Siebel CRM

Advisor Administration Guide

Siebel Innovation Pack 2015

E24718-01

May 2015

ORACLE

Siebel CRM Advisor Administration Guide, Siebel Innovation Pack 2015

E24718-01

Contents

3 Overview of the Siebel Advisor Interface

4 Overview of Building Advisor Applications

5 Working with Advisor Projects

6 Working with Advisor Pagesets

7 Working with Advisor Feature Tables

8 Working with Advisor Configuration Tables

9 Building the UI with Advisor

10 Using Advisor Contents Lists

11 Working with Deployed Advisor Applications

12 Advanced Modeling for Advisor

13 Customizing the UI of Advisor Pages

14 Referencing Other Siebel Data from Advisor

15 Working with Advisor Project Files

List of Tables

List of Figures

List of Examples

Preface

This guide covers the features and functionality of Siebel Advisor Administration.

Audience

This guide is intended for users of Siebel Advisor Administration.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at

http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc

Access to Oracle Support

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

Related Documents

For more information, see the following documents on Oracle Technology Network:

- *Siebel Advisor API Reference*
- *Applications Administration Guide*
- *Siebel Business Process Framework: Workflow Guide*
- *Siebel Order Management Guide*
- *Siebel Personalization Administration Guide*
- *Siebel Pricing Administration Guide*
- *Siebel Product Administration Guide*
- *Siebel Security Guide*
- *Siebel Tools Reference*

Conventions

The following text conventions are used in this document:

Convention	Meaning
italic	Italic type indicates book titles, emphasis, a defined term, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, code in examples, text that appears on the screen, or text that you enter.

1

What's New in This Release

This chapter describes the changes in this version of the documentation.

What's New in Oracle's Siebel CRM Advisor Administration Guide, Siebel Innovation Pack 2015

No new features have been added to this guide for this release. This guide has been updated to reflect only product name changes.

Note: Siebel Innovation Pack 2015 is a continuation of the Siebel 8.1/8.2 release.

2

Overview of Siebel Advisor

This chapter does the following:

- Introduces Advisor
- Describes the browser-based application architecture and the Advisor administration environment for creating browser-based applications

This chapter includes the following sections:

About Siebel Advisor

Advisor is the administration environment used to author and maintain interactive selling applications. Use Advisor to create projects which contain browser-based applications. After you deploy the project, it is referred to as an Advisor application.

Advisor applications use a Web browser to present product specifications, prices, images, schematics, applicability guidelines, and other information that customers require to select products.

Advisor can be used to:

- Recommend products services, or a course of action based on user needs
- Select and configure products based on product features and user requirements

Business services associated with Advisor administration generate the JavaScript, HTML, and image files needed for an Advisor application. The applications run inside other Siebel applications as part of a Siebel application or as stand-alone applications.

When a user opens an Advisor 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 define an application, you enter information about features and configuration of your products. You also enter basic information about how you want your application to appear.

Business services associated with Advisor use this information to produce JavaScript and HTML files.

After Advisor generates applications, you can further customize their appearance and behavior with tools such as HTML editors.

Advisor Application Architecture

In Advisor 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 Advisor applications, files needed for the application are statically published after the application data model and the user interface are developed. In some deployments, you 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 About Referencing Other Siebel Data from Advisor Applications and *Siebel Advisor API Reference*.

Figure 2–1, "Siebel Advisor Application Architecture" illustrates Advisor application architecture.

Figure 2–1 Siebel Advisor Application Architecture

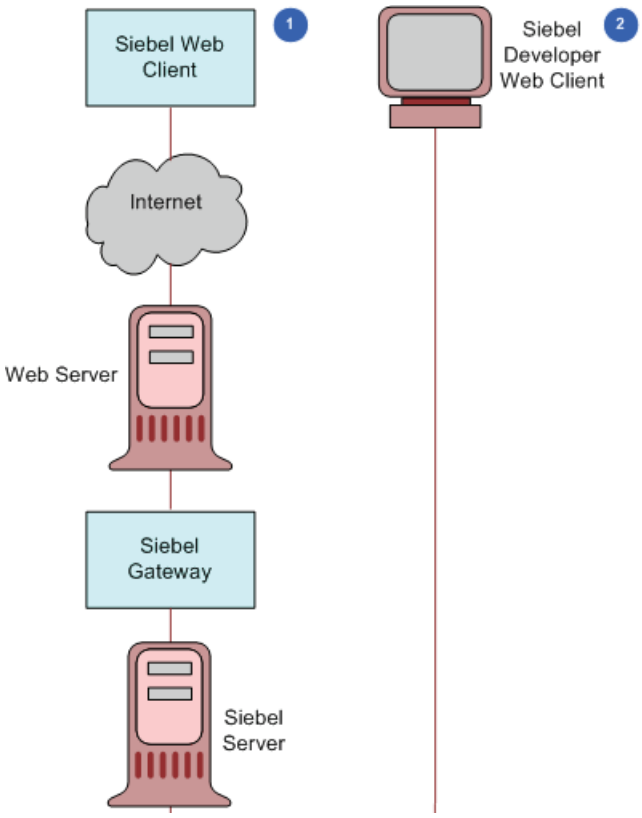


Figure 2–1, "Siebel Advisor Application Architecture" shows two types of deployment:

- Siebel Web Client
- Siebel Developer Web Client

Siebel Web Client Deployment

In Figure 2–1, "Siebel Advisor Application Architecture", the Siebel Web Client deployment path is marked 1. As you create your application in Advisor, 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 an Advisor Project" on page 5-5.

Siebel Developer Web Client Deployment

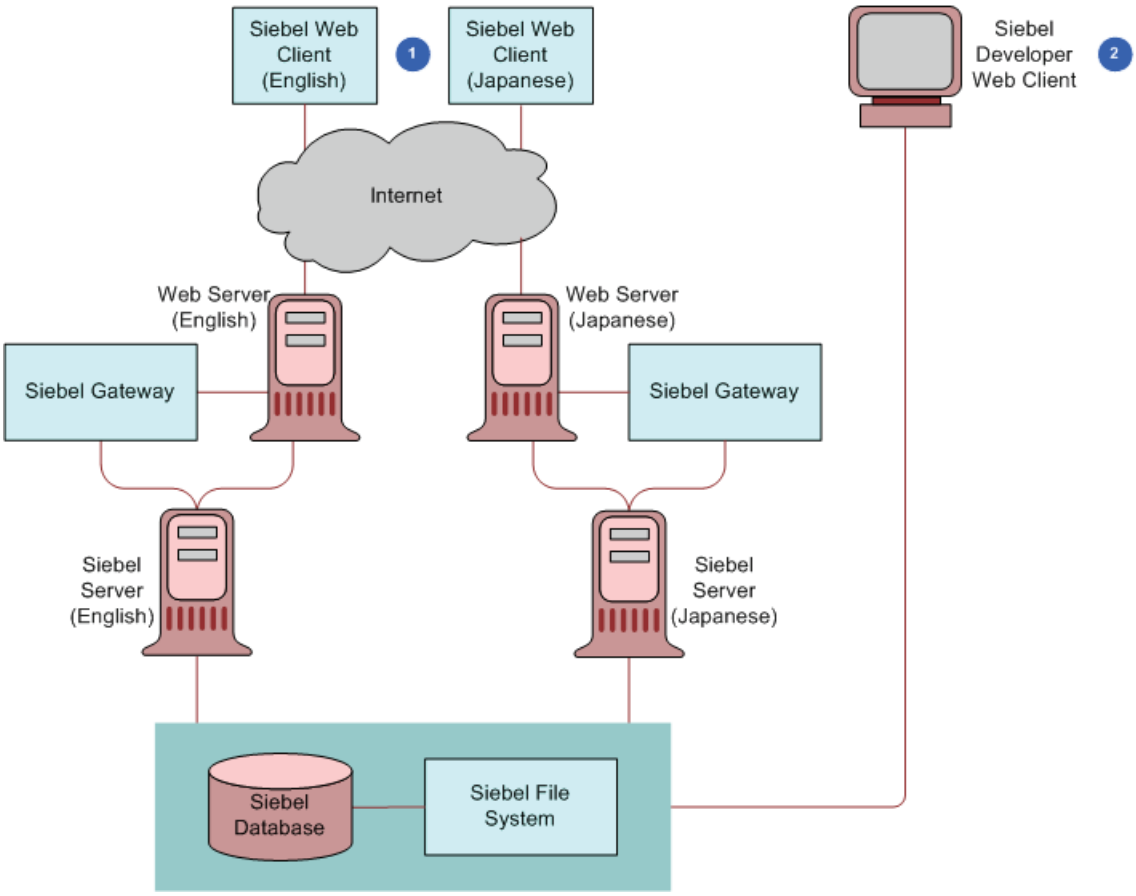
In Figure 2–1, "Siebel Advisor Application Architecture", the Siebel Developer Web Client deployment path is marked 2. In this setup, the Web client, Web server, and Siebel server are on the same machine. You work in Advisor on the Developer Web Client. You save your data to the Siebel database and Siebel 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 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 an Advisor Project" on page 5-5.

Figure 2–2, "Global Deployment Architecture" illustrates the global deployment architecture for an Advisor application.

Figure 2–2 Global Deployment Architecture



Performance Considerations for Advisor Applications

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

Project Size

When determining how you 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 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 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 Advisor 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 Application Deployment Methods

Siebel offers two deployment options for Configurator applications:

- Browser-based, administered through Advisor

  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. Only browser-based Configurator is discussed in this guide.

