select the contents list item you want to remove.
- 4** From the applet menu, choose Delete Record.

A dialog box appears confirming that you want to delete the record.

- 5** Click OK.

Editing a Contents List

Use the following procedure to edit a contents list or contents list item.

To edit a contents list

- 1** Select a project.
- 2** On the Contents List tab, select a contents list and click Edit.

An Edit Contents List applet appears.

- 3** Edit the fields and click Save.

To edit a contents list item

- 1** Select a project.
- 2** On the Contents List tab, select a contents list.
- 3** In the Editor applet, select the contents list item you want to edit and make changes in the row.
- 4** Click Save.

Customizing Your UI **11**

This chapter describes how to customize the UI for your display pages by directly editing the Interactive Designer HTML files. It describes methods for modifying the frames, contents list, and UI controls used in your display pages.

About Customizing Your UI

Use a text or HTML editor to add Interactive Designer functions that customize application behavior and appearance. You can directly modify the project UI files using HTML, DHTML, and any other code acceptable for modifying HTML pages.

It is recommended that you finalize your data model before making changes to the UI files so that you do not need to regenerate the files after adding custom code. When you click Generate, the code between the BEGIN GENERATE HTML and END GENERATE HTML comments are regenerated in the UI file. Be sure to write any custom code outside of this area if you still plan to use Interactive Designer input and output UI controls after the customization.

For more information about the Interactive Designer functions, refer to *Siebel Interactive Designer API Reference*.

Editing the Project UI Files

To open, edit, and save your Project UI files, use the following procedure.

NOTE: Do not modify files in the CS directory.

To edit a project UI file

- 1** In the Project Files tab of the Project screen, select the row that has the file you want to edit. Click on the file name hyperlink to open up the file in a new browser window.
- 2** From the Internet browser File menu, choose Save As and save the file to your hard drive.
- 3** Edit the file you saved on the hard drive in [Step 2](#) using an HTML editor.
- 4** If you plan to continue using Interactive Designer input and output UI controls, add your code outside of the section enclosed by the comments BEGIN GENERATE HTML and END GENERATE HTML.
- 5** Save the file.
- 6** In the Siebel Application, highlight the file you want to modify and click the Edit button.
- 7** In the Edit form applet, use the pick icon in the Attachment Name field to select the file on your hard drive that you modified in [Step 3](#) through [Step 5](#).
- 8** Click Save in the Edit form applet.

About the Project UI Files

Each application contains a single Application UI Definition file, `\ui\ol_ui.htm`, that determines the structure and appearance of the outermost layer of an application. The HTML frameset layout defined in the Application UI Definition file represents what appears when the application first loads. It also represents the static areas, such as the contents list, surrounding the display pages as a user continues to interact with the application.

Each FRAME tag in this file defines an area in the application. One of the frames defined in the Application UI Definition file outlines the area, called `mainArea`, in the Interactive Designer templates, into which all the display pages in your application load. You must use the `RegisterUI` function to indicate which frame defines this pageset display page area. The nested frameset structure of this single area is managed by a Pageset UI Definition file.

By default, Interactive Designer associates the default Pageset UI Definition file, `\pg\oc_default_ui.htm`, with all the pagesets in the Interactive Designer project. In this case, all display pages in your application appear inside the same frameset, regardless of which pageset they are part of.

If you want to change the frameset structure of the display page area based on which pageset the display pages belong to, you can create additional Pageset UI Definition files (`\pg*_ui.htm`) that define unique framesets. To associate a Pageset UI Definition file with a particular pageset, use the Interactive Designer application function `RegisterFrameSet` inside the Pageset UI Registry file, `\pg\pagesetID_i.htm`, of that pageset.

The Pageset UI Registry file also uses the `RegisterPageLocation` and `RegisterExceptionFrames` functions. These functions define which display pages and exception messages appear in each of the frames defined in the Pageset UI Definition file of the pageset. The Input UI and Output UI Generate functions always produce `pageset_1.htm` and `pageset_2.htm` files regardless of the `pageset_i.htm` settings.

As an application designer, you can put input UI controls, such as the Exterior Color list box, and output targets, such as the red car graphic, on a single display page. This requires you to write custom DHTML code to minimize “flashing” on the display page every time a user makes a selection and the page reloads. The default UI layout for an application therefore separates the two kinds of items across two different pages, and appear in separate frames.

Modifying the Application UI Definition File

Use the following procedure to modify the application UI definition file.

To modify the Application UI Definition file

- 1** In the Project Files tab, navigate to the UI directory and select ol_ui.htm.
- 2** Edit and save the file.

For more information, see [“Editing the Project UI Files” on page 167](#).

About Modifying the Contents List Location

You can manage the display of the contents list in the Application UI Definition file or a Pageset UI Definition file.

In the Application Definition File

If you use the Application UI Definition file to define where the contents list appears, the contents list display becomes part of the application UI. It appears in a static frame throughout the application, regardless of which pageset is active (see [Figure 22](#)). Use the RegisterContentsListFrame function to indicate which application frame the contents list loads into.

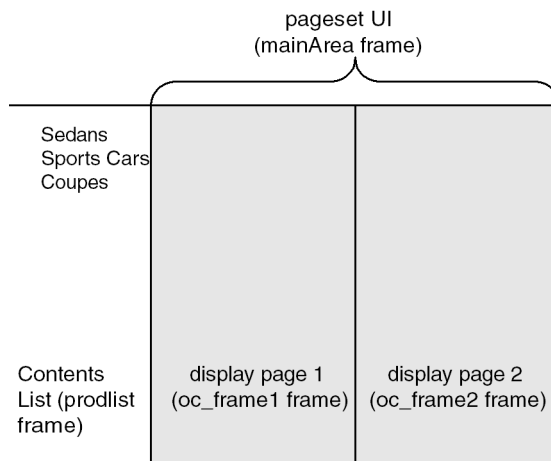


Figure 22. Sample Layout in Which the Contents List is Defined in the Application UI Definition File

In a Pageset UI Definition File

If you use a Pageset UI Definition file to define where the contents list appears, the contents list display becomes part of the UI definition of an individual pageset. Whether it appears depends on which pageset is active. Use the `SetContentsListFrame` function inside a Pageset UI Definition file to have the contents list appear and disappear based on which pageset your users are interacting with (see [Figure 23](#)).

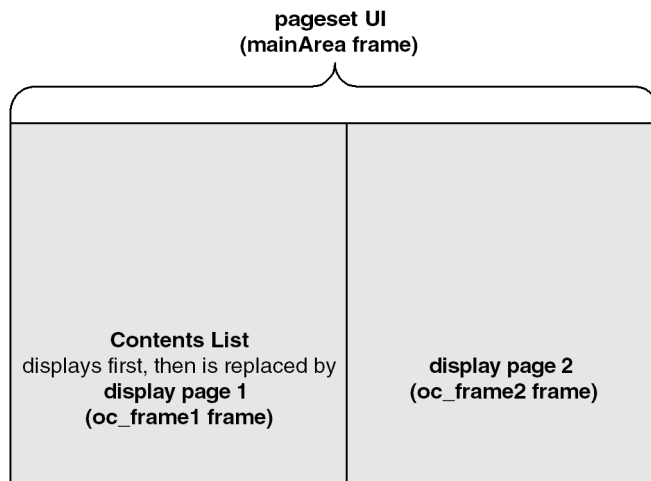


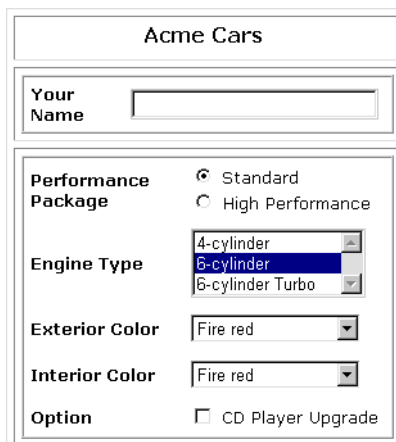
Figure 23. Sample Layout in Which the Contents List is Defined in the Pageset UI Definition File

About the UI Controls

Interactive Designer application functions, such as BuildWidget and BuildTarget, add Input UI controls and Output targets to show configuration choices and results on display pages. The LoadPageset function links between display pages in different pagesets.

Input UI Display Example

Figure 24 shows an example of an input UI display page. The input UI controls that appear on this page are created using the browser-based application function named BuildWidget and, depending on the type of control, a Feature table.



Acme Cars

Your Name

Performance Package Standard High Performance

Engine Type

Exterior Color

Interior Color

Option CD Player Upgrade

Figure 24. Input UI Display Page

The function call used to create the Engine Type list box in Figure 24 looks like this:

```
<script>document.write(ISS.BuildWidget("LISTBOX",window,"ENGINE",3,32,true));</script>
```

The function call indicates that the control type to create is a list box, LISTBOX, that will display the description values, DESC, in the Feature table, ENGINE.

Output UI Controls Example

Figure 25 shows the display page with output UI controls that result from a valid configuration.

Hi Steven!

This car matches your selections:



Your Current Selections:

Standard performance package
6-cylinder engine
Fire red exterior color
Fire red interior color
Stereo w/o CD player

Base Price:	\$ 15499
Upgrades:	\$ 1099
Total Price:	\$ 16598

Figure 25. Output UI Display Page

The output UI Control text and images on display pages are dynamically created using an Interactive Designer BuildTarget function. BuildTarget creates text and images based on the user's selections as well as the valid configuration information.

For example, the following code displays the text entered in the Your Name text box. When your selections match a valid configuration, this text appears in the Results page.

```
<script>document.write( ISS.BuildTarget( "TEXT", window, "CUSTNAME" ) );</script>
```

Using UI Templates

You can maintain a common look and feel across multiple projects by using UI templates. Setting up a UI template requires two steps:

- 1** Setting up the template directory.
- 2** Adding the template to a project.

The following procedures use Windows NT path syntax and assume an English language (enu) installation. The procedure uses ACME_Template_01 as the template name.

To set up the template directory

- 1** In the Siebel application installation, locate the ISSTEMPL\enu\ProjectTemplates directory.
- 2** In this directory create a subdirectory and give it the template name, for example ACME_Template_01. Then create the directory structure beneath it, as shown.

```
Acme_Template_01
  PG
    oc_template_i.tpl
    oc_template_i.tpl
    oc_template_2.tpl
  UI
```

PG and UI are subdirectories; oc_template_i.tpl is the UI registry file; oc_template_i.tpl is the UI input file and oc_template_2.tpl. is the UI output file.

- 3** Place the image and HTML files related to the template in the UI directory.

The last step is to add the template to the project.

To add the template to a project

- 1** Navigate to Interactive Designer and create a project.
You cannot add a template file to an existing project.
- 2** In the project form, enter the name of the UI template in the Template Name field.

Referencing Other Siebel Data **12**

Interactive Designer allows you to publish other Siebel data and set up interactions with other Siebel applications. This chapter outlines the methods for publishing this data and setting up interactions with Siebel applications and data.

NOTE: This chapter does not apply to stand-alone applications. Stand-alone application users should skip this chapter.

About Referencing Other Siebel Data

To reference other Siebel data, you will:

- Map columns in your Feature and Configuration tables to business components.

The mapped columns are populated with data from the selected business components when you deploy or preview your project.