- 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. Server-based Configurator is discussed in *Siebel Product Administration Guide*.

While it is possible for Advisor to be used to complete the configuration of a customizable product, the standard best practice within the Siebel Application would be to use Advisor to guide the customer to the selection of a particular customizable product and then pass the user to the Siebel Configurator for final configuration of the customizable product.

Siebel Advisor and Open UI

Siebel Advisor does not support Open UI for IP2014. If you are using the High-Interactivity user interface, you can upgrade to IP2014 and continue to use Siebel Advisor.

If you are using Open UI, you can use Oracle Policy Automation to create a rule base that guide customers through purchasing decisions in the same way that Siebel Advisor does.

Setting Up Oracle Policy Automation to Replace Siebel Advisor

To set up Oracle Policy Automation to replace Siebel Advisor, the administrator must:

- Set up Oracle Policy Administration Connector to integrate the Siebel application with Oracle Policy Administration.
- Create the Oracle Policy Automation rule bases that are needed to help with purchasing decisions.

For more information creating rule bases, see *Oracle Policy Automation Connector for Siebel Developer's Guide* on Oracle Technology Network.

Using Oracle Policy Automation to Replace Siebel Advisor

After administrators have set up Oracle Policy Automation as a replacement for Siebel Advisor, users who are using Open UI can access the rule bases to guide purchasing decisions by using the following procedure.

To use Oracle Policy Automation to replace Siebel Advisor

1. Navigate to the Quotes or Sales Order screen, then to the My Quotes or My Sales Orders view.
2. Click the name of a quote or sales order to display the Catalog, and then click Get Advice in the view bar.

   The Siebel application displays a list of Advisors, listing all Oracle Policy Automation rules bases available for needs analysis.
3. Click the Advisor that you want.

   The Siebel application opens an Oracle Policy Automation session that guides the user through the decision and displays a list product results that meet the customer's needs.
4. Copy a product name from the list of product results and paste it into a Quote or Order line item.

Siebel Advisor and Open UI

Siebel Advisor does not support Open UI.

If you are using the High-Interactivity user interface, you can upgrade to IP2014 and continue to use Siebel Advisor. You must disable the Advisors button, which supports Open UI. For more information, see Disabling the Advisors Button.

If you are using Open UI, you can use Oracle Policy Automation to create a rule base that guides customers through purchasing decisions in the same way that Siebel Advisor does. For more information, see Process of Setting Up Oracle Policy Automation to Replace Siebel Advisor.

Disabling the Advisors Button

If you are using the High Interactivity interface instead of using Open UI, you must disable the Advisors button, which users click to display the Advisors for Open UI.

To disable this button, configure this user property in Siebel Tools as follows:

- Applet Name: ISSCDA Personalized Contents List Applet
- User Property Name: CanInvokeMethod: ShowPopup
- Value: IIf(GetProfileAttr("IsOpenUI") IS NOT NULL AND GetProfileAttr("IsOpenUI")=1,TRUE,FALSE)

Process of Setting Up Oracle Policy Automation to Replace Siebel Advisor

To set up Oracle Policy Automation to replace Siebel Advisor, the administrator performs the following steps:

1. Integrating the Siebel Application with Oracle Policy Administration
2. Creating the Oracle Policy Automation Rule Base
3. Verifying the Integration

Integrating the Siebel Application with Oracle Policy Administration

You must integrate the Siebel application with Oracle Policy Administration by specifying a symbolic URL.

To integrate the Siebel Application with Oracle Policy Administration

1. In the Siebel application, navigate to Administration - Integration, then WI Symbolic URL List.
2. In the Name field, query for AdvisorsRuleSet
3. If it is not present create a new record with that name.
4. Complete the necessary fields, as shown in the following table.

Table 2–1 Fields of Symbolic URL List

Field	Value
URL	http://<Server>:<Port>/web-determinations/ with OPA server and Port details For example: `http://slc02vaa.us.oracle.com:7065/web-determinations`
Fixup Name	InsideApplet
SSO Disposition	IFrame

5. In the Symbolic URL Argument form (the child applet), complete the necessary fields, as shown in the following table.

Table 2–2 Fields of Symbolic URL Argument Form

Field	Value
Name	IFrame
Required Argument	TRUE
Argument Type	Command
Argument Value	i`Frame width=750 height=500 frameBorder=On` Adjust the Adjust the width and height to suit your requirements for displaying the Advisor popups.
Append Argument	TRUE
Sequence	1

6. Log out of the Siebel application, and then log in again.

Creating the Oracle Policy Automation Rule Base

You must create the Oracle Policy Automation rule bases that are needed to help with purchasing decisions. The rules that are needed depend on your business model and the products you sell.

For more information creating rule bases, see Oracle Policy Automation Connector for Siebel Developer's Guide on Oracle Technology Network.

Verifying the Integration

After you have set up the integration and the rule bases, you can verify the integration.

To verify the integration

1. Navigate to the Sales Order or Quotes Screen, then to the Catalog View and the Get Advice view.
2. Click the Advisors button.

   Note: Right-click this button if you want to open the OPA screen in a different browser window or tab, so you can continue to use the Siebel application without having to close the Advisor.

   What appears depends on how OPA was configured on the server and on what location on the server the URL points at. If you defined multiple of rule bases, it is best to display a list of Advisors, so the user can choose one of the rule bases you created. If you defined only one rule base, it is best to display that rule base directly, without requiring the user to select it from a list.
3. If a list is displayed, click one of the Advisors in the list.

   An Advisor is displayed in the same popup window that the list was displayed in.

Using Oracle Policy Automation to Replace Siebel Advisor

After administrators have set up Oracle Policy Automation as a replacement for Siebel Advisor, users who are using Open UI can access the rule bases to guide purchasing decisions.

The procedure for the user to access the rule base is the same as the procedure that the administrator uses to verify the integration. For more information, see Verifying the Integration.

3

Overview of the Siebel Advisor Interface

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

It includes the following sections:

Opening Advisor

You can open Advisor to look at these views while you read this chapter.

To open Advisor

1. Navigate to the Administration - Product screen.
2. In the link bar click one of the following:
   - Advisor Projects
   - Advisor Pagesets

Advisor Projects View

Use the Advisor Projects view to create and work with your application projects.

A project is the highest level container, which holds all of the information about the project. For example, if you are creating an Advisor application to configure automobiles, the project would contain all the information about all the makes and models of cars you sell.

For more information, see Chapter 5, "Working with Advisor Projects."

Advisor Projects, Pagesets View

This view lists all the pagesets that are part of this project. You work with each individual pageset using the Pagesets screen, described in "Pageset Details Screen" on page 3-2.

Pageset are the basic building blocks of the project. For example, if you are creating an Advisor application to configure automobiles, each pageset could contain all the information about one model of car.

Advisor Projects, Contents List View

Use the Contents List view to create and work with your contents lists. The contents list is a collapsible list used to navigate an application. The end user can click a contents list item in runtime to view the pageset associated with that item.

For example, if you are creating an Advisor application to configure automobiles, contents list would be a list of all the makes and models of cars you sell. The end user could click a make of car to display all the models associated with that make. Then the end user could click a model of car to display the pageset for that model.

For more information, see Chapter 10, "Using Advisor Contents Lists."

Advisor Projects, Project Files View

When you work on a project, Advisor automatically adds files for that project to the Siebel File System. You can use the Project Files view to add or modify the files that belong to a project.

For information on working in this view, see Chapter 15, "Working with Advisor Project Files."

Advisor Projects, Validation Results View

Use the Validation Results view to make sure your project model has no errors. When you validate your project, this view 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 an Advisor Project" on page 5-3.

Advisor Projects, Project Search View

In a large project, use the Search feature to locate the text you are looking for. In the Search view, you can search the project data for text strings 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 an Advisor Project Data Search" on page 5-3.

Pageset Details Screen

Use the Pagesets screen to work with and to specify all the information related to 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.

For example, if you are creating an Advisor application to configure automobiles, each pageset would contain all the information about one model of car. You would use the views of this screen to specify this information.

Feature Tables View

Feature Tables view is used to enter information about the features in this pageset. Each Feature table represents a single feature, and contains all possible values for that feature.

For example, if you are creating an Advisor application to configure automobiles, each pageset would contain all the information about one model of car. One feature table

for the pageset would list all the colors available for this model. A second feature table would list options available for this model.

The Feature Tables tab provides two methods of creating feature tables, which you can view by scrolling down to see the list under the Feature Tables list:

- **Editor.** Allows you to enter the feature table data directly.
- **Designer.** Allows you to add columns to your feature tables, which you can associate with Siebel business component fields or Siebel classification system attributes.

For more information, see "Process of Creating Advisor Feature Tables" on page 7-2.

Configuration Table Editor View

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

For example, if you are creating an Advisor application to configure automobiles, the feature table would specify which options are available with which color car. It would also specify the messages that are displayed if the user selects an invalid combination, such as “The sunroof is not available with a black car.”

The Configuration Tables tab provides two methods of creating configuration tables, which you can view by scrolling down to see the list under the Configuration Tables list:

- **Editor.** Allows you to enter the configuration table data directly.
- **Designer.** Allows you to add columns to your configuration tables, which you can associate with Siebel business component fields or Siebel classification system attributes.

Input UI View

Use the Input UI view to specify 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.

For example, if you are creating an Advisor application to configure automobiles, you can specify that the user uses check boxes to choose options and radio buttons to choose color.

For more information, see "Creating Input UI Controls with Advisor" on page 9-3.

Output UI View

Use the Output UI view 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.