- Structure your data model for customizable products.
- Add or customize links to any of the following:
 - ePricer to provide pricing information in your application
 - Server-based eConfigurator
 - The Siebel shopping cart or quote
 - Product detail view

About Binding to Siebel Business Components

Interactive Designer Feature and Configuration tables are populated with static data or referenced Siebel data. Static data is entered directly into the table, and referenced Siebel data is updated in Interactive Designer when it changes in the Siebel database. When you bind to a business component in a Feature or Configuration Table, you provide a connection back to the Siebel database. Siebel Interactive Selling products share a single data source, or product item master, stored in the Siebel database.

NOTE: After you publish a project, the published data will not change if the data changes in the Siebel database. You will need to deploy your project to refresh the data.

NOTE: Make sure that all the products referenced within a browser-based model to set up interactions with other Siebel Applications are set to the same organization that the runtime users may belong to. Note that a product can be set to multiple organizations. If products are not set up in this way, runtime users may encounter errors while making the transition from browser-based applications to other Siebel applications, such as Getting Price and Adding to Cart, and not see certain products, both in connected and in disconnected mode.

How the Binding Works

Feature and Configuration Table Designer provide three columns to use to bind to your Siebel data.

- Business Component

Use the picklist to specify a Siebel business component.

- Field Name

Use the picklist to specify a field in the Siebel business component you have chosen.

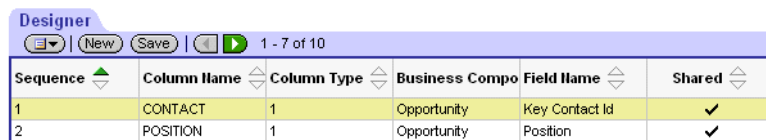
- Shared

If more than one column shares the same row of data in the business component, use this check box to specify those extra columns.

The binding works by mapping the table column to a Siebel business component. For more information, refer to *Siebel Interactive Designer API Reference*.

Configuration Table Designer Example

Figure 26 shows the Configuration Table Designer with two shared columns.




Sequence	Column Name	Column Type	Business Compo	Field Name	Shared
1	CONTACT	1	Opportunity	Key Contact Id	✓
2	POSITION	1	Opportunity	Position	✓

Figure 26. Shared Columns in a Configuration Table

The CONTACT column is mapped to the Key Contact ID field in the Siebel business component named Opportunity. The POSITION column is mapped to the POSITION field in the Siebel business component named OPPORTUNITY. The two columns, CONTACT and POSITION, share the same target, that is, they will both get their values from the same row of data in the Opportunity business component.

You will pick actual values for the mapped columns in the Table Editor. The cells in each mapped column appear with a pop-up picklist. Choose values from the business component to populate the mapped column.

In the example shown in Figure 27, A.K. Parker was chosen in the CONTACT column. The selected contact's position populated the POSITION column because the CONTACT and POSITION columns are linked with a shared target.



Row Type	Sequence	CONTACT(1)	POSITION(1)
DATA	10	A.K. Parker	Call Center Agent 2

Figure 27. Shared Columns in a Configuration Table

Adding Access to Additional Business Components

When designing a feature table or configuration table, select a business component and field to populate a column with values. Interactive Designer provides access to the following business components:

- Account
- Channel Partner
- Contact
- Internal Product
- Opportunity
- Order Entry - Orders
- Quote
- Sales Tool

If you need to access other business components, add them by performing the following steps:

- Create a new Business Component Name record.
- Create a new picklist Name record.
- Create a new Pick Applet record.

To create a new Business Component Name record

- 1** In the Application Administration screen, select List of Values from the drop-down list.
- 2** Click New.
- 3** From the Type drop-down list, select ISSCDA_DESIGNER_BC_TYPE.
- 4** In the Display Value text box, enter the name for the new business component, for example, Project.
- 5** In the Language Independent Code text box, enter the same business component name you entered in [Step 4](#), for example, Project.

- 6** Click Language Name to open a picklist and select a language.
- 7** In the Order text box, enter a number to indicate where the new business component will appear in the list.

For example, if you enter 2, the new business component will appear second in the list of business components.
- 8** Check the Active check box.
- 9** Click Save.

To create a new Picklist Name record

- 1** In the List of Values screen, click New.
- 2** From the Type drop-down list, select ISSCDA_DESIGNER_PICKLIST_TYPE.
- 3** In the Display Value text box, enter the name for the new picklist, for example, PickList Project.
- 4** In the Language Independent Code text box, enter the same Picklist name you entered in [Step 3](#).
- 5** Click Language Name to open a picklist and select a language.
- 6** Check the Active check box.
- 7** Click Save.

To create a new Pick Applet Name record

- 1** In the List of Values screen, click New.
- 2** From the Type drop-down list, select ISSCDA_DESIGNER_PICKAPP_TYPE.
- 3** In the Display Value text box, enter the name for the new pick applet, for example, Project Pick Applet.
- 4** In the Language Independent Code text box, enter the Pick Applet name you entered in [Step 3](#).
- 5** Click Language Name to open a picklist and select English-American.
- 6** Check the Active check box.
- 7** Click Save.

About Modeling for Customizable Products

In order to model your application so that it can create a customizable product structure and pass it to the server-based shopping cart or quote, eConfigurator, or ePricer, you will need to:

- Structure the data model to match the product structure in Product Administration.
- Invoke the appropriate API from the application at runtime.

After you complete these steps, your application can create the customizable product structure and pass it to the server when the following occurs:

- A user adds the customizable product to the Quote or Shopping Cart.
- A user wants to further customize the product using the server-based eConfigurator.
- The user receives a personalized price based on selections made in the application.

Before structuring your data model to integrate customizable products with your application, you should understand how data is evaluated in your data model at runtime.

For more information on customizable products, refer to *Product Administration Guide*.

Data Evaluation in Feature Tables

Each set of possible selections is contained within a Feature table. During runtime, the active data is typically determined by the row in the Feature table that corresponds to the user's selections in the UI. For example, for the exterior color of a car, the available options might be Beige, White, Black, Green, Blue, and Red. The modeler would create a Feature table that contained these colors and map an Input UI Control to it. In this Feature table, the modeler would create six rows, one for each of the colors. At runtime, if the user selects the color Black, the row corresponding to the color Black is active. At runtime, each Feature table has exactly one active row.

This row is made active by one of the following:

- User selection
- Default setting
- The Configuration table setting it to be active (see [“Best Practices” on page 189](#) for more information)

Data Evaluation in Configuration Tables

Each Configuration table has exactly one active row. This row is determined by the active rows for each of the Feature tables and by the columns in the Configuration tables that map to these Feature tables.

Evaluation of the Customizable Product Structure

Because at any time the state of the application is determined by the active row in each Feature table and Configuration table, the data from these active rows is used to create the customizable product structure. When a `GetPrice`, `AddtoSSCart`, or `GotoSSConfigurator` API is invoked at runtime, the application creates a structure from the active rows and passes it to the server for executing the request.

The application can create the customizable product structure in two ways:

- Automatic Creation
- Creation in string form

Use automatic creation for one or two-layer customizable product structures to extract attributes for the root product. For more complex product structures, use the set of supporting functions documented for `ISS.BuldProductStr` in the Siebel-Specific Functions chapter in *Siebel Interactive Designer API Reference*.

Automatic Creation of the Customizable Product Structure

When the customizable product structure is automatically created at runtime from Configuration and Feature table data, it is determined by the following factors:

- The first product found in the active row of a Configuration table is treated as the root product.

If more than one product is found in the row, the first product encountered scanning from left to right is assumed to be the root product. Subsequent products are treated as children of the root product. However, because a relationship cannot be added for automatic extraction, this method is not recommended.

- Any attribute found in the active Configuration table row is treated as the value of the attribute for the root product. You can add an unlimited number of attributes by creating a new column for each attribute.
- Any product found in the active row of a Feature table is treated as a child product of the root product. If more than one product is found in the row, the first product encountered scanning from left to right is considered the child and all others are ignored.
- Any attribute found in the active Feature table row is treated as the value of the attribute for the root product, unless a product is found in the same table. If a product is found in the Feature table, the attribute is treated as an attribute of that product and not of the root product.
- The name found for the Relationship (in the Relationship column) in the active Feature table row is treated as the Relationship within the root product to which the child belongs.

To model in Interactive Designer to connect to Customizable Products, use the following steps:

- Map root products in the Configuration table.
- Map root product attributes in the Configuration table.
- Map root product attributes in Feature tables.
- Map child products in Feature tables.
- Map attributes of the child product in Feature tables.

The following section provides the procedures for completing each of these steps. These procedures refer to an example where a Main Configuration table is modeled to add a minivan with a green exterior, white interior, and convenience package. This example is shown in [Figure 28](#).



The screenshot shows the Siebel Configuration Tables interface. The top navigation bar includes tabs for More Info, Feature Tables, Configuration Tables, Input UI, Output UI, Pageset Files, Validation Results, and Pageset Search. Below the navigation bar, there is a 'Show:' dropdown set to 'Editor' and a 'Query' button. The main table has columns for Name, Notes, Last Updated, and Last Updated By. A single row is visible with the name 'MAIN', last updated on 07/31/2001 at 5:26:33 PM by PMOORE. Below the main table is an 'Editor' window with a 'Save' button and a '1 - 2 of 2' indicator. The Editor window contains a table with columns for Row Type, Sequence, CHOOSE_MINIVAN RULE(#), ROOT(#), EXTERIOR_COLOR, INTERIOR_COLOR, and OPTIONS(#). A single row is visible with Row Type 'DATA', Sequence '10', CHOOSE_MINIVAN RULE('#) 'Y', ROOT('#) 'Horizon Minivan', EXTERIOR_COLOR 'Green', INTERIOR_COLOR 'White', and OPTIONS('#) 'CCN'.

Name	Notes	Last Updated	Last Updated By
MAIN		07/31/2001 5:26:33	PMOORE

Row Type	Sequence	CHOOSE_MINIVAN RULE(#)	ROOT(#)	EXTERIOR_COLOR	INTERIOR_COLOR	OPTIONS(#)
DATA	10	Y	Horizon Minivan	Green	White	CCN

Figure 28. Modeling Example

Mapping Root Products in the Configuration Table

Use the following procedure to map root products in the Configuration table.

To map root products in the Configuration table

- 1** Select a Configuration table to map the root product and its attributes.
Depending on your modeling needs, you may create a Configuration table specifically for the purpose of mapping products.
- 2** Switch to the Designer view for the Configuration table.
- 3** Create a type (0) column and give it any name other than a reserved name.
- 4** Map the column to the business component “Internal Product” by selecting it from the picklist.

- 5 Select a field from the list of fields in the Internal Product business component.

For product structure creation, it does not matter what field you select. However, the value for the field you select is published in the data model and is available for the modeler to display in the UI at runtime like any other data in any other cell.

The following figure shows an example of a row where the column name ROOT is mapped to the business component Internal Product and the field name Name.

NOTE: ROOT is used as an example. It is not a reserved word.

Sequence	Column Name	Column Type	Business Compo	Field Name
1	CHOOSE_MINIVAN	1		
3	RULE	0		
11	ROOT	0	Internal Product	Name

- 6 Switch to the Editor view.

A Picklist icon appears in the column you created and mapped in [Step 5](#).

- 7 Click the Picklist icon to open a list of Internal Product business components.
- 8 Select the product you want to select as the root product for each row of the Configuration table.

The product you select for a row is treated as the root product whenever that row of the Configuration table is active. In the following example, Horizon Minivan was chosen as the root product.

Row Type	Sequence	CHOOSE_MINIVAN	RULE(*)	ROOT(*)
DATA	10	Y		Horizon Minivan

- 9 You can select the same product for more than one row. However, you must do so by picking it from the picklist each time.