For example, if you are creating an Advisor application to configure automobiles, you might use picture as the output. The application displays a picture of a car with the color and options that the user selected.

For more information, see "Creating Output UI Controls with Advisor" on page 9-6.

Pageset Files View

The Pageset Files view lists the files that belong to a pageset.

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

For example, if you are creating an Advisor application to configure automobiles, and you are displaying pictures of automobiles as output, you would add the pictures of the automobiles here.

For information about these directories and files, see the Browser-Based Application File Reference chapter in *Siebel Advisor API Reference*.

4

Overview of Building Advisor Applications

This chapter gives you an overview of the processes used for building Advisor applications. It includes the following topics:

About Using Siebel Advisor

Use the Advisor 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 Advisor allows customers to base their decision on their needs, without extensive product knowledge.

The Advisor 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 Advisor application opens, displaying a list of questions. For example, if a customer is shopping for a car, Advisor might present the following questions:

- What will be the maximum 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. To allow the user to further refine their configuration, an Advisor session is often integrated with a server-based Configurator application. The Advisor application collects the user's requirements to provide a possible solution. A link to the Configurator application is provided so the user can further configure that solution.

The Advisor links are provided in Siebel Catalog, shopping cart, and quote applications.

For information on creating an Advisor application, see "Roadmap for Creating an Advisor Application" on page 4-2.

Activating Workflows for Advisor

Advisor is based on Siebel Workflows. You must activate these workflows before using the product. For information about activating workflows, see Siebel Business Process Framework: Workflow Guide.

Activate the following workflows:

- eAdvisor Opportunity Confirmation Nav
- eAdvisor Opportunity Creation
- eAdvisor Opportunity Find Product
- eAdvisor Opportunity Integration
- eAdvisor Opportunity Integration (Homepage Nav)
- eAdvisor Opportunity User Profile
- eAdvisor Opportunity User Registration

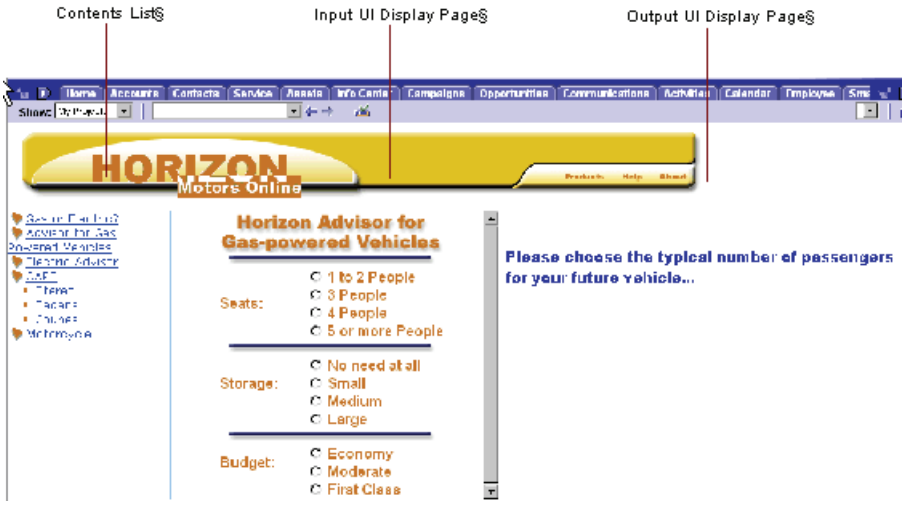
Roadmap for Creating an Advisor Application

You can create an Advisor application for analyzing a user's needs and presenting a solution or configuration.

In Advisor, 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 Advisor application to an Configurator 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 Advisor. For more information, see "Dynamic Defaults in Advisor Applications" on page 12-6.

Figure 4–1 shows an example of an Advisor application at runtime.

Figure 4–1 Siebel Advisor Application at Runtime



To Create an Advisor Application

Before you can create an Advisor application you must activate workflows as described in "Activating Workflows for Advisor" on page 4-1.

To create an Advisor application go through the following process:

1. **Create a project.** For more information, see "Creating an Advisor Project" on page 5-2.
2. **Create one pageset for each possible solution.** For more information, see "Creating Advisor Pagesets" on page 6-1.

3. **Create feature tables.** Customers should arrive at a solution by answering a few questions. About five questions is the best practice for end-user usability. After you design your questions, 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 "Process of Creating Advisor Feature Tables" on page 7-2.

4. **Create configuration tables.** Create a Configuration subtable for each recommendation and reference the subtable from a subtable column in the MAIN Configuration table. In the Configuration subtables:

   a. create an input column for each Feature table (question).

   b. Add an output column to store pageset IDs for the recommended product or solution.

   Enter all valid combinations for the Advisor questions and recommended solutions into the configuration subtables. For information on creating configuration tables, see "Process of Creating Advisor Configuration Tables" on page 8-7. For information on changing default values in controls based on how a user answered previous questions, see "Dynamic Defaults in Advisor Applications" on page 12-6

5. **Create the contents list.** For more information, see "Creating a Contents List Record" on page 10-1.

6. **Add Input UI controls.** The input controls display the questions on the inputs page. For more information, see "About Advisor Input UI Controls" on page 9-2.

7. **Add Output UI controls.** The output controls display information about the recommended solution on the Results page. For more information, see "About Advisor Output UI Controls" on page 9-5.

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

9. **Allow the user to configure the solution (optional).** On the Results page, add a link which the user can click to further configure the recommended solution. 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>
   ```

   For more information, see *Siebel Advisor API Reference*.

10. **Preview the project.** For more information, see "Previewing an Advisor Project" on page 5-4.

11. **Deploy the application.** For more information, see "Deploying an Advisor Project" on page 5-5.

12. **Give users access to the application.** Link the application to quotes, the shopping cart, and other Siebel products, so users can access it. For more information, see Chapter 11, "Working with Deployed Advisor Applications."

Note: For information about displaying Advisor projects in catalogs, see the section about catalogs in *Siebel Order Management Guide*.

5

Working with Advisor Projects

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

This chapter includes the following topics:

About Advisor Projects

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

Advisor provides two views of projects:

- My Projects

  Displays a list of projects for everyone in your group. For example, if your Business 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 are 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 Administration - Application screen in Siebel Business Applications. For more information, see *Applications Administration Guide*.

Creating an Advisor Project

Use the following procedure to create an Advisor project.

To create a project

1. Navigate to Administration - Product screen, then Advisor Projects and My Projects.
2. Click New.
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 Template Name 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 *Siebel 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.

Migrating an Advisor Project

If you are working with a 3.x project, you must migrate it to 4.0 before migrating to 7.x.

To migrate an Advisor project

1. Navigate to Administration-Product, then Advisor Projects screen.
2. Select the project name to migrate.
3. Select Migrate from the list Menu to update 4.0 projects to 7.x.

Copying an Advisor 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 are 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 record appears, displaying the data for the project you are copying.
3. Enter a new project name and directory.

   These values must bye unique throughout the application.

Performing an Advisor Project Data Search

Perform the Advisor 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 an Advisor project Data search using the Project Search tab

1. Navigate to Administration - Project, then Advisor Projects.
2. Select a project.
3. Click the Project Search tab.
4. Enter the text you want to find.
5. Specify what to search: Cell Data, Column Names, or Table Names.

   Select all three to search all three options.
6. From the Project Search form Menu, choose Search.

   All matches for your search appear in the Search Results list.
7. Click a search result to open the view, pageset, and table that contains the search result.

To perform an Advisor project data search using the Query button

1. Navigate to Administration - Project, then Advisor Projects.
2. Select a project.
3. On the tab in which you want to search for data, select Query.

   A Query screen appears that lists all the fields for that tab.
4. Enter data in the field you want to search by.
5. Click Go.

   The results for your search are displayed in the Query Results list.

Validating an Advisor 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 an Advisor project

1. Navigate to Administration - Product, then Advisor Projects.
2. Select the project you want to validate.
3. From the Advisor Projects list Menu, choose Validate.

   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.
4. Click an error to view it.

   When you click an error, Advisor opens the correct screen and tab and selects the record containing the error.
5. 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 an Advisor Project

Both Previewing a Project and Deploying a Project use a non-standard method of changing the parameter. According to Siebel standards, this parameter should be changed using Server Manager, not using the CFG file. This non-standard programming may be changed in a later release of the product. This feature is correctly documented, as it currently stands in the product, but the product does not meet Siebel standards for changing the parameter.

To make sure that the edits you make to the Advisor 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 an Advisor 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. Navigate to Administration - Product, then Advisor Projects.
4. Select the project you want to preview.
5. Validate the project.

   For more information, see "Validating an Advisor Project" on page 5-3.

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

6. In your Siebel application .cfg file, set the WebClientSiteDir parameter 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 are saved to this folder under the ISSRUN\CDAPROJECTS directory. For example, if the WebClientSiteDir parameter 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.

7. If you are deploying in multiple languages, repeat Step 6 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 parameter in the .cfg file to point to the appropriate Siebel Web Engine for that language's deployment.

8. From the Projects list Menu, choose Preview.

   The application appears as it will at runtime.

Deploying an Advisor Project

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

To deploy an Advisor project

1. Navigate to Administration - Project, then Advisor Projects.
2. Open the project you want to deploy.
3. Unlock all pagesets for the project.

   For more information, see Section , "To unlock an Advisor pageset".

4. If you have not already done so, complete Step 6 and Step 7 of "To preview an Advisor project" on page 5-4.
5. From the Project form 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 parameter in your Siebel application .cfg file.

Exporting and Importing an Advisor 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 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 an Advisor project

1. While on the server you are moving the project from, navigate to Administration - Project, then Advisor Projects.
2. In the Advisor Projects list, select a project to export.

3. From the Project list 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.

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 an Advisor project

1. While on the server you are moving the project to, navigate to Administration - Product, then Advisor Projects

2. From the Projects list menu, select Import Project.

   The Import Project dialog appears.

3. Click Browse.

4. In the Choose File dialog, navigate to the project XML file and click Open.

5. Click Import Project.

   The project is added to the My Projects 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.

Advisor 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, Advisor 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 Advisor 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 Advisor user interface.

For more information on setting up view access, see *Siebel Security Guide*.

Team Access

Use Advisor 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.

When you create a project or pageset, you become the owner of the project or pageset and your position becomes the primary position on the team. You can give other people access to a project or pageset by adding them to the team.

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. For more information see "To lock a pageset" on page 6-2.

To add users to an Advisor pageset team

1. Navigate to Administration - Product, then Advisor Pagesets and My Pagesets.
2. Select a pageset.
3. Click the Pageset Team button on the More Info tab.
4. In the picklist, select the employees you want to add to the team.

   The pageset will appear to all members of the team when they select My Pagesets, and they will be able to modify the pageset.

To add users to an Advisor project team

1. Navigate to Administration - Product, then Advisor Projects and My Projects.
2. Select a project.
3. Click the Project Team button on the More Info tab.
4. In the picklist, select the employees you want to add to the team.

   The project will appear to all members of the team when they select My Projects, and they will be able to modify the project.

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

1. Navigate to Administration - Product, then Advisor Projects.
2. From the Show drop-down list, select My Projects.

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

6

Working with Advisor Pagesets

This chapter discusses Advisor pagesets. It includes the following sections:

About Advisor 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 Advisor API Reference*.

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

Creating Advisor Pagesets

Use the following procedure to create a pageset.

To create an Advisor pageset

1. Navigate to Administration - Product, then Advisor Pagesets.

2. In the Advisor Pagesets list, click New.

   A New Pageset record appears.

3. Complete the fields as described in the following table.

Field	Description
Name	Enter a name that describes the pageset, for example, ACME Sedans.
Pageset ID	Enter a pageset ID, for example, sedans. All browser-based code uses the pageset ID to reference the pageset.
Pageset	Click in the Pageset field 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.
Display Category	Click in the Display Category field to open a dialog in which you can select 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.
Display Name	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.
Display Description	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.
Project	Click the Project field to open a dialog and select the project the pageset is associated with. **Note:** Each pageset is physically associated with one project, but can be referenced by other projects.
Notes	Optionally, enter notes.

Locking Advisor 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 "Advisor Projects in a Team Environment" on page 5-6.

To lock a pageset

1. Navigate to Administration - Product, then Advisor Pagesets and My Pagesets.
2. In the My Pagesets list, select the pageset you want to lock.
3. In the Locked field, check Locked.

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

To unlock an Advisor pageset

1. Navigate to Administration - Product, then Advisor Pagesets and My Pagesets.
2. In the My Pagesets list, select the pageset you want to unlock.
3. Remove the check from the Locked field.

   Other users can now edit the pageset.

Specifying a Pageset to Configure a Product

In the Advisor Pagesets list, you can associate a pageset with a product. Then, when a user selects the product in a quote or catalog and clicks Customize, the associated pageset appears.

To associate an Advisor pageset with a product

1. Navigate to the Administration - Product, then Advisor Pagesets list.
2. In the Pagesets list, select the pageset you want to associate with a project.
3. Click the Project field.

   A Projects list appears.
4. Select a project.

   A pageset can only be associated with one project.

Managing the Display of Pagesets in Advisor

When a user navigates to Administration - Product, then Advisor Pagesets, the Advisor pagesets list appears. 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 *Siebel 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 or migrate projects from previous versions of the Siebel product, *In-Development* is assigned as the value for Display Category, Display Name, and Display Description.

Copying Advisor Pagesets

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 is copied along with the pageset.

To copy a pageset

1. Navigate to Administration - Product, then Advisor Pagesets.
2. Select and lock a pageset.

   For more information, see "To lock a pageset" on page 6-2.
3. From the list menu, choose Copy Record.

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

4. 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.

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. Navigate to Administration - Product, then Advisor Pagesets and Pageset Files.
2. Select a pageset.
3. Make a copy of the file in the Project Files view.
4. In the Pageset Files view, click New.
5. In the Files dialog box, select the file.
6. 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 are copied with the pageset.

Validating Advisor Pagesets

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. Navigate to Administration - Product, then Advisor Pagesets.