Mapping Root Product Attributes in the Configuration Table

Use the following procedure to map root products attributes in the Configuration table.

To map root product attributes in the Configuration table

- 1 Switch to the Designer view for the Configuration table.
- 2 Create a type (0) column and give it any name other than a reserved name.
- 3 Map the column to a class by picking a class from the Class picklist.

Map the column to a specific attribute of the class by picking an attribute from the Attribute picklist. The following example shows a column named EXTERIOR_COLOR mapped to the class Autos and the attribute Exterior Color and a column named INTERIOR_COLOR mapped to the class Autos and the attribute Interior Color.

Sequence	Column Name	Column Type	Business Compo	Field Name	Shared	Class	Attribute
1	CHOOSE_MINIVAN	1					
3	RULE	0					
11	ROOT	0	Internal Product	None			
21	EXTERIOR_COLOR	0				Autos	Exterior Color
31	INTERIOR_COLOR	0				Autos	Interior Color

- 4 Switch to the Editor view.
- 5 Enter the exact value for the attribute you want the root product to have when the Configuration table row is active.

The following example shows the value Green selected for the attribute EXTERIOR_COLOR and the value White selected for the attribute INTERIOR_COLOR.

Row Type	Sequence	CHOOSE_MINIVAN RULE(0)	ROOT(0)	EXTERIOR_COLOR	INTERIOR_COLOR
DATA	10	Y	Horizon Minivan	Green	White

- 6 Repeat [Step 5](#) to map all rows of the Configuration table.

- 7 Repeat this procedure to map additional attributes and their values to the root product.

Mapping Root Product Attributes in Feature Tables

Use the following procedure to map root product attributes in Feature tables.

To map root product attributes in Feature tables

- 1 Select the Feature table that you want to map the attribute to.
- 2 Switch to the Designer view.
- 3 Create a Standard column and give it any name other than a reserved name.
- 4 Map the column to a class by selecting a class from the Class picklist.
- 5 Map the column to a specific attribute of the class by selecting it from the Attribute picklist.
- 6 Switch to the Editor view.
- 7 Enter the exact value for the attribute you want the root product to have when the Feature table row is active.
- 8 Repeat [Step 7](#) for each row of the Feature table to map all the values of the attributes, each of which becomes the value selected when its row is active.

Mapping Child Products in Feature Tables

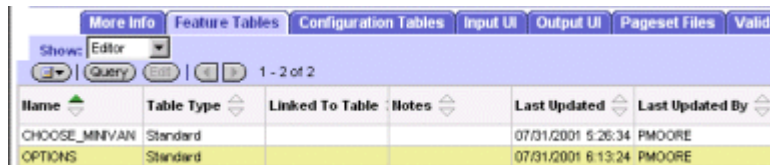
Note that you must select a product for the root, but you can choose not to select a product for a child. You may choose not to add a child by:

- Not mapping a column for it, in which case the user will never get a child.
- Entering the value null for a mapped row.

To map child products in Feature tables

- 1 Select the Feature table you want to map to the child product.

This procedure uses the following feature table as an example.

A screenshot of the Siebel Feature Tables table. The table has columns: Name, Table Type, Linked To Table, Notes, Last Updated, and Last Updated By. The rows are: CHOOSE_MINIVAN (Standard, 07/31/2001 5:26:34, PMOORE) and OPTIONS (Standard, 07/31/2001 6:13:24, PMOORE). The table is displayed in a web browser interface with tabs for More Info, Feature Tables, Configuration Tables, Input UI, Output UI, Pageset Files, and Validation. The current view is 'Editor' and shows 1 - 2 of 2 records.

Name	Table Type	Linked To Table	Notes	Last Updated	Last Updated By
CHOOSE_MINIVAN	Standard			07/31/2001 5:26:34	PMOORE
OPTIONS	Standard			07/31/2001 6:13:24	PMOORE

- 2 Switch to the Designer view.
- 3 Create a column and give it any name other than a reserved name.
- 4 Map the column to the Internal Product business component by selecting it from the picklist.
- 5 Select a field from the list of fields in the Internal Product business component.
For product structure creation, you can select any field. However, the value for the field you select is published in the data model and is available for the modeler to display in the UI at runtime like any other data in any other cell.
- 6 Create another column and give it the reserved name RELATIONSHIP and do not map it to anything. RELATIONSHIP is a reserved word.
- 7 Switch to the Editor view.
- 8 In the column you mapped to a business component, click the Picklist icon to select the product you want to select as the child product for each row of the Feature table.

- 9 In the Relationship column, enter the exact name of the Relationship of the root product that this child should belong to within the customizable product structure.

If you want a row not to be mapped to a product, do not choose an item in the picklist, or delete the entry in the row. The relationship name does not need to be set in this case.

The following figure shows an example where Option Packages is the relationship of the root product.

Row Type	Sequence	CODE	DESC	RELATIONSHIP
DATA	10	CCN		Option Packages

Mapping Attributes of the Child Product in Feature Tables

To map attributes of the child product in Feature tables, complete the following two procedures:

- [“Mapping Child Products in Feature Tables” on page 187.](#)
- [“Mapping Root Product Attributes in Feature Tables” on page 187.](#)

Best Practices

Before you start modeling in Interactive Designer to connect to customizable products, consider the following:

- You should know the structure of the customizable products, including the exact names of the Relationships within the customizable product, and the names and values for the attributes you want to use.
- If you want to connect to a customizable product that has more than one level and pass off parameters to multiple levels, you will need to use one of the following methods:
 - Subconfiguration, where the root of the child pageset is the child of the parent root product.

- Create a string representation of the product yourself to pass off parameters. For more information on creating a product string by hand, refer to `ISS.BuldProductStr` in *Siebel Interactive Designer API Reference*.
- You can pass incomplete information about a customizable product, but cannot pass incorrect information. For example, if there are five Relationships in a customizable product, but you want to connect to the customizable product by passing just one Relationship and one child within the selected Relationship. The connection would be successful as long as you have mapped the Root, Relationship Name, and Child Product correctly. However, if you pass an incorrect Relationship Name, Child, or Attribute Value, an error message will report this conflict. However, if you pass incomplete information, the resulting product may not be consistent with the rules defined in the server-based eConfigurator and incomplete products will be added when you use Add To Cart.
- If there is no one-to-one correspondence between user selections and attribute values or child product selections (which is normally the case), you can still model by creating Feature tables that are not mapped to input UI controls. To model without mapping to an Input UI controls, create TYPE (0) columns in the Configuration table to make appropriate rows active within the Feature tables. The active Feature table rows will be based on a combination of user selections in other user selectable Feature tables.

Runtime Interaction with Your Shopping Cart or Quote

By default, a link to the Siebel shopping cart or quote is included on the default UI Output page of your application. You can also add more links to the Siebel shopping cart or quote in your application.

If you are using your application outside of the Siebel framework (in stand-alone mode), refer to *Siebel Interactive Selling Suite Transact Server Interface Reference* for information on connecting to your shopping cart.

Figure 29 shows an application that includes an Add to Cart button.



Figure 29. Application That Links to a Quote

In the example shown in [Figure 30](#), when a user clicks the Add to Cart button, the sedan is added to the user's quote.

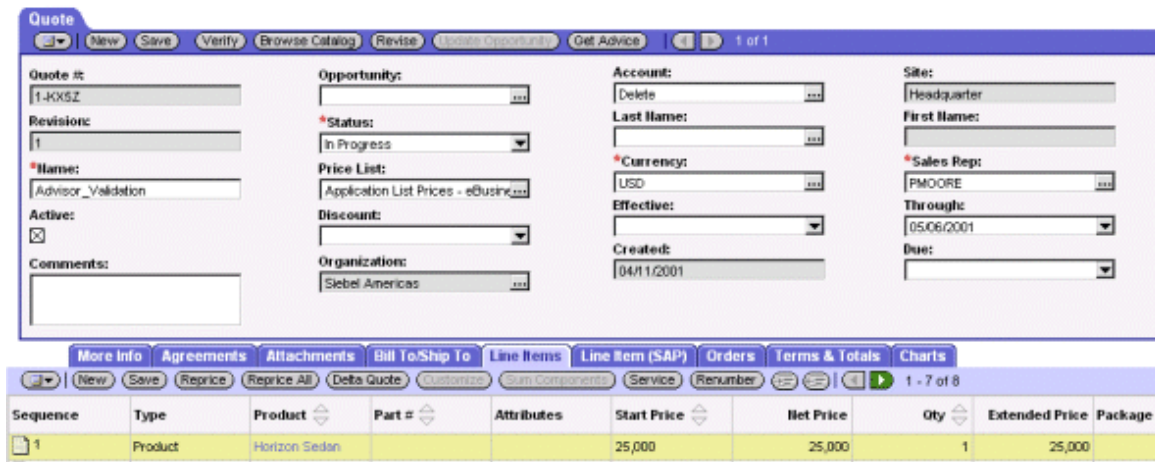


Figure 30. New Quote Line Item for Product Selected in an Application

To integrate your application with the Siebel shopping cart or quote

- 1 Add a button UI control labeled Add to Cart or Add to Quote to the output UI page for the pageset.
- 2 Open the UI file and add code to call AddToSSCart from an OnClick event for the new button.
 - AddToSSCart(optional productDescriptionString)

AddToSSCart adds the current product to an order or quote. If the string parameter is included, it is used as the description for the customizable product to be added. If it is not included, the model and state are examined to build a customizable product description. After the product is added to the order or quote, the browser-based view is replaced by the order or quote view.

For more information, refer to *Siebel Interactive Designer API Reference*.

NOTE: To add any items to the cart or quote from applications, the recommendation list or product must exist in the Siebel product master. The product master runs off the Siebel Internal Product business component.

Runtime Interaction with the Server-Based eConfigurator

You can integrate your browser-based applications with the server-based eConfigurator. For example, you may want to use an eAdvisor application to gather data from users and present a solution, such as a recommended product, but for further customization provide a link to the server-based deployment of eConfigurator. For information on the difference between browser-based and server-based applications, see “[Siebel eConfigurator Deployment Methods](#)” on page 23.

Figure 31 shows a eAdvisor application that links to the server-based eConfigurator.

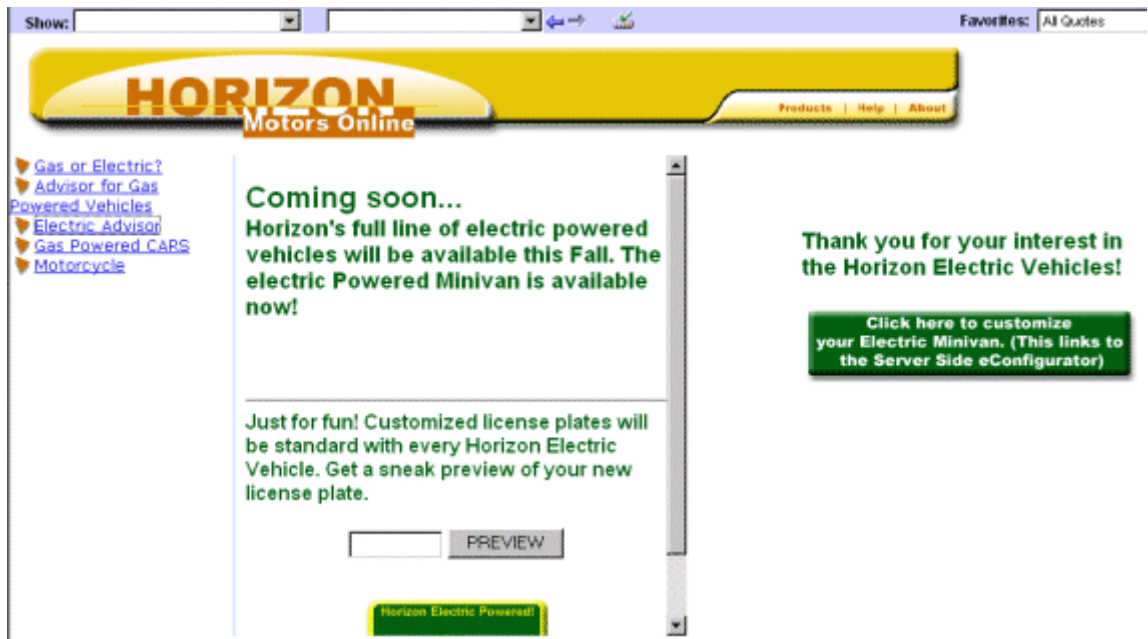


Figure 31. An eAdvisor Application That Links to Server-Based eConfigurator

In this example, when a user clicks the Configuration button shown in [Figure 32](#), the server-based eConfigurator opens to allow the user to further customize their minivan.