2. In the Pageset list, select the pageset you want to validate.
3. 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.

4. Click an error to view it.

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

5. Fix the errors and return to the Validation Results tab to validate the pageset again.

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

Performing a Pageset Data Search

You can perform Advisor data searches on a 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 pageset data search using the Pageset Search tab

1. Navigate to Administration - Product, then Advisor Pagesets.
2. Select the pageset you want to search.
3. On the Pageset Search tab, enter the text you want to find.
4. Specify what to search: Cell Data, Column Names, Table Names.

   Select all three to search all three options.
5. In the Pageset Search form Menu, select Search.

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

To perform a pageset data search using the Query button

1. Navigate to Administration - Product, then Advisor Pagesets.
2. Select the pageset you want to search.
3. 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.
4. Enter data in the field you want to search by.
5. Click Go.

   The Query returns the results for your search.

7

Working with Advisor Feature Tables

This chapter discusses Advisor feature tables. It includes the following sections:

About Advisor 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.

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 "Process of Creating Advisor Configuration Tables" on page 8-7.

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 "Trigger and Target Feature Tables for Advisor Applications" on page 12-1.

Process of Creating Advisor Feature Tables

Creating an Advisor feature table requires you to complete the following steps:

1. Analyzing the Problem Before Creating Feature Tables
2. Creating an Advisor Feature Table
3. Designing an Advisor Feature Table
4. Entering Data in the Advisor Feature Table

Analyzing the Problem Before Creating Feature Tables

You analyze the problem by dividing your data into features and selecting default values.

This task is a step in Process of Creating Advisor Feature Tables.

First, you must decide which features the user can select.

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 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 Advisor 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.

Next, you must determine the possible values for each feature. For example, for the exterior color feature, determine what colors are offered. You populate the feature table with these values, for example Red, Black, and Silver.

Next, 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 12-2.

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 use this information when creating your feature tables in the next steps.

Creating an Advisor Feature Table

To create a feature table, you create a record in the feature tables list with basic information about the table.

This task is a step in "Process of Creating Advisor Feature Tables" on page 7-2.

To create an Advisor feature table

1. Navigate to Administration - Product, then Advisor Pagesets and My Pagesets.
2. Drill down on the Name field of a pageset.
3. Select the Feature Tables tab and, from the Menu, choose New Record.

   A new Feature Table record appears.
4. Enter a name, without spaces, for the feature table.

   The name automatically converts 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 Advisor Feature Tables" on page 7-6.
   - Standard
   - Target

     For more information, see "Creating Trigger and Target Feature Tables" on page 12-2.
   - Trigger

     For more information, see "Creating Trigger and Target Feature Tables" on page 12-2.

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.

Designing an Advisor Feature Table

In the Designer view, you design the feature table in which you enter your feature data.

This task is a step in "Process of Creating Advisor Feature Tables" on page 7-2.

Each feature table must contain at least three columns: CODE, DESC, and DEFAULT, which are described in Table 7–1. When you create a feature table in Advisor, these columns are automatically created.

Table 7–1 Required Columns for Feature Tables

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.

To design an Advisor feature table

1. Navigate to Administration - Product, then Advisor Pagesets and My Pagesets.
2. Select a pageset.
3. From the Feature Tables tab, select the Designer tab.

   The Feature Table designer opens.
4. Complete the fields for the new record in the Feature Table Designer.

   The following table describes the columns in the Feature Table Designer.

Column	Description	Optional
Sequence	The sequence numbers determine the order in which the feature columns appear in the Feature editor. These numbers are automatically generated, but you can overwrite them.	Yes
Column Name	Enter a column name for each feature. For example, for the CARS feature table, you might enter columns for MODEL, COLOR, and PRICE.	No

Column	Description	Optional
Business Component	This column maps to the selected column (business component) in the Siebel database. For more information on using business components, see *Using Siebel Tools*. 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. **Note:** If you select a business component and field, you cannot select a class and attribute. These options are mutually exclusive.	No
Field Name	If you selected a business component, you must also select a Field Name. Select a field name from the business component you selected.	No
Shared	Check the Shared field to reference one row ID in a business component. 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.	Yes
Class	This field is optional. This field maps to the selected class in the Siebel database. For more information on classes and attributes, see *Siebel Product Administration Guide*. **Note:** If you select a class and attribute, you cannot select a business component and field. These options are mutually exclusive.	Yes
Attribute	If you selected a class, select an attribute for the class.	Yes
Target Table	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 12-2.	No
Notes	Enter any notes.	Yes

Entering Data in the Advisor Feature Table

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.

This task is a step in "Process of Creating Advisor Feature Tables" on page 7-2.

The Row type column provides a picklist for Target tables. For information about creating Trigger and Target feature tables, see "Creating Trigger and Target Feature Tables" on page 12-2. 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 "Process of Creating Advisor Feature Tables" on page 7-2.

To enter data in an Advisor pageset feature table

1. Navigate to Administration - Product, then Advisor Pagesets and Feature Tables.
2. Select a Feature Table.

3. Select the Editor tab.
4. In the Feature Table editor, enter the sequence of the features as they 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.
5. 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 are 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.
6. Enter a full description of the feature.
7. Fill in information for any other columns you have added.
8. Select the value you want to appear by default in the associated UI control and enter DEFAULT in the DEFAULT column.
9. 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 Pricing Information in Advisor Applications" on page 14-14.

Creating Linked Advisor 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 Advisor pageset feature table

1. Navigate to Administration - Product, then Advisor Pagesets and My Pagesets.
2. In the My Pagesets list, select the pageset for which you want to create a linked feature table.
3. Select the Feature Tables tab and click New.

   A new Feature Table record appears.
4. Enter a name, without spaces, for the feature table.
5. Click the Linked to field.
6. Click the button in the Linked to field.

   A dialog box appears.
7. Select the table whose data you want to use.
8. Click OK.

Managing Advisor Feature Table Columns

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.

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. Navigate to Administration - Product, then Advisor Pagesets and 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.

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 this section.

To design the Feature table

1. Navigate to Administration - Product, then Advisor Pagesets.
2. In the Pagesets screen, select a pageset.
3. In the Feature Tables tab, select a Feature table.
4. Select the Designer tab.
5. In the Designer applet, select the row you want to edit.
6. Make changes directly in the row.

To edit rows in the Feature table

1. Navigate to Administration - Product, then Advisor Pagesets.
2. In the Pagesets screen, select a pageset.
3. In the Feature Tables tab, select a Feature table.
4. Select the Editor tab.
5. Select the feature row you want to edit in the Editor applet.
6. Make changes directly in the row.

8

Working with Advisor Configuration Tables

This chapter discusses Advisor Configuration Tables. It includes the following sections:

About Advisor 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. 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 Advisor 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.

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

The Configuration Matching Process

When users of Advisor applications 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 for Advisor Applications" on page 12-12.

Configuration Column Types

Like Feature tables, Advisor 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 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 for Advisor Configuration Tables" on page 8-7 for more information.

In Figure 8–1, the highlighted row shows that the Model GXE is available with a red exterior and interior.

Figure 8–1 Configuration Editor Displaying Input Columns

Editor

New | Save | 1 - 19 of 19

Row Type	Sequence	MODEL(1)	COLOR_EXT(1)	COLOR_INT(1)	RULE(0)
DATA	10	XE	(=BE,WH,BK,GR)	(=BE,WH,BK,GR)	
DATA	20	LE	(=BE,WH,BK,GR,BL	(=BE,WH,BK,GR)	
DATA	30	LE	BL	BL	
DATA	40	GLE	(=BE,WH,BK,GR,BL	(=BE,WH,BK,GR,BL	
DATA	50	GLE	BL	BL	
DATA	60	GXE	(=BE,WH,BK,GR,BL	(=BE,WH,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 <font color="blue">...
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 <font color="blue">...
EXCEPTION	90	GXE	(!=BL)	BL	Sorry, the <font color="blue">...
EXCEPTION	100	GXE	(!=RE)	RE	Sorry, the <font color="red">r...
EXCEPTION	110				TEST for MODEL_COLOR

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. Advisor 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 8-2.

Cell Functions for Advisor Configuration Tables

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

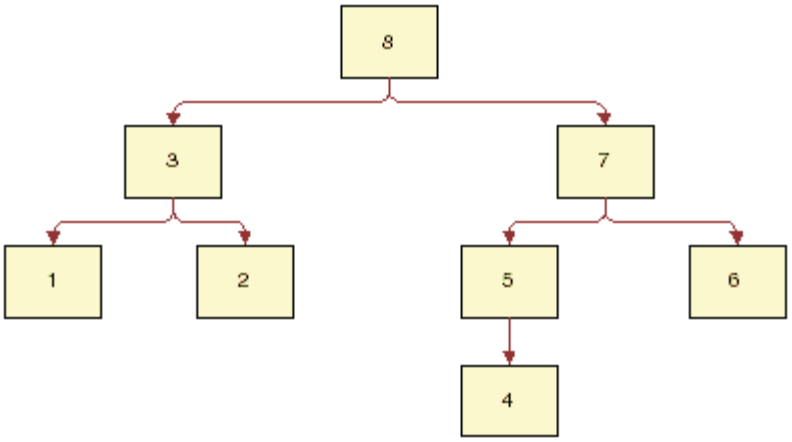
Use standard javascript syntax to perform calculations in a table cell.

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 8–2.

Figure 8–2 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.

Cell Function Example

In this example, you 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 an Advisor pageset cell function

1. Navigate to Administration - Product, then the Advisor Pagesets and My Pagesets view.
2. Select a pageset.
3. On the Configuration Tables tab, select the MAIN Configuration table.
4. Select New Record from the Designer tab menu.

   A new record appears.
5. Name the column SPEND_MAX and enter Output for the type of column.
6. In Edit view, enter the following formula into all valid configuration rows of the new SPEND_MAX(Output) column: ((SALARY.AMT*.75*.05*5)+CASH.AMT).

Range Functions for Advisor Configuration Tables

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 causes an error when generating HTML files in Advisor. Instead, use the syntax (=bm1,bm2,bm3) or (=*).

Table 8–1shows examples of range functions:

Table 8–1 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

Process of Creating Advisor Configuration Tables

To create a Configuration table, go through the following steps:

Opening the Advisor Configuration Table

Designing the Advisor Configuration Table

Entering Data in the Advisor Configuration Table

Opening the Advisor Configuration Table

Begin by opening the configuration table you are working on. You can do this by opening the Main configuration table, or by creating a configuration subtable.

The Main Configuration table is included in your project by default.

For complex projects, you can create configuration subtables 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.

This task is a step in "Process of Creating Advisor Configuration Tables" on page 8-7.

To create a Configuration subtable

1. Navigate to Administration - Product, then the Advisor Pagesets and My Pagesets view.
2. Select a pageset.
3. On the Configuration Tables tab menu, select New Record.