Horizon Minivan

Description:
Total Price:
Messages:

Exterior Color
Green

Interior Color
White

Model
Basic Model XE

New/Used
New

Transmission

Figure 32. An eConfigurator Session Started from an eAdvisor Application

For more information about the functions referred to in the following procedure, refer to *Siebel Interactive Designer API Reference*.

To integrate with the server-based eConfigurator

- 1 Model your Feature and Configuration table data to integrate with customizable products if you are using them.

For more information, see [“About Modeling for Customizable Products” on page 181](#).

- 2 Add a link, such as “Configure This Product,” to your UI.
- 3 Open the Pageset UI file and add an OnClick event to call one of the following functions:

- GotoSSConfigurator(optional productDescriptionString)

GotoSSConfigurator hands off a product to the Siebel server-based eConfigurator, replacing the browser-based view in the browser with the server-based eConfigurator view. If a string is specified, it is used to describe the customizable product, which is used to start up the eConfigurator. If a string is not displayed, the current model and state are used to build a customizable product.

If you are building a product description string, you will need to build customizable product strings in the data model using cell functions. Helper functions are provided to help create a string in the correct format. For more information, refer to `ISS.BuldProductStr` in the Siebel-Specific Functions chapter of *Siebel Interactive Designer API Reference*.

- GetPrice(optional product)

GetPrice shows the final price of the selected product for the current user in a popup window. This price is based on the pricing information you have set up in Siebel ePricer.

- ShowProductDetails(product)

ShowProductDetails opens the detail view for the currently selected product ID using the parameters defined in the Siebel configuration file to determine which Siebel view to open. ShowProductDetails can be called from any frame within the application and can be executed anywhere a JavaScript function can be used. For more information, refer to ShowProductDetails in the Siebel-Specific Functions chapter of *Siebel Interactive Designer API Reference*.

- GotoSSView(viewName)

GotoSSView switches the current browser-based view to the specified view name.

For other interactions, use SendSelectionInformationToServer and write your own business service method.

4 Set the following parameters in the Siebel application CFG file to specify the product detail view the browser-based application will navigate to at runtime.

- ISSCDAProdDetBusCompName: defines the Business Component Name that a product detail view will use.
- ISSCDAProdDetBusObjName: defines the Business Object Name that a product detail view will use.
- ISSCDAProdDetViewName: defines the product detail view name.

Example

This section provides a procedure for creating an example of an up-sell from a cat with a leash to a dog in the server-based eConfigurator. In this example, the user can select a kind of cat and leash in a browser-based application. An up-sell link is provided “Get a dog instead!” When clicked, the dog is displayed in server-based eConfigurator.

For more information on the following steps, refer to *Siebel Interactive Designer API Reference*.

To create the cross-sell example

- 1** Create a Feature table named LEASH.
- 2** Enter the following data in the LEASH Feature table.

To enter data in the Desc column, map the column to one of two leashes in the Internal Product business component.

Code	Desc	Default
S	Short Leash	DEFAULT
L	Long Leash	

- 3 Create a Feature table named PERSONALITY.
- 4 Enter the following data in the PERSONALITY Feature table.

Code	Desc	Default
IND	Independent	DEFAULT
SUB	Submissive	

- 5 In the MAIN Configuration table, enter the following data.

MAP ROOT_PROD to one of two cats in the Internal Product business component.

Map UPSELL_ROOT_PROD to one of two dogs in the Internal Product business component.

LEASH(1)	PERSONALITY(1)	ROOT_PROD(0)	UPSELL_ROOT_PROD (0)
(=*)	IND	CAT-TABBY	DOG-PITBULL
(=*)	SUB	CAT-BURMESE	DOG-LAB

- 6 On the Output UI page, add the following link to the up-sell:

```
<a href="#" onclick =
"ISS.GotoSSConfigurator(ISS.GetBusCompID('UPSELL_ROOT_PROD'))";
return false;">Get a dog instead!</a>
```

- 7 Create the following prodStr:

```
ISS.AddToCart(ISS.BuildProductStr(ISS.GetBusCompID("ROOT_PROD"),
1, null, ISS.BuildChildList(ISS.BuildProductStr("LEASH.DESC", 1,
null, null, "RESTRAINT")));
```

Runtime Access to Your Pricing Information

There are two ways of displaying pricing information in browser-based applications:

- Publishing static pricing in static pagesets

Static list price based pricing information can be displayed in pagesets by publishing projects associated with price lists. Product-specific price list information is displayed in pagesets when projects are generated.

- Providing a context-specific price during runtime.

End users can access context specific pricing information at runtime by clicking the Get Price button, as shown in Figure 33.

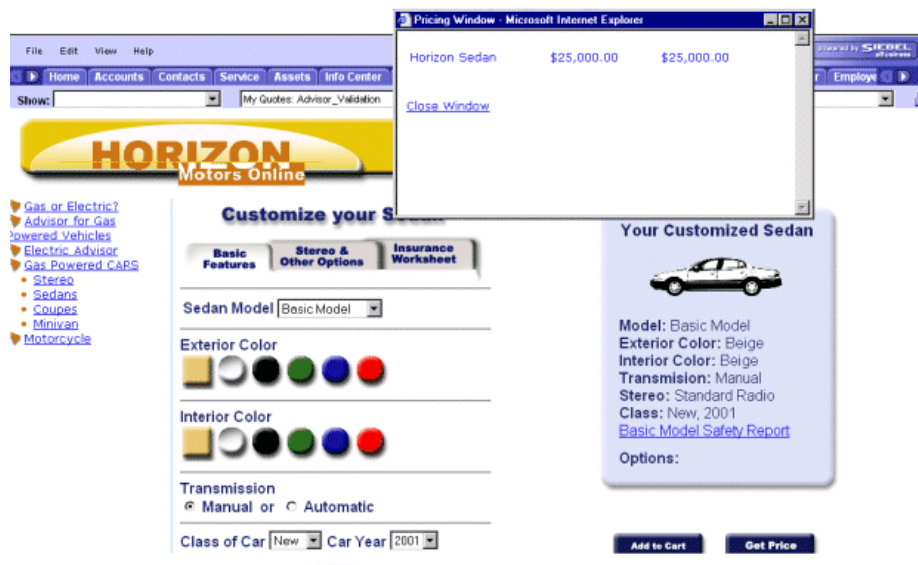


Figure 33. Browser-Based Application Displaying Price

About Publishing Pricing Information in Pagesets

To provide price values for your browser-based applications, you can associate a pricing column in your browser-based application with a Siebel price list. For situations where products and attributes are represented in the browser-based tables, but you do not want to refer to a Price List, you can enter specific values in the pricing column.

The Output UI template provides a Get Price button which, when clicked, returns the user's price. Pricing is then determined using the following calculation:

$$\text{start price} * \text{adjustment} = \text{final price}$$

Start price includes a price from the list price (list price or promotional price). The price list provides a static price for each internal product in the price list. To integrate this functionality with Interactive Designer, you will select a price list at the project level. For more information, see [“Associating a Price List with Your Browser-Based Model” on page 200](#).

Adjustment is defined in ePricer. To apply adjustments (such as promotional prices, volume discounts, or context-specific adjustments) to the static list price, you will need to use functionality provided by Siebel ePricer.

Using ePricer, you can use the adjustment variable of this equation to apply:

- Pricing rules
- Volume discounts
- Promotions
- Configurable product (bundling) rules

For example, you can apply a pricing rule to the A.K. Parker account that automatically applies a 10% discount to all purchases. If a contact from A.K. Parker logs in and orders the Model A Monitor with a list price of \$500, when he hits Get Price the following calculation is performed:

$$500 * .9 = 450$$

and a final price of \$450 is returned.

For more information, refer to *Pricing Administration Guide*.

To provide access to your pricing information

- 1** Associate your price list with the Browser-Based model in Interactive Designer.
- 2** Modify the GetPrice function as needed.

The Get Price link is included in your browser-based application's UI template by default. For more information on this step, refer to *Siebel Interactive Designer API Reference*.

- 3** Edit the app_config.js file to modify the pricing display information.
- 4** Add profile variables in Siebel Tools to further personalize price.
- 5** Edit the Siebel application .cfg file to set parameters for the runtime behavior of Get Price.

The following section provides procedures for each step.

Associating a Price List with Your Browser-Based Model

To associate a price list with the browser-based model, you need to make the association at the project level as well as the Feature or Configuration table level.

NOTE: Because you can select only one price list per project, and all users access the same price list, it is recommended that you use the default price list. To determine which price list is the default, go to Application Administration > List of Values and search for Master Price List. The Display Value field displays the price list ID of the default price list.

NOTE: Whenever the price list or prices for price list line items are updated, you will need to regenerate the pagesets that reference that price list.

To associate your price list with the browser-based model

- 1 In the Projects screen, select a project and, in the More Info tab, select the price list you want to use from the Price List drop-down list.

This price list generates data for list price and promotion price fields you create in your feature tables. You can change the associated price list at any time. For information on creating pricing lists, refer to *Pricing Administration Guide*.

NOTE: For a list of products missing from the price list, validate the project and view the error messages on the Validation Results tab.

- 2 In Feature and Configuration tables where you want to include price, open the Table designer and add a Price column.

NOTE: If you have multiple pricing options, create multiple pricing columns with different, descriptive names. For example, to emulate eSales behavior, create the columns List Price and Your Price.

- 3 From the business component column, select Internal Product.
- 4 In the Field Name column, open the Field picklist and select:
 - List Price: to always display the list price.
 - Promotional Price: to display the promotional price.
- 5 Check the Shared column to reference a row ID in a business component that other columns in the table reference. This step is optional.

All columns in the Table Designer that reference the same business component, and have the Shared field selected, will be populated with data when a value is selected for any one of them.

- 6 Open the Table editor and select products from the Pricing column fields.
- 7 From the Pagesets menu, select Validate and click the Validation Results tab.
- 8 Make sure there are no messages showing that products are missing from the price list that was used to publish the pageset.

About Modifying Display Information in `app_config.js`

To display additional data about the object being priced, edit the `APP_PRICE_DATA` variable in `app_config.js`, specifying the information you want displayed by `GetPrice`.

For example:

```
APP_PRICE_DATA = new Array('price', 'description', 'engine');
```

would display:

```
price = "$26,900"  
description = "Audi A4"  
engine = "1.8T"
```

You can also use other properties of the object, such as:

```
color = "Blue"  
interior = "Black"
```

Whatever is set in `APP_PRICE_DATA` is what will appear, and in the order listed.