   The new Configuration Table form appears.
4. Enter a name for the Configuration table, and any notes.

5. On the Designer tab, click New.
6. Fill in the fields as described in the following table.

Field	Description
Sequence	Enter a sequence number.
Column Name	Enter the name of the Configuration subtable.
Column Type	From the Column Type drop-down list, select Sub Table.

7. 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 Advisor 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.

This task is a step in "Process of Creating Advisor Configuration Tables" on page 8-7.

To design the Advisor pageset Configuration table

1. Open the Configuration table you want to design and select the Designer tab.
2. In the Designer form, 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 Advisor 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 8-3.

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, see *Using Siebel Tools*.

7. If you selected a business component in 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, see *Using Siebel Tools*.

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 Advisor 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.

This task is a step in "Process of Creating Advisor Configuration Tables" on page 8-7.

To enter data in the Configuration table

1. In the Configuration Tables view, 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 inStep 3, create a guidance message for cross and up-sell messages or Advisor questions. For more information, see "Creating Cross-Sell and Up-Sell Messages in Advisor" on page 8-11 or "Roadmap for Creating an Advisor Application" on page 4-2.
5. Enter the sequence number. This is an optional step.

   This determines the order in which the new row appears 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 Pricing Information in Advisor Applications" on page 14-14.

Creating Exception Messages in Advisor

Create exception messages to help customers purchase the right product or service. If a customer chooses an invalid combination, the application displays 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 8–2 lists some examples of informative exception messages.

Table 8–2 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 in Advisor" on page 8-11.

To create an exception message

1. Navigate to Administration - Product, then Configuration Tables, and select a configuration table.
2. On the editor tab, 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 creates 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 in Advisor

An Advisor 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, Advisor 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 add the message to an output non-RULE column in the appropriate configuration rows.

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

  You add a LINK control to your Output UI. You edit the file that displays your UI to provide a location where the message appears 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 Configurator, see "About Referencing Other Siebel Data from Advisor Applications" on page 14-1

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

1. Navigate to Administration - Product, then Configuration Tables, and select a configuration table.
2. On the designer tab, add two new rows above the RULE row.
3. Name one of the rows UPSELL_LINK.

   This column requires the pageset ID of the linked pageset.
4. Name the other row “UPSELL_MSG.”

   This column includes the up-sell message.
5. In the Column Type column, click the button to open a picklist.
6. In the picklist, select Output Column and click OK.
7. Select the Configuration table you want to add the up-sell message to and select the Editor tab.
8. 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.
9. 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. Navigate to Administration - Product, then Configuration Tables, and select a configuration table.
2. In the Output UI view, create a new LINK control.
3. Choose a table and column.
4. 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.

9

Building the UI with Advisor

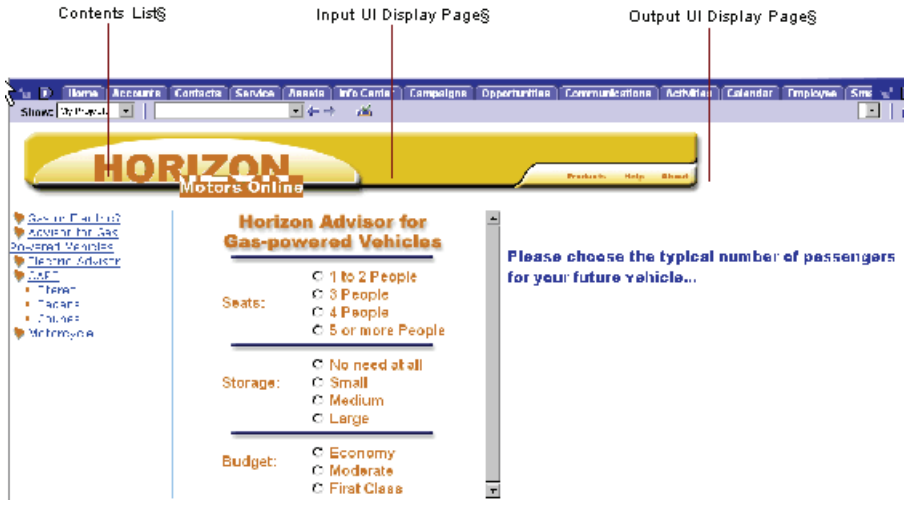
This chapter discusses how to use Advisor to build the user interface. It includes the following sections:

About Building a UI with Advisor

Advisor allows you to build a UI layout that consists of three areas (see Figure 9–1, "The UI Layout of a Browser-Based Application".):

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

Figure 9–1 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.

For information on how to create a contents list, see Chapter 10, "Using Advisor Contents Lists."

Input UI Display Page

Use the Input UI tab in Advisor 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.

For information on how to create the Input UI display page, see "Generating Your Input and Output UI Display Page with Advisor" on page 9-8.

Output UI Display Page

Use the Output UI tab in Advisor 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:

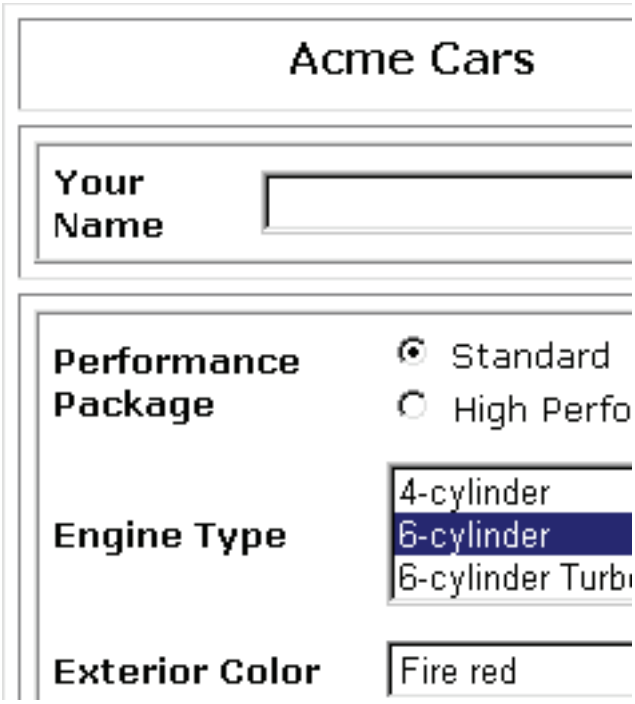
- Output UI controls, such as images representing a selected choice, display the recommendations to a valid feature selection made with an input UI control.
- Exception messages, usually text warnings, guide users to a valid product configuration when they choose feature selections that do not match 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 in Advisor" on page 8-11.

For information on creating an Output UI Display page, see "Generating Your Input and Output UI Display Page with Advisor" on page 9-8.

About Advisor Input UI Controls

An Input UI control appears in the Input UI display page and allows users to select features. Figure 9–2 displays the Input UI controls.

Figure 9–2 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 Advisor functions that are generated from the information you enter in Advisor 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 with Advisor

Use the following procedure to create input UI controls.

To create input UI controls with Advisor

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

   For more information, see "Process of Creating Advisor Feature Tables" on page 7-2.

2. In the Pageset screen, select a pageset.

3. In the Input UI tab, click New.

   A new row appears.

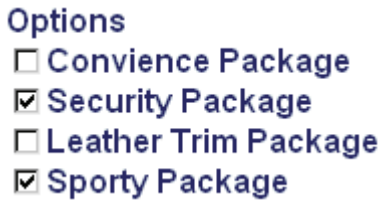
4. Enter the sequence number in which the control is laid out on the page.

   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 is used as the label for the check box in the UI.

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



   - Get Text

     Allows a user to enter a text string.

   - List Box

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

   - Radio

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

   - MAP

     Allows a user to assign different links to different parts of the same image using HTML.

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 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 <IMG SRC> 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.

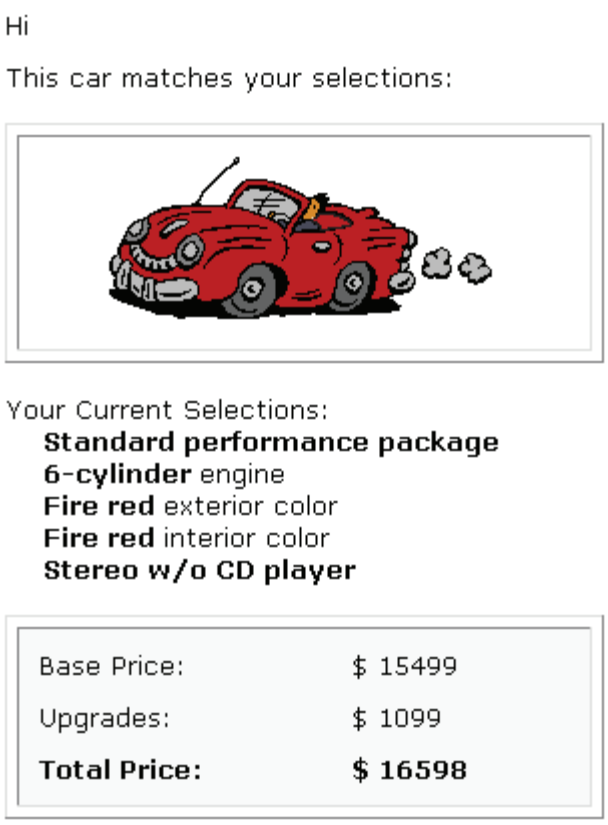
    – Map Coordinates populates the parameter for the COORDS attribute in the <AREA> HTML tag, for example, "9,67,37,76,30".
12. Click Generate to add the new UI controls to the Input UI display page.

Note: It is recommended that you build your UI using the Input UI and Output UI tabs in Advisor. 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 whenever you regenerate the UI. Be sure to write any custom code outside of this area.

About Advisor 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 9–3 is an example of output UI controls.

Figure 9–3 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 with Advisor

Use the following procedure to create output UI controls.

To create output UI controls with Advisor

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

   For more information, see "Process of Creating Advisor Feature Tables" on page 7-2 and "Process of Creating Advisor Configuration Tables" on page 8-7.

2. In the My Pagesets list, 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 Configurator.

- 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.

- GOTOVIEW

  Takes the user to the Siebel View identified in the Siebel Column in the Output UI administration list.

- 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.

- SUBCONFIG_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 the button in the Table field to open a picklist and select the table whose values populate the control.

8. Click the button in the Column field to open a picklist and select the column, from the table you selected in Step 7, whose values populate the control.

   For example, if you are creating a LINK UI control that displays 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 appears as a link. For example: Click here to see this product.

13. 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 Advisor. 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.

Generating Your Input and Output UI Display Page with Advisor

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

To generate your UI display pages with Advisor

1. Navigate to Administration - Product, then Advisor Pagesets and 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.

   For more information, see "To create input UI controls with Advisor" on page 9-3 or "To create output UI controls with Advisor" on page 9-6.

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 Advisor. 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.

10

Using Advisor Contents Lists

This chapter discusses the Advisor contents list. It includes the following sections:

About Advisor Contents Lists

The contents list is a collapsible list that allows the user to navigate an application. Clicking an item in the contents list displays the page associated with that item.

By default, the contents list appears to the left of the display pages. You can edit the position of the contents list in an HTML editor.

For more information on how the content list relates to the other elements of the UI, see Chapter 9, "Building the UI with Advisor."

This chapter discusses how to create the contents list. It also discusses special considerations for working with the contents lists of Advisor applications.

Process of Creating an Advisor Contents List

To create a new contents list, go through the following steps:

Creating a Contents List Record

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

This task is a step in "Process of Creating an Advisor Contents List" on page 10-1.

To create a contents list with Advisor

1. Navigate to Administration - Product, then Advisor Projects and My Projects.
2. Select the project for which you would like to create a contents list.
3. On the Contents Lists tab, choose New.

   A new Contents List form appears.
4. 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.

5. Access the new contents list file by using the ShowContentsList function.

   See *Siebel Advisor API Reference* for more information.

Creating Contents List Items

Use the following procedure to create contents list items.

This task is a step in "Process of Creating an Advisor Contents List" on page 10-1.

To enter contents list items in Advisor

1. Navigate to Administration - Product, then Advisor Projects and My Projects.
2. Select a project.
3. On the Contents Lists tab, select the contents list and, in the Editor form, click New.

   A new row appears.
4. Enter the sequence in which the item appears in the contents list.
5. 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.
6. 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.
7. 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.
8. 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.

Contents Lists for Advisor

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 Chapter 13, "Customizing the UI of Advisor Pages." 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 manage the display of the contents list in the Application UI Definition file or a Pageset UI Definition file. For more information, see Chapter 13, "Customizing the UI of Advisor Pages." 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 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 Advisor. Whether or not you use them, do not delete the default project and its contents list.

11

Working with Deployed Advisor Applications

This chapter describes modifications you can make to your deployed application. It covers the following topics:

- Running Advisor Applications in Stand-Alone or Standard Mode
- Calling Your Advisor Application from Another Siebel Application
- Synchronization Setup for Advisor
- About Working with Advisor Applications in Mobile Client Mode

Running Advisor Applications 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 = ISSStr+".displayArea"(Set displayArea to your
particular display area.)
```