About Modifying the Application Configuration File

Browser-based applications provide pricing information by resolving the relevant context-specific information to provide personalized prices. If the context-specific information exists in multiple variables during a session, certain variables have priority over others. To resolve variable values, at runtime the application executes the following algorithm:

- Session-specific variables are identified using session profile attributes that were declared in Siebel Tools.

For more information on personalization, refer to *Siebel Tools Reference*.

- Variable values available at the session level are calculated.
- The values in the current quote or order override existing session values.

For information on the quote and order variables you can set in the Siebel application CFG file, refer to the Browser-Based Application File Reference chapter in *Siebel Interactive Designer API Reference*.

In your Siebel application CFG file, you can also set the `ISSCDAGetMyPriceFields` variable to specify the field names to be returned to the `GetMyPrice` function. These fields will appear in the Description field in the browser at runtime. These fields must be included in the profile attributes for the ISSCDA Get My Price virtual business component.

For more information on the Siebel application configuration variables, refer to the Browser-Based Application File Reference chapter in *Siebel Interactive Designer API Reference*.

Adding Rules Based Pricing

At runtime, users can click Get My Price to access a personalized price. This price includes prices based on pricing rules. These rules are created in the ePricer Pricing engine administration screens. To add rules-based pricing, create a new pricing model with pricing rules. This pricing model then needs to be attached to either the default price list or the organization specific price list so that context specific prices can be provided in response to the `GetPrice` call.

To create pricing rules

- 1 In ePricer, create rules on quotes and orders.

Create rules on quotes if quotes are the object that products are added to when the Add to Cart button is clicked. Create rules on orders if orders are the integration object. The integration object (quotes or orders) is specified in the Siebel application CFG file.

In the following example, rules are created in the Quote Item business component name. These rules will then appear in the pricing window, quote, and shopping cart.

The screenshot shows the 'Pricer Factors' dialog box in Siebel Tools. The dialog has a title bar with 'Pricer Factors' and '1 of 4'. Below the title bar are 'New' and 'Save' buttons. The main area contains several fields:

- *Name:** A text box containing 'March sale'.
- Business Component Name:** A dropdown menu with 'Quote Item' selected. Below the dropdown, a list of other business components is visible: 'Price List Copy', 'Price List Item', 'Quote', 'Quote (Complex Product)', 'Quote Item', and 'Quote Item (Complex Product)'. A label 'Quote Item Business Component' with a line points to the 'Quote Item' selection.
- Attribute Name:** An empty text box.
- Operator:** A dropdown menu with 'EXISTS IN' selected.
- Mapping Name:** A dropdown menu with 'March sale' selected.
- Next Factor When False:** A dropdown menu with an empty field and a three-dot menu icon.
- Comments:** A text box containing '5% discount for any product that'.

- 2 In Siebel Tools, add new fields to quotes or orders (depending on which integration object you are using).
- 3 In Siebel Tools, specify the new profile attributes in the ISSCDA Get My Price virtual business component.

For more information on completing these steps, refer to *Pricing Administration Guide* and *Siebel Tools Reference*.

About Configuring the Get My Price Virtual Business Component

In Siebel Tools, the ISSCDA Get My Price business component contains a listing of all the fields that contain session-specific information when users hit the Get My Price button. The pricing engine uses session specific information to return personalized prices. In [Figure 34](#), quote and quote item are used because the default integration object for Add to Cart is quotes.

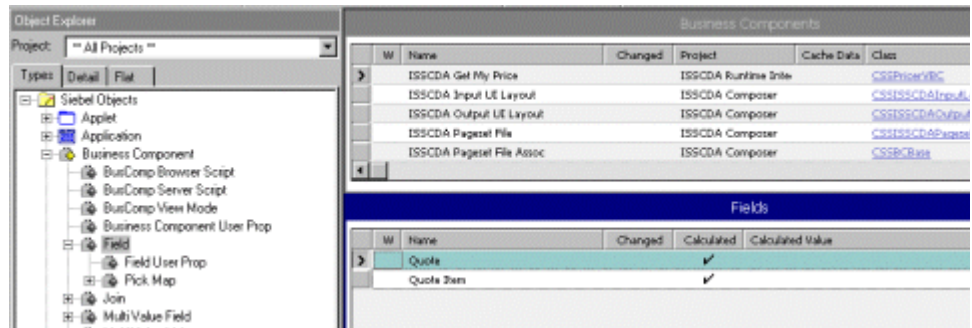


Figure 34. The ISSCDA Get My Price Business Component

When the user clicks Get MyPrice, ISS CDA RT UI Services business service generates session specific information.

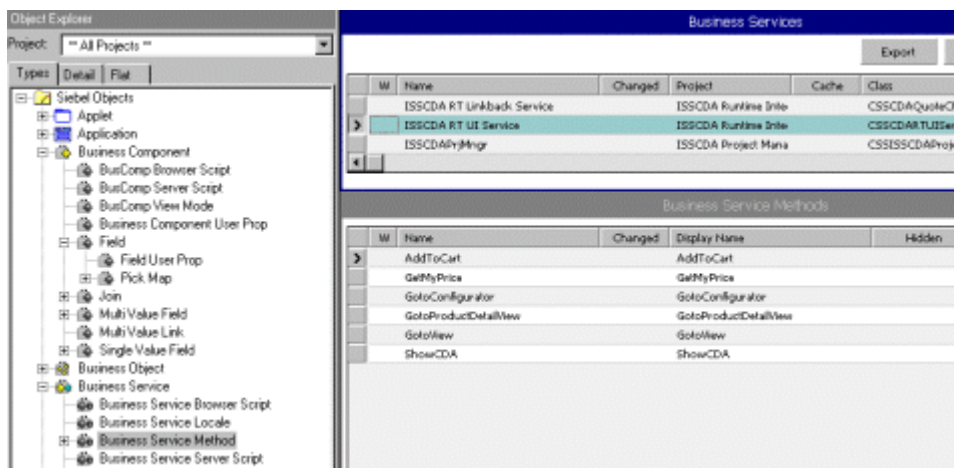


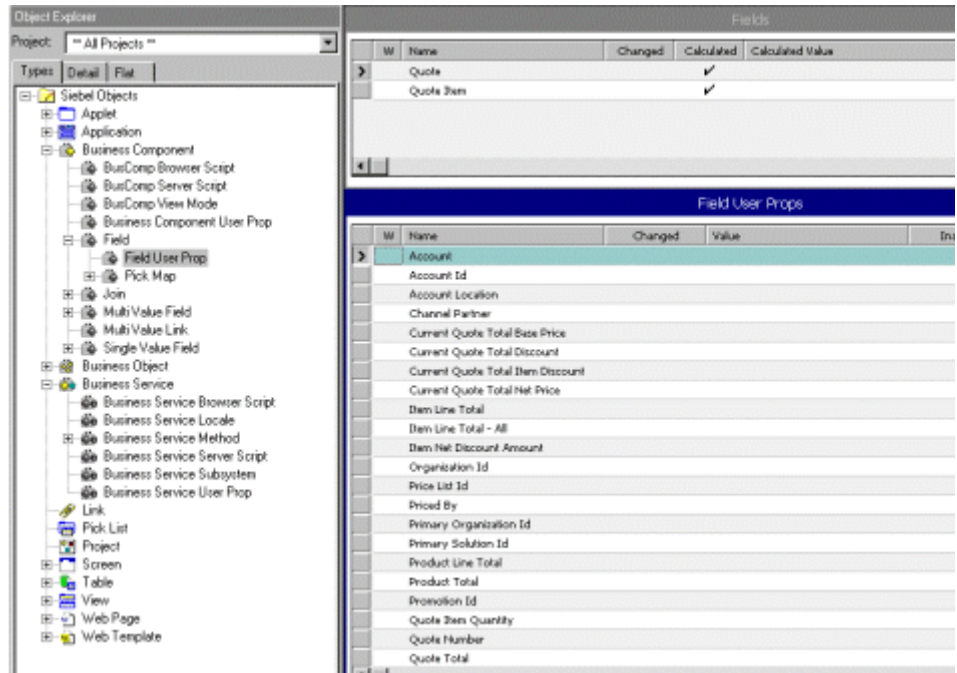
Figure 35. The ISSCDA RT UI Service

The ISS CDA RT UI Services business service generates session specific information in the following order:

- 1 The configurable product information for the current pageset is generated.

See [“Evaluation of the Customizable Product Structure” on page 182](#) for more information.

- 2 If a quote exists, quote specific information (such as Price List Name and Account Name) is generated. Any additional variable is included in the ISSCDA Get My Price virtual business component are also looked up from the quote or order. The fields that are searched on are listed under the Business Component User Properties.



In this example, the business service will try to look up values for Account, Account ID, Account Location, and so on from the quote.

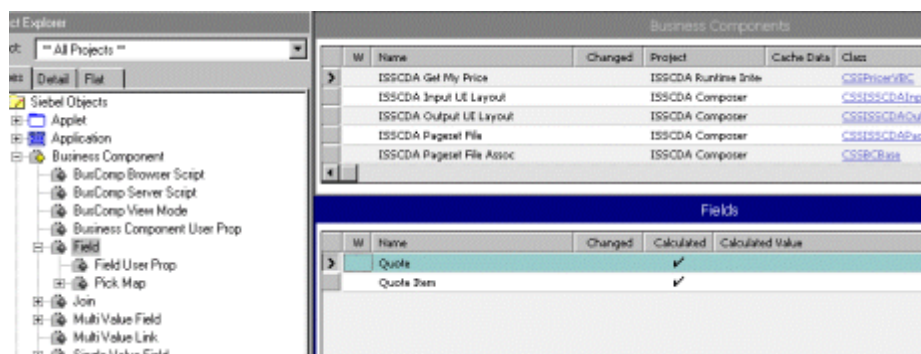
- 3 The business service looks for additional information from the user's profile attributes. If specific values are contained in a quote, they override the values in the profile attributes.
- 4 The session variables are then passed to the pricing engine (the Pricer Business service).

- 5 The Pricer business service along with the ISSCDA RT UI Service business service return a set of prices and associated information (such as product description) back to the user's session.

To configure the application, you can add new variables in the ISS CDA Get My Price virtual business component.

To add variables in the ISS CDA Get My Price virtual business component

- 1 Log into Siebel Tools.
- 2 Go to the business component ISSCDA Get My Price.
- 3 Make sure the integration objects are correct. In this case, the objects used are Quote and Quote Item.



- 4 In the navigation pane, select Business Component > Field > Field User Prop.
- 5 Add fields from the quote or quote item as appropriate.

You can add other variables in the user's configuration file to define information that will appear in the Get My Price window.

To add the field description to the window

- 1** Open the relevant .cfg file in a text editor.
- 2** Go to the section containing the entry for ISSCDAGetMyPriceFields.

ISSCDAGetMyPriceFields = List Price,Product Name,Current Price, Pricing
Comments

- 3** Add more fields to this list using a comma as a delimiter.

When adding additional fields to quotes or orders (depending on the integration object) using Siebel Tools, you must add corresponding fields in the ISS CDA Get My Price Virtual Business component. For more information, see [“About Configuring the Get My Price Virtual Business Component” on page 205](#).

Referencing Other Siebel Data

Runtime Access to Your Pricing Information

Working with a Deployed Application **13**

This chapter describes modifications you can make to your deployed application.

Running in Stand-Alone or Standard Mode

You can run your application in either stand-alone mode or in standard mode.

To run your application in stand-alone mode

- 1 Set the following variables in `app_config.js`:

```
var APP_LOAD_UI_ON_STARTUP = true;
var APP_SIEBEL_INTEGRATION_ON = false;
```

- 2 Connect to your application using a URL that opens `home.htm`.

To run your application in standard mode

- 1 Set the following variables in `app_config.js`:

```
var APP_LOAD_UI_ON_STARTUP = false;
var APP_SIEBEL_INTEGRATION_ON = true;
var APP_SESSION_LENGTH = 15; (Set this number to equal the web
server setting for the number of minutes before the session times
out.)
var APP_DISPLAY_AREA_FRAME = ISStr+".displayArea"(Set
displayArea to your particular display area.)
```

- 2 Connect to your application using `showCDA`.

For more information, see `showCDA` in *Siebel Interactive Designer API Reference*.

Calling Your Application from Another Siebel Application

A Need Advice button that links to an eAdvisor application is provided from Siebel Quotes, eSales Shopping Cart, and eSales Home Page. Click the Need Advice button to launch a placeholder contents list. Develop the contents list in Interactive Designer. The Need Advice button can also link to a specified contents list. For more information, see [“Customizing Get Advice Button Functionality” on page 158](#).

NOTE: Make sure that all the products referenced from Siebel data within a browser-based model to set up interactions with other Siebel applications are set to the same organization that the runtime users may belong to. A product can be set to multiple organizations. If products are not set up in this way, runtime users may encounter errors while making the transition from browser-based applications to other Siebel applications, such as Getting Price and Adding to Cart, and not see certain products, both in connected and in disconnected mode.

To call your application from other Siebel applications, use one of the following methods:

- Associate a pageset with a customizable product.
- Create a button that links to a browser-based project contents list.

See the following sections for a description of how to use these methods.

Referencing Pagesets from Customizable Products

Customizable Products can be configured at runtime by either the server-based eConfigurator or the browser-based eConfigurator. To open the browser-based eConfigurator for a product, you need to associate a pageset with the product in the Product Administration screen. After you make this association, when a user clicks the Customize button for a product, the associated pageset opens in the browser-based eConfigurator.

To associate a pageset with a product

- 1** Create a customizable product and define its structure in the server-based Product Designer.

Do not create any rules and leave all the cardinalities without any values.

For information on creating customizable products, refer to *Product Administration Guide*.

- 2** In Interactive Designer, create an eConfigurator application pageset.
- 3** Release the customizable product.
- 4** Associate the pageset to the product in the Product Master.
 - a** In the Product Administration Screen, select All Products View.
 - b** From the Pageset picklist, select the pageset.

NOTE: This pageset will not be populated with product data. You will need to build the pageset.

Invoking the ShowCDA Method from a Button

Use the following procedure to invoke the ShowCDA method from a button.

To call your application from another Siebel application

- 1** In Siebel Tools, create a button control on your applet.

See *Siebel Tools Reference* for information.

- 2** For the Method property of the button, invoke the ShowCDA method.

Use the syntax:

```
ShowCDA('ProjectDirectory|Pageset ID')
```

NOTE: This step will work only if your applet class inherits from CSSFrame Base.

The contents list of the default project appears when the button is clicked in your applet at runtime, unless you specified a different project.

For more information about the ShowCDA function, refer to *Siebel Interactive Designer API Reference*.

You can also invoke the ShowCDA method to display an application at the applet level rather than the project level. For more information, see the following section.

About Passing in Parameters When Invoking the ShowCDA Method

In Siebel Tools, when you create an applet, you can specify additional information be passed along to the ShowCDA method when a button is clicked, including:

- Project, Pageset, and Dynamic Defaults
- Configuration Properties

When you pass these parameters into the applet's user properties, these parameters will apply to all buttons defined in that applet. If you want to give a button different parameters, you need to do so in a different applet.

Passing in Project, Pageset, and Dynamic Default Information

To pass project, pageset, and dynamic default information to the ShowCDA function through Siebel Tools, use the syntax in the following examples to create entries in the applet's user properties. This allows you to specify what pageset and dynamic defaults are displayed when a user clicks a button.

Example

This example assumes you want to create three Get Advice buttons that each have different projects, pagesets, and dynamic defaults.

To create three buttons that display different applets

- 1 In Siebel Tools, create three buttons in the applets.

For example:

Name	MethodInvoke
Button1	ShowCDA
Button2	ShowCDA1
Button3	ShowCDA2

- 2 Create an applet user property for each method.

Button 1

Name	Value
Method ShowCDA Parameter 1	QuoteId Field Id
Method ShowCDA Parameter 2	Projectlocation Value project1
Method ShowCDA Parameter 3	Pageset Value pageset1
Method ShowCDA Parameter 4	Dyndefs Value dyndefs1
Method ShowCDA Parameter 5	End

Button 2

Name	Value
Method ShowCDA Parameter 1	QuoteId Field Id
Method ShowCDA Parameter 2	Projectlocation Value project2
Method ShowCDA Parameter 3	Pageset Value pageset2
Method ShowCDA Parameter 4	Dyndefs Value dyndefs2
Method ShowCDA Parameter 5	End

Button 3

Name	Value
Method ShowCDA Parameter 1	QuoteId Field Id
Method ShowCDA Parameter 2	Projectlocation Value project3
Method ShowCDA Parameter 3	Pageset Value pageset3
Method ShowCDA Parameter 4	Dyndefs Value dyndefs3
Method ShowCDA Parameter 5	End

In this example, project1, project2, and project3 are the values passed from eConfigurator. The syntax is:

```
Method|ShowCDAxx|Parameter x =  
parameterName|ParameterType|ParameterValue
```

ParameterType can be either Field or Value. If parameterType is Field, ParameterValue is the field name, and the parameter value will be pulled from that field in the business component.

About Passing in Values for Configuration Variables

While you can set variables in the application CFG file as explained in the File Reference chapter of *Siebel Interactive Designer API Reference*, you can also set and pass these values in Siebel Tools. This method allows you to customize the runtime behavior for each applet. For example, you can specify that one applet navigates to a product detail view named Product List view and that another applet navigates to a different view named Product Detail View - Feature View (eSales).

To pass CFG properties to the ShowCDA function through Siebel Tools, use the syntax in the following examples to create entries in the applet's user properties. Create buttons and entries in the applet's user properties just as you did in the previous example.

Example

Name	Value
Method ShowCDA Parameter 1	ISSCDAHeaderBusObjName Value Quote
Method ShowCDA Parameter 2	ISSCDAHeaderBusCompName Value Quote
Method ShowCDA Parameter 3	ISSCDAIntegrationObjName Value Quote
Method ShowCDA Parameter 4	End

Synchronization Setup

This section describes methods for modifying the default synchronization behavior that occurs for:

- Users who are synchronizing project data.

When users select File > Sync > Database, synchronization updates the Siebel database from the server database to the local database.

- End users when they click Get Advice.

When end users access a project by clicking Get Advice, the ISS Project synchronization occurs, updating the project files from the database to the Publish directory. When an end user works in mobile mode, they do not have access to the physical project data. Synchronization downloads a zipped file to the Publish directory that creates the project but does not allow end users to make data changes to the projects.

Modifying the Siebel Synchronize Database Behavior

You can modify the Siebel Synchronize Database behavior so that your mobile users need to navigate to the Synchronize CDA Projects screen to choose synchronization options and synchronize. After setting up the synchronization behavior according to the following method, see [“Using the Synchronize CDA Projects Screen” on page 223](#).

To modify the Siebel database synchronization behavior:

- Determine which synchronization behavior you want to use.
- Select the synchronization behavior in Siebel Tools.
- Compile the SRF file.
- Expose the Synchronize CDA Projects screen if necessary.
- Distribute the SRF file to clients.

Each of these steps is explained in detail in the following sections.

To determine which of the synchronization behaviors to use

- Select from one of the following three options, depending on whether you want to use automatic synchronization (the default behavior) or one of the selective synchronization methods.
 - Download File attachments automatically (default behavior).
 - Request that file attachments are downloaded once and get them automatically thereafter.
 - Always determine whether or not file attachments are downloaded.

To change the synchronization behavior in Siebel Tools

- 1 Open Siebel Tools and log in as the administrator.
- 2 To display all the fields, select Business Components > ISSCDA Sync Projects > Fields.
- 3 In the Pre-Default Value column, set the values for the fields based on the behavior you decided to use in the previous procedure.

If file attachments are downloaded automatically (default behavior), use the following values for the following fields:

Field Name	Pre-Default Value
ISSCDASyncAutoUpdFlg	Y
ISSCDASyncDeferFlg	P
ISSCDASyncDockReqFlg	N
ISSCDASyncDockStatus	N

If the user requests the download once and gets it automatically thereafter, use the following values for the following fields:

Field Name	Pre-Default Value
ISSCDASyncAutoUpdFlg	Y
ISSCDASyncDeferFlg	R
ISSCDASyncDockReqFlg	N
ISSCDASyncDockStatus	N

If the user always determines whether or not file attachments are downloaded, use the following values for the following fields:

Field Name	Pre-Default Value
ISSCDASyncAutoUpdFlg	N
ISSCDASyncDeferFlg	R
ISSCDASyncDockReqFlg	N
ISSCDASyncDockStatus	N

To expose the Synchronize CDA Project screen

NOTE: If you chose to have synchronization occur automatically, you do not need to expose this screen. If you chose the other options, you will need to expose this screen; users will go to this screen to request file attachments.

- 1 Start the Siebel application.
- 2 Log in as SADMIN on the server.
- 3 Select Application Administration > Views.
- 4 Add the “ISSCDA Synchronize View” to the existing list of views and select Local.

- 5 Select Application Administration > Responsibilities.
- 6 Add the “ISSCDA Synchronize View” to the desired responsibility.

To compile the SRF file

- 1 Open Siebel Tools.
- 2 Go to Tools > Compile.
- 3 Select the ISSCDA Project Manager project in the list.
- 4 Change the target Repository File if necessary.
- 5 Click Compile.

To distribute the SRF file

- Distribute the new SRF file to mobile users using the Siebel software distribution method.

Using the Synchronize CDA Projects Screen

If you modified Synchronizing Projects behavior to synchronize selectively, your mobile users will need to navigate to the Synchronize CDA Projects screen to synchronize projects.

To use the Synchronize CDA Projects screen

- 1 From the Site Map, select UserProfile Preferences.
- 2 Select Synchronize CDA Projects.

The Synchronize CDA Projects screen appears.
- 3 If the Local field is not selected and the version is not zero, select the Request field for the project you want updated.
- 4 If the Update Available column is checked for a project, you can obtain a more recent version of that project by synchronizing.
- 5 Select File > Synchronize > Database to synchronize the database.
- 6 If your browser is open when you synchronize, clear the cache to see the latest data.

About Modifying the Project Synchronization Behavior

In mobile mode, Siebel synchronization updates the projects from the server database to the local database. When the end user clicks Get Advice, the Siebel application updates the project from the local database and publishes it under the project Publish directory.

In connected mode, there is no need to do a Siebel sync because you are connected to the server database. When the end user clicks Get Advice, the Siebel application updates the project from the server database and publishes it under the project Publish directory.

By default, the project is synchronized when the end user accesses the project in both mobile and connected modes. Two configuration variables are provided to modify this synchronization behavior in connected mode:

- ISSCDAAutoDeployment
- ISSCDADeploymentMode

Although these variables are active in mobile mode, it is not recommended that you modify them in mobile mode.