2. Connect to your application using showCDA.

   For more information, see showCDA in *Siebel Advisor API Reference*.

Calling Your Advisor Application from Another Siebel Application

A Need Advice button that links to an Advisor 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 Advisor. The Need Advice button can also link to a specified contents list.

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" on page 11-2.
- "Invoking the ShowCDA Method from a Button" on page 11-2.
- "Passing in Parameters When Invoking the ShowCDA Method" on page 11-3.

Referencing Pagesets from Customizable Products

Customizable Products can be configured at runtime by the server-based Configurator and Advisor. To open Advisor for a product, you need to associate a pageset with the product in the Administration - Product screen. After you make this association, when a user clicks the Customize button for a product, the associated pageset opens in Advisor.

To create a customizable product and associate it with a pageset

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, see *Siebel Product Administration Guide*.

2. Release the customizable product.
3. In Advisor, create a pageset.

   For more information, see "To create an Advisor pageset" on page 6-1."

4. Associate the pageset to the product.

   For more information, see "To associate an Advisor pageset with a product" on page 6-3.

Invoking the ShowCDA Method from a Button

Use the following procedure to invoke the ShowCDA method from a button.

To call your Advisor application from another Siebel application

1. In Siebel Tools, create a button control on your applet.

See *Using Siebel Tools* for information.

2. For the Method property of the button, invoke the ShowCDA method.

   Use the syntax:

   ```
   ShowCDA('ProjectDirectory|Pageset ID')
   ```

   Note: This step works 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, see *Siebel Advisor 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.

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 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 11–1 Creating Three Buttons that Display Different Applets

This example assumes you want to create three 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 | ShowCDA1 | Parameter 1	QuoteId | Field | Id
Method | ShowCDA1 | Parameter 2	Projectlocation | Value | project2
Method | ShowCDA1 | Parameter 3	Pageset | Value | pageset2
Method | ShowCDA1 | Parameter 4	Dyndefs | Value | dyndefs2
Method | ShowCDA1 | Parameter 5	End

Button 3

Name	Value
Method | ShowCDA2 | Parameter 1	QuoteId | Field | Id
Method | ShowCDA2 | Parameter 2	Projectlocation | Value | project3
Method | ShowCDA2 | Parameter 3	Pageset | Value | pageset3
Method | ShowCDA2 | Parameter 4	Dyndefs | Value | dyndefs3
Method | ShowCDA2 | Parameter 5	End

In this example, project1, project2, and project3 are the values passed to Advisor. 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 is 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 Advisor 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 Table 11–1 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.

Table 11–1 Example Syntax

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 for Advisor

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.

This section covers the following topics:

- "Modifying the Siebel Synchronize Database Behavior" on page 11-5.
- "Using the Synchronize CDA Projects Screen" on page 11-7.
- "Modifying the Project Synchronization Behavior" on page 11-7.

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 11-7.

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 in 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 then ISSCDA Sync Projects and 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

> **Note:** If you chose to have synchronization occur automatically, you do not need to expose this screen. If you chose the other options, you need to expose this screen; users go to this screen to request file attachments.

To expose the Synchronize CDA Project screen

1. Start the Siebel application.

2. Log in as SADMIN on the server.
3. Select Application Administration, then Views.
4. Add the "ISSCDA Synchronize View" to the existing list of views and select Local.
5. Select Application Administration, then Responsibilities.
6. Add the "ISSCDA Synchronize View" to the desired responsibility.

To compile the SRF file

1. Open Siebel Tools.
2. Go to Tools, then 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 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, then Synchronize and Database to synchronize the database.
6. If your browser is open when you synchronize, clear the cache to see the latest data.

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 are 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 with Advisor Applications 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 need to manually delete files that have been removed from projects. You 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.

12

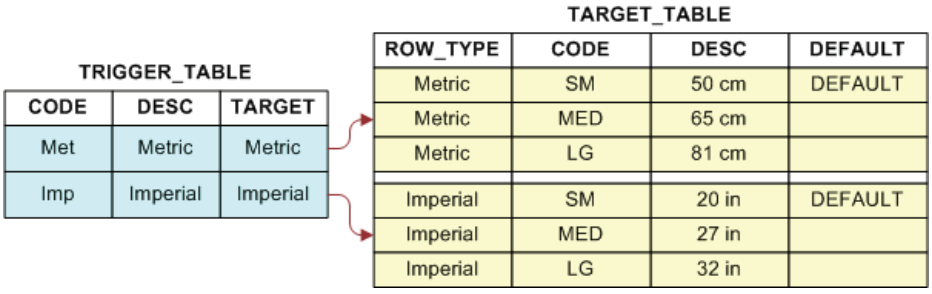
Advanced Modeling for Advisor

This chapter describes advanced techniques for working with your Advisor data model. It includes the following sections:

Trigger and Target Feature Tables for Advisor Applications

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 12–1.

Figure 12–1 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 appears 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

To create trigger and target feature tables for Advisor applications, perform the following tasks:

- To create a target feature table
- To create new Row Types for the target feature table
- To edit rows in the Feature table

  See "Editing a Feature Table" on page 7-7 for instructions on completing this step.
- To create a trigger feature table
- To tie the Trigger and Target Feature tables to UI controls

Each of these steps is explained in the following procedures.

To create a target feature table

1. Navigate to Administration - Product, then Advisor Pagesets and My Pagesets.
2. Select the pageset for which you want to create a target feature table, and click the Feature Tables view tab.
3. Choose New Record from the Feature table list menu.

   A new Feature Table window appears.
4. Enter a name, without spaces, for the Feature table.
5. From the Table Type drop-down list, select Target.

To create new Row Types for the target feature table

1. Navigate to Administration - Product, then Advisor Pagesets and My Pagesets.

2. Select a pageset, and click the Feature Tables view tab.
3. Select a Target table.
4. Click the Editor view tab.
5. In the Editor view, click New.

   A new row appears in the editor.
6. Click the Row Type picklist icon to open the Feature Table Rowset picklist.

The four types of feature tables are described in Chapter 7, "Working with Advisor Feature Tables."

To create a trigger feature table

1. Navigate to Administration - Product, then Advisor Pagesets and My Pagesets.
2. Select a pageset, and click the Feature Tables view tab.
3. From the Feature Table list menu, choose New Record.

   A new Feature Table record appears.
4. Complete the following required fields.

Field Name	Description
Name	Enter a name, without spaces, for the Feature table.
Table Type	From the Table Type drop-down list, select Trigger.

5. Select the Designer tab, and choose New Record from the list menu.

   A new record appears.
6. Complete the following fields.

Field Name	Description
Column Name	This column links to the Target table.
Target Table	Open the Target Table picklist and select the Target table you created in the previous procedure.

7. Select the Editor tab and follow the steps in "To enter data in an Advisor pageset feature table" on page 7-5.

> **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 Advisor Input UI Controls" on page 9-2 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.

Example of Creating Trigger and Target Tables

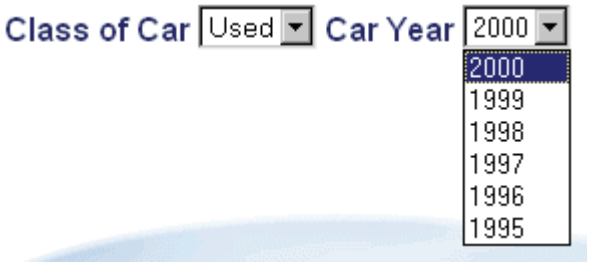
In this example, you add Trigger and Target Feature tables that populate a drop-down list with appropriate year values, based on whether a user wants to purchase a new or used car.

At runtime, if users of the Advisor application select New for Class of Car, they have one option in the Car Year drop-down list, see Figure 12–2.

Figure 12–2 Single-Option List

If users select Used, they can choose from a list of year values, see Figure 12–3.

Figure 12–3 List of Year Values



Creating the YEAR Target Table

Creating the YEAR target table requires three steps:

1. Creating a target table named YEAR.
2. Creating row types for the table.
3. Populating the target table with values.

Each of these steps is explained in the following procedures.

To create the YEAR target table

1. Navigate to Administration - Product, then Advisor Pagesets and My Pagesets.
2. Select a pageset, and click the Feature Tables tab.
3. Choose New Record from the list menu.

   A new Feature Table record appears.
4. Enter YEAR in the Name field.
5. From the Table Type drop-down list, select Target.

To create the row types for the YEAR Target table

1. Navigate to Administration - Product, then Advisor Pagesets and My Pagesets.
2. Select a pageset, and click the Feature Tables view tab.
3. In the Feature Tables tab, select the YEAR Target table.

4. On the Editor tab, click New.

   A new row appears in the editor.
5. Click the Row Type picklist icon to open the Feature Table Rowset picklist.
6. Click New.

   The Feature Table Rowset dialog appears.
7. In the Name text box, enter YEAR_NEW.
8. Click Save.
9. Repeat Steps 4 through 6 and create a second new rowset named YEAR_USED.

To populate the YEAR target table

1. Navigate to Administration - Product, then Advisor Pagesets and My Pagesets.
2. Select a pageset, and click the Feature Tables view tab.
3. Select the YEAR target table from the Feature Tables list.
4. On the Editor tab, create a row with values for the YEAR_NEW rowset.
5. 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. Navigate to Administration - Product, then Advisor Pagesets and My Pagesets.
2. Select a pageset, and click the Feature Tables view tab.
3. From the Feature Table applet menu, choose New.

   A new Feature Table window appears.
4. In the Name text box, enter CLASS.