Disabling Automatic Project Synchronization

If you will be developing and modifying projects on the server, you may want to disable automatic synchronization.

To disable automatic project synchronization

- 1** In your Siebel application configuration file (for example, uagent.cfg for Call Center), search for [ISSCDA].
- 2** Under this section, add the following line:

```
ISSCDAAutoDeployment = FALSE
```
- 3** Save the file.
- 4** Restart the application.

Synchronizing All Projects When Get Advice is Clicked

If your end users want to synchronize all projects with one click, you can change the ISSCDADeploymentMode variable. By default, this variable is set to ONDEMAND and projects get synchronized to the project Publish directory only when the user accesses a project.

To synchronize all projects when Get Advice is clicked

- 1** In your Siebel application configuration file (for example, uagent.cfg for Call Center), search for [ISSCDA].
- 2** Under this section, add the following line:

```
ISSCDADeploymentMode = ALL
```
- 3** Save the file.
- 4** Restart the Siebel application.

About Working in Mobile Client Mode

Make end users aware of the following guidelines for working in mobile client mode.

- Synchronizing does not remove outdated files from the client. After your end users synchronize data, they will need to manually delete files that have been removed from projects. You will need to communicate to your end users which files have been deleted from a project and should be removed from their client machine.
- End users should always synchronize their database before and after using mobile client mode.
- If the end user uses the same installation for connected and mobile mode, make sure the mobile database is synchronized after switching from connected mode to mobile mode.
- If a project is modified while a user is working, the cached project may remain active. The browser's cache remembers the pages that were loaded before and might not load the modified project. In order to load the modified project, clear the cache or restart the application.

About Unframing Customer and Partner Applications

Customer and Partner applications use HTML frames for optimum compatibility when running with applications. HTML frames allow portions of the browser window to scroll independently of the rest of the window. For example, with HTML frames it is possible to place main navigation in one frame and content in another frame. You can scroll the content while the main navigation remains in a fixed location.

Limitations of Unframing

Customer and Partner applications can also be unframed. Siebel eBusiness applications require frames and cannot be unframed.

Before choosing to unframe your application, consider the following limitations to using unframed applications with applications.

- HTML Frames allow the user to view the Customer and Partner global navigation links and other information while they are in an application. This allows them to exit or cancel the session by navigating to another view. When the Customer and Partner application is unframed, an application occupies the entire browser window when launched. This makes the Global Navigation Links inaccessible.
- In an unframed application, all UI elements exist in the same window. Therefore, the contents list may scroll off the page as a user scrolls down. This is a limitation of the unframed HTML environment and a factor in choosing to frame applications.

If you decide to implement your Customer and Partner application without HTML frames, use the following procedure to unframe your application. The following example unframes a Siebel eSales application. For more information, refer to *Siebel Tools Reference*.

Implementing a Customer and Partner Application Without Frames

Use the following procedure to implement a Customer and Partner application without frames.

To implement a Customer and Partner application without frames

- 1** Open Siebel Tools.
 - a** In Object Explorer, select the Type tab and click the Application node.
 - b** From the Application list, select Siebel eSales.
- 2** From the View menu, choose Windows and select the Properties window.
- 3** Record the value for the property called Container Web Page.

In Siebel eSales the value is CC Container Page (eSales).
- 4** In Object Explorer, select Web Page, and in the Web Pages list, select the Container Web Page named CC Container Page (eSales).
- 5** Copy this Web Page and label it CC Container Page (eSales) No Frames.
- 6** With the new Web Page selected in the list, select the value for the Web Template property in the Properties window. The value should read Page Container (Framed). From the drop-down list, choose Page Container No Frames. Step off the current Web Page record to store this change.
- 7** In Object Explorer, select Application.

Verify that the Siebel eSales application is selected in the Application list.
- 8** In the Properties window, select the property Container Web Page. From the drop-down list, choose the newly created CC Container Page (eSales) No Frames. Step off the current application record to store this change.
- 9** Compile your changes.

The Siebel eSales application is now associated with the unframed Customer and Partner application container. This container is associated with the unframed SWE container template (dCCPageContainer_NoFrames.swt). The next time you launch the application, you will notice that content and navigation are loaded into a single window.

Working with a Deployed Application

About Unframing Customer and Partner Applications

Working with the Project Files **14**

This chapter describes the Interactive Designer files that make up your application, and the methods for viewing and editing them.

About the Project Files

Interactive Designer uses the information you enter in the project to create the HTML and JavaScript files. These files define the basic structure and appearance of the application. These files are stored in the Siebel File System and listed in the Project Files tab. When you preview or deploy the project, these files and directories are created in two different directories, both under ISSRUN\CDAPROJECTS. For information about the Interactive Designer files, refer to *Siebel Interactive Designer API Reference*.

Interactive Designer provides a structured approach to creating and managing the files that make up an application. An Interactive Designer project is a collection of project objects that represent the files in an application.

Many of the project objects are identical, in name and underlying format, to the application files they represent. For example, the Application UI Definition file object in an Interactive Designer project is precisely the same file as the Application UI Definition file in the associated application.

However, some application files are not represented as file objects in Interactive Designer. For example, a Contents file in an application is represented in Interactive Designer as a contents list table.

Certain application files are represented by a collection of Interactive Designer project objects, as opposed to a single object. For example, a single Configuration Data file in an application is represented in Interactive Designer as a collection of Configuration tables.

Interactive Designer generates application files from Interactive Designer project objects when you preview or deploy your project.

Interactive Designer Project Structure

The organization in which Interactive Designer presents the project objects reflects the organization of the associated application.

Project objects that define the overall appearance and functionality of the application are considered application-level objects. These application-level objects include the *Application UI Definition file* and a *contents list table*.

The application presents information related to one or more categories. Interactive Designer groups all of the project objects related to a single category into a pageset.

Within each pageset, the project objects are organized into two categories: data objects and interface objects. Each data object is further classified as a feature data object or a configuration data object. The feature data and configuration data objects are *Feature tables* and *Configuration tables* respectively. The interface objects within a pageset are the *Pageset UI Definition file* and the *Pageset UI Registry file*.

Every pageset in an application, at a minimum, contains the following items:

- A Pageset UI Definition file (oc_default_ui.htm)
- A Pageset UI Registry file (pCar_i.htm)
- At least one Configuration table
- One or more Feature tables
- One or more display pages

The Project Files Tab

The Project Files tab contains a list of the directories and files that make up your application. These files include HTML, Javascript, and images. You can use a text editor to change some of the files displayed on the Files tab.

Every application contains the following directories and files:

■ Engine File Directories

When you deploy or preview a project, first the engine files are copied from the file system source directory on the server,

< *Siebel Root Directory* > \ISSTEMPL\ < *Localization Code* > \EngineSourceFiles, to the project runtime directory for the Project Files Tab. Then any project files stored in the database are copied to the runtime directory. If you customize engine files on the server and files with the same name exist in the database (app_config.js, for example), your customized files will be overwritten. Therefore, if you want your customizations to persist after deployment or preview of the project, store them in the database by adding them to the Project Files tab.

The following engine directories and files are associated with each project:

- The custom directory contains JavaScript files that you can edit to customize the behavior of the applications' engine without modifying the core engine code itself. You can also use the app_config.js file in this directory to set configuration variables, such as the size of the About window, for your application.
 - The ds directory contains the Pageset Properties file, pagesetID_x.js. It also contains product data files generated from the information you enter in Interactive Designer Configuration and Feature tables. Do not edit the generated product data files. You can change information in the tables and regenerate the data files if you need to.
 - The jd directory contains the application module registry and its associated files. Do not edit files in this directory unless you are developing your own application modules.
- #### ■ Application File Directories

The pg directory contains the following application files:

- Pageset UI Registry files.
- Pageset UI Definition files.
- Display pages for all the pagesets in your application.
- A cascading style sheet, onlink.css, that defines the appearance of the contents list.

- Any additional HTML and image files used to customize the appearance of display pages.

The UI Directory contains files that define the appearance of the eAdvisor application. These files apply to the application in general and not to any specific pageset.

Viewing the Project Files

Use the following procedure to view the project files.

To view the project files

- Click the Project Files tab of the Project screen.

All files that make up the project are listed.

The following information appears:

- File name
- File type
- Directory in which the file lives
- Last modified date
- Notes about the file

To open and view a project file

- 1** In the Project Files tab of the Project screen, place your mouse pointer over the name of the file you want to open until it appears as an underlined link.
- 2** For HTML files, click the link.
The text appears in your browser.
- 3** For files of a type other than HTML, click the file.
A dialog opens asking if you want to open or save the file.
- 4** Save the file to your hard drive.
To view the file, open it in the appropriate application.

For more information on the default project directories and files, refer to *Siebel Interactive Designer API Reference*.

Creating a File Attachment

Use the following procedure to create a file attachment.

NOTE: Attached files increase the amount of time some Interactive Designer operations take. Migrate, Import/Export, Deploy, Copy Project, and Preview can take between 0.25 and 0.50 seconds longer per attached file. To improve operation time, keep your attached files to a minimum.

To create a file attachment

- 1** In the Project Files tab of the Project screen, choose New Record from the applet menu.
A new File Attachment applet appears.
- 2** Click the applet menu button to select the attachment name.

- 3 Enter the directory and any notes.

NOTE: The combination of the attachment name and the directory name must be unique.

- 4 Click Save.

The file is added as an attachment to the list of files displayed in the Project Files tab.

Copying a File Attachment

Copy a file to create a new instance of the file in the Siebel File System.

To copy a file

- 1 Select the file.
- 2 From the applet menu, choose Copy Record.

A New File applet appears, displaying the data for the file you are copying.

- 3 Enter a new file name and directory.

These values must be unique throughout the project.

- 4 Click Save.

The new record appears in the Project Files list.

Deleting a File Attachment

Use the following procedure to delete a file attachment.

NOTE: If you delete a file that is needed for the application, your application might not run properly. After deleting files, preview your application to make sure it still runs properly.

To delete a file attachment

- 1 Select the file you want to delete.
- 2 Click Delete.

Editing a File Attachment

Use the following procedure to edit a project file or project file record.

NOTE: Do not modify files in the CS directory.

To edit a project file

- 1 Save the file you want to edit to your hard disk.
- 2 Edit the file.
- 3 In the project, delete the file you edited on your hard drive.
- 4 Add the updated file to the project.

To edit a project file record

- 1 In the Project Files tab of the project screen, select a file.
- 2 Click Edit.

A dialog appears with the file attachment information.
- 3 Edit the information and click Save.

Implementation of Multi-Variable and Cascading Triggers

A

This chapter discusses the implementation of multi-variable and cascading triggers.

Installation of the Multi-Variable and Cascading Triggers Module

In order to use the multi-variable triggers and cascading triggers features, follow the installation and implementation instructions below.

CAUTION: This functionality is highly advanced; use of it may not guarantee referential integrity. Make sure that all feature table items are correctly associated to the multi-variable configuration table. It is highly recommended that you work with a systems integrator if you use this advanced functionality.

To install the module

- 1 Modify the Module Registry file (jd/moduleRegistry.htm).
 - a Include the Triggers API script file by adding the following script include code:

```
<script src="triggersEvents.js"></script>
```

- b Register the Triggers Module by adding the following call to the registry:

```
ISS.RegisterModule("triggerscode",ISS.GetCSPath()+"triggersCode.htm", ISSStr+".triggerscode", ISS.ON_DEMAND);
```

Be sure the call is placed before the "ISS.ModuleRegistrationComplete();" call.

- 2 Add the variable APP_EVALUATE_ALL_TABLES to custom/app_config.js as follows. If the variable is already defined, set the value to true.

```
var APP_EVALUATE_ALL_TABLES = true;
```

- 3 Modify the Kernel file (kernel.htm).
 - a Include the Triggers API script file by adding the following script include code:

```
<script src="cs/triggersAPI.js"></script>
```

- b** Create a new hidden frame in the kernel frameset.

Expand the frameset definition to accommodate the extra frame by adding an extra `,*` to indicate an extra row.

```
<FRAMESET rows="*,*,*,*,*,*,*,*,*,25%,*,*,*,*" border=0
frameborder=0 framespacing=0>
```

The frame definition should be as follows:

```
<FRAME marginwidth=0 marginheight=0 src="javascript:''"
name="triggerscode" scrolling="no">
```

- 4** Upload the (jd/moduleRegistry.htm) and (kernel.htm) into Interactive Designer.

To implement the module

- 1** Add the PEP_DATASET_LOADED function to custom/customCode.js as follows. If this function is already defined, modify it to call `ISS.Triggers_PreDatasetLoaded` and return `false`, as shown.

```
function PEP_DATASET_LOADED(event,window,data) {
    ISS.Triggers_PreDatasetLoaded(event,window,data);
    return false;
}
```

- 2** Add the COP_AfterInputValueSet function to custom/customCode.js. If this function is already defined, modify it to call `ISS.Triggers_AfterInputValueSet` as shown.

```
function COP_AfterInputValueSet(table,selIndex) {
    ISS.Triggers_AfterInputValueSet(table,selIndex);
}
```

Implementation of Multi-Variable and Cascading Triggers

Installation of the Multi-Variable and Cascading Triggers Module

- 3 Add the PEP_ENGINE_RESULTS_GENERATED function to custom/customCode.js. If this function is already defined, modify it to call ISS.Triggers_EngineResultsGenerated and return false, as shown.

```
function PEP_ENGINE_RESULTS_GENERATED(event,window,results) {  
    ISS.Triggers_EngineResultsGenerated(event,window,results);  
    return false;  
}
```

- 4 Register each multi-variable or cascade trigger construct in the pageset's UI Information file (p100_i.htm). These calls should be placed following the ISS.StartUIInfo() call. One API call is made for each multi-variable or cascade construct.
- 5 Verify that the API RegisterPriorityPages is being used in the UI Information file; any location which contains a multi-variable target must be registered as a priority page.

Index

A

- advising solution. *See* eAdvisor
- app_config.js file, described 232
- Application UI Definition file
 - about 168
 - modifying 169
 - path and file name 168
- Application UI Definition files
 - Contents List, about modifying 170

B

- browser-based application
 - directories, contained in 231
 - hybrid mode, running 212
 - module registry location 232
 - price list, associating with 201
 - stand alone mode, running 212
 - unframed .Com applications, about 226
 - unframed .Com applications,
 - implementing 227
- browser-based applications
 - architecture diagram 18
 - deploying application 57
 - eAdvisor application, about 26
 - eAdvisor application, creating 28, 30
 - eConfigurator application, about 27
 - eConfigurator application, creating 31, 32
 - purpose and about 17
 - Siebel application, calling from 215
- BuildTarget
 - about and example 172, 173
 - described 172
- BuildWidget
 - about and example 172, 173
 - described 172

Business Components

- about binding to Feature or Configuration tables 177
- column, described 177
- list of 179
- name record, creating new 179
- Pick Applet Name record, creating new 180
- Pick List Name record, creating new 180

C

- CODE column, described 101
- columns
 - Configuration tables, editing 82
 - Feature tables, editing in 111
- .Com applications
 - unframing, implementing 227
 - unframing, about 226
- Configuration Table Designer view, using
 - and screen example 43
- Configuration tables
 - about 64
 - about using 100
 - column types, described 68, 70
 - columns, editing 82
 - Configuration Table Designer view, using
 - and screen example 43
 - Configuration Table view, using and screen example 42
 - copying 80, 110
 - creating 75
 - cross-sell/up-sell messages, about 84
 - cross-sell/up-sell messages, adding to data model 85
 - cross-sell/up-sell messages, adding
 - uplink 85

- defining output targets 100
 - deleting 81
 - exception messages, about and
 - examples 83
 - exception messages, creating 83
 - matching process, steps of 65, 67, 134
 - rows, editing 82
 - Search view, using and screen
 - example 38
 - viewing 75
 - configuring product/solution
 - eConfigurator application, about 27
 - eConfigurator application, creating 31, 32
 - Contents List
 - about 156
 - application, defining display inside 170
 - Contents List view, using and screen
 - example 36
 - copying 161
 - creating 159
 - defining display inside a pageset 171
 - defining display inside the
 - application 170
 - deleting 162
 - displaying 170
 - editing 163
 - HTML properties, defining 157
 - location, about modifying and sample
 - layout 170
 - pageset, defining display inside of 171
 - populating 158
 - RegisterContentsListFrame function 170
 - SetContentsListFrame function 171
 - viewing 158
 - copying
 - Configuration tables 80, 110
 - Contents List 161
 - input UI controls 145
 - output targets 150
 - projects 51
 - cross-sell messages
 - about 84
 - data model, adding to 85
 - uplink, adding 85
 - custom directory, described 232
- D**
- data models
 - pagesets, validating 96
 - validating 54
 - DEFAULT column, described 101
 - deleting
 - Configuration tables 81
 - Contents List 162
 - Feature tables 111
 - input UI controls 145
 - output targets 151
 - pagesets 95
 - projects 52
 - DESC column, described 101
 - directories, contents 231
 - ds directory, described 232
- E**
- eAdvisor application
 - about 26
 - creating application 28, 30
 - eConfigurator application
 - about 27
 - creating 31, 32
 - editing 95
 - columns in Configuration tables 82
 - Contents List 163
 - Feature table rows 112
 - Feature tables columns 111
 - input UI controls 146
 - output targets 152
 - pagesets 95
 - Project files 236
 - projects 52
 - rows in Configuration tables 82
 - exception messages
 - about and examples 83
 - creating 83, 84
 - description 64

F

- Feature Table Designer view, using and screen example 41
- Feature Table Editor view, using and screen example 39
- Feature tables
 - column layout and examples 101
 - columns, editing 111
 - Configuration Table view, using and screen example 42
 - creating 103
 - defining input widget values 100
 - deleting 111
 - Feature Table Designer view, using and screen example 41
 - linked tables, creating 109
 - rows, editing 112
 - Search view, using and screen example 38
 - viewing 103
- Field Name column, described 177
- file attachment
 - creating 234
 - deleting 236

H

- HTML
 - properties, defining in Contents List 157
 - unframing .Com applications, about 226
 - unframing .Com applications, implementing 227
- hybrid mode
 - projects, previewing 55
 - running 212

I

- id directory, described 232
- incremental versioning, using 62
- input UI controls
 - about and example 140
 - copying 145

- creating 141, 144, 146, 150, 153
- customizing, about and example 172, 173
- deleting 145
- display page, generating 146
- editing 146
- integrating
 - Siebel shopping cart, integrating with browser-based application 192

K

- key column, about and example 68, 69

L

- linked tables, creating 109
- locking
 - pagesets 91
 - pagesets in a team environment 61

M

- MAIN configuration table, about 64
- mainArea, Interactive Designer template, about 168
- migrating a project to 7.0 51

O

- organization of guide 12
- output targets
 - copying 150
 - creating 148
 - deleting 151
 - editing 152
- output UI controls
 - customizing about and example 172, 173
 - Output UI Layout view, using and screen example 44
- output UI, display page, generating 153

P

- pageset
 - display page area, about defining 168

- MAIN configuration table, about 64
 - project objects, contained within 231
 - Validate Results view, using and screen example 37
 - Pageset Properties file, location of 232
 - Pageset Team, controlling team access 61
 - Pageset UI Definition files
 - Contents List, about modifying 170
 - location of 232
 - RegisterUI function 168
 - Pageset UI Registry files
 - location of 232
 - path and file name 168
 - RegisterExceptionFrames function 168
 - RegisterPageLocation function 168
 - pagesets
 - about 88
 - creating 92
 - data search, performing 53, 97
 - deleting 95
 - editing 95
 - Feature Table Editor view, using and screen example 39
 - locking/unlocking 91
 - Pageset Files view, using and screen example 45
 - particular pagesets, viewing 91
 - team environment, locking pagesets 61
 - validating 96
 - your pagesets, viewing 90
 - particular projects, viewing 49
 - pg directory, described 232
 - Pick Applet Name record, creating 180
 - Pick List Name record, creating new 180
 - positions, setting and modifying 48
 - price list
 - browser-based model, associating with 201
 - pricing columns, about creating multiple pricing options 201
 - Project files
 - about 230
 - editing 236
 - file attachment, creating 234
 - file attachment, deleting 236
 - project structure, described 230
 - record, editing 236
 - viewing 233
 - Project Files tab, contents of 231
 - Project Files view, using and screen example 37
 - Project Team, setting up team access 61
 - project UI files
 - about 168
 - editing 167
 - projects
 - about 48
 - copying 51
 - creating 50
 - deleting 52
 - deploying 57
 - editing 52
 - migrating a 4.0 project to 7.0 51
 - particular projects, viewing 49
 - previewing 55
 - Project Files view, using and screen example 37
 - team environment, working in 60
 - Validate Results view, using and screen example 37
 - validating 54
 - version number, viewing 62
 - your projects, viewing 49
- R**
- roles, setting and modifying 91
 - Row type column, about 102
 - rows
 - Configuration tables, editing 82
 - Feature tables rows, editing 112
- S**
- search
 - data search, performing 53, 97

- Search view, using and screen
 - example 38
 - Sequence column
 - about 102
 - SetContentsListFrame function, using 171
 - Shared Target column, described 178
 - shopping cart
 - Siebel shopping cart, integrating with browser-based application 192
 - Siebel applications
 - browser-based application, calling from 215
 - Siebel search, performing 53, 97
 - Siebel shopping cart, integrating with browser-based application 192
 - stand alone mode
 - running 212
 - stand-alone mode
 - projects, previewing 55
- T**
- tables, linking 109
 - Target tables
 - about 116
 - creating 118, 121
 - example, creating 121
 - UI controls, tying table to 120, 123
 - team access, setting up 61
 - team environment, working on projects 60
 - Trigger tables
 - about 116
 - creating 118, 119, 121, 122
 - example, creating 121
 - UI controls, tying table to 120, 123
 - type 0 columns, described 69
 - type 1 column, about and example 68
 - type 99 column, described 70
- U**
- UI controls
 - input UI control, about and example 140
 - input UI control, creating 141, 144, 146, 150, 153
 - input UI controls, editing 146
 - Output UI Layout view, using and screen example 44
 - restricting the values displayed 116
 - UI Layout view, using and screen example 44
 - UI layout
 - browser-based application sample and description 138
 - view, using and screen example 44
 - UI, customizing
 - project UI file, editing 167
 - project UI files, about 168
 - unframing .Com applications
 - about 226
 - implementing 227
 - unlocking pagesets 91
 - updating projects to 7.0 51
 - up-sell messages
 - about 84
 - data model, adding to 85
 - uplink, adding 85
 - user interface controls. *See* UI controls
- V**
- valid configuration, description 64
 - Validate Results view, using and screen example 37
 - validating
 - pagesets 96
 - projects 54
 - version numbers, viewing 62
 - viewing, controlling access in team environment 60