   From the Type drop-down list, select Trigger.
5. From the Designer tab form menu, choose New Record.
6. In the Column Name text box, enter YEAR.

   In the Designer tab, the Target Table button is enabled.
7. Click the Target Table button to open the Target Table picklist.
8. Select the YEAR target table and click OK.
9. Follow the steps in "To enter data in an Advisor pageset feature table" on page 7-5.

> **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. Navigate to Administration - Product, then Advisor Pagesets and My Pagesets.
2. Select a pageset.
3. In the Input UI tab, click New.
4. Enter values in the new row.

   For more information, see Section , “To create input UI controls with Advisor”.
5. Click New again.
6. Enter values for the second row.

Additional Trigger Capabilities

As mentioned in "Trigger and Target Feature Tables" on page 7-2 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 in Advisor Applications

When Advisor 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 Advisor questions. In the car example, if the user indicates interest 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 for Advisor Applications

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. See the LoadPageset and LoadPagesetWithDynDefObj sections in *Siebel Advisor API Reference*.

To create dynamic defaults

1. Navigate to Administration - Product, then Advisor Pagesets and Configuration Tables.
2. In a Configuration table, create a DYNDEFS column.

   In this example, a Pagesetname column is not necessary because the PagesetName will be part of the string in the DYNDEFS column.
3. In the DYNDEFS cells, enter the pagesetID and desired Feature table values in the format:

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

4. 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 appears as the link.

Notice the single quote, comma, single quote in-between *PagesetID* and *1stFTName* in Step 7. Because the wrapper puts 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,2ndFTNam
e=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 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 Defaults in Advisor Applications

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

The following procedure requires a bit more definition.

To create the dynamic defaults example

1. Add a drop-down list UI control to the needs analysis page.
2. For more information, see "Creating Input UI Controls with Advisor" on page 9-3.
3. Create a Feature table to populate the drop-down list with the values:
   – Laptops
   – Desktops

– Work stations

For more information, see "To create an Advisor feature table" on page 7-3.

4. In a Configuration table that references the Feature table you created in Step 3, add an output column named REC_PROD to store the pageset ID of the recommended product.
5. In the same Configuration table, add an output column named DYNDEFS to store the defaults to change on the linked-to display page.
6. 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 XR50.

REC_PROD(0)	DYNDEFS(0)
laptops	MODEL=XR50,PROC=366,MEM=32
laptops	MODEL=XR80,PROC=400,MEM=64
desktops	MODEL=XP50,PROC=366,MEM=32
workstations	MODEL=XP80,PROC=366,MEM=64

7. 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 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,"DYNDEFS"));
</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, see the Pageset Functions chapter in *Siebel Advisor API Reference*.

The Result of the Example

If the user's selections match the recommended solution described in the Configuration table, a link appears to further configure the computer. When a shopper clicks the link, a display page for the pageset recommended in the REC_PROD column appears. Also, the MODEL UI control is set to the CODE value indicated in the DYNDEFS column. For example, the default MODEL description is XR50, but if the user is recommended the XR80, the UI control displays XR80.

Working with Subconfiguration in Advisor

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 of Subconfiguration in Advisor Applications

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 Advisor 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, see the Pageset Functions chapter in *Siebel Advisor API Reference*.
- Use references to Feature tables.

About Referencing Feature Tables in Subconfigured Data Models

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 12–1 Referencing a Pageset in a Subconfigured Data Model

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 Subconfigured Data Models

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 Advisor

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 12–1 for sample values.

Table 12–1 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 load into the CHILD. A CHILD pageset only gets 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 retains the user's selection, and therefore does not 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 12–2 and Table 12–3:

Table 12–2 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 12–3 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 is 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) matches 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 in Siebel Tools can 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 does not work. The cell function calculates before the output variable is set, causing the application to be out of sync.

Performance Considerations for Subconfigured Data Models

When designing an Advisor 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.

Duplicate Configuration Column Names in Subconfigured Data Models

If two Configuration column names have the same name, the latter column name in the evaluation cycle overwrites 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 is evaluated, each of the options should have the same outputs, and you need to duplicate column names. Only one of the columns is evaluated for any particular set of selections.

Creating Javascript Conditional Statements for Advisor Applications

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 8-2.

To create an OL_CONDITIONS Configuration table

1. Create a Configuration table.

   See "Process of Creating Advisor Configuration Tables" on page 8-7 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 12–2 JavaScript Conditional Statement

Table 12–4 shows an example of a JavaScript conditional statement in the TEST column and the message in the RULE column that appears if the statement is true.

Table 12–4 JavaScript Conditional Statement Example

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

13

Customizing the UI of Advisor Pages

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

This chapter covers the following topics:

About Customizing Your UI

Use a text or HTML editor to add Advisor 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 Advisor input and output UI controls after the customization.

For more information about the Advisor functions, see *Siebel Advisor 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. Navigate to Administration - Product, then Advisor Projects and My Projects.
2. Select a project.

3. 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.
4. From the Internet browser File menu, choose Save As and save the file to your hard drive.
5. Edit the file you saved on the hard drive in Step 4 using an HTML editor.

   If you plan to continue using Advisor input and output UI controls, add your code outside of the section enclosed by the comments BEGIN GENERATE HTML and END GENERATE HTML.
6. Save the file.
7. In the Siebel application, in the Advisor Project Files view, delete the previous version of the file.
8. Click new record and add the modified version of the file.

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 Advisor 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, Advisor associates the default Pageset UI Definition file, \pg\oc_default_ui.htm, with all the pagesets in the Advisor 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 Advisor 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. Navigate to Administration - Product, then Advisor Projects and My Projects.
2. Select a project.
3. In the Project Files tab, navigate to the UI directory and select ol_ui.htm.
4. Edit and save the file.

   For more information, see "Editing the Project UI Files" on page 13-1.

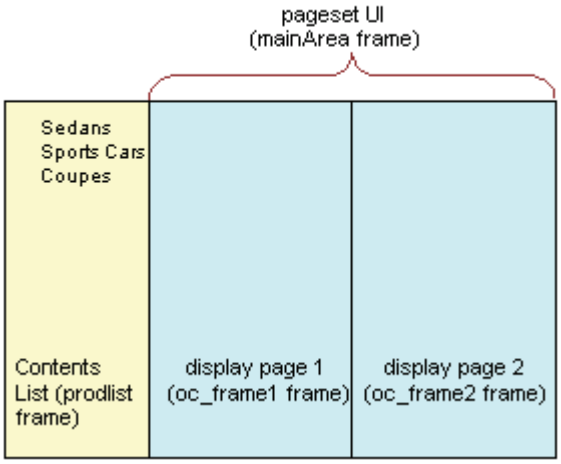
About Modifying the Contents List Location

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 13–1). Use the RegisterContentsListFrame function to indicate which application frame the contents list loads into.

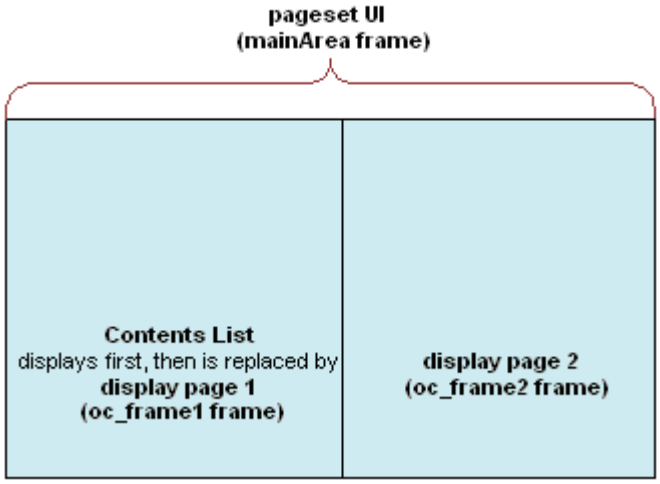
Figure 13–1 Sample Layout in Which the Contents List is Defined in the Application UI Definition File



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 13–2).

Figure 13–2 Sample Layout in Which the Contents List is Defined in the Pageset UI Definition File



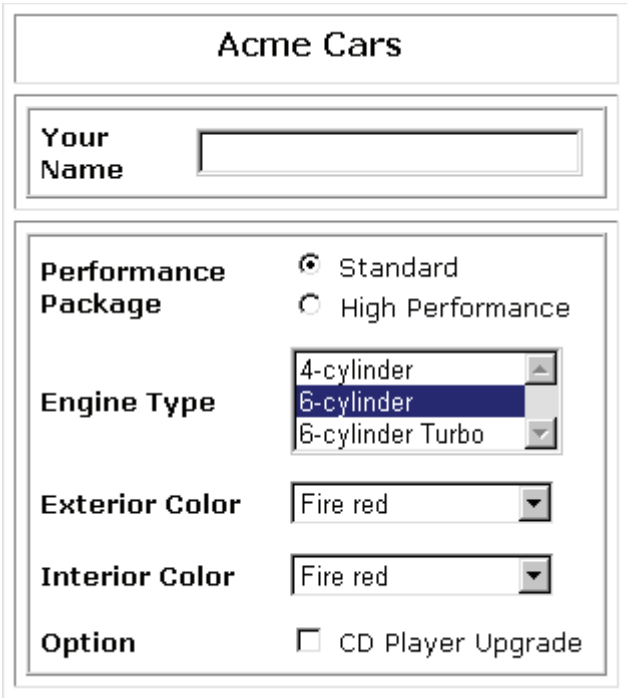
About the UI Controls

Advisor 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.

Example 13–1 Input UI Display Page

Figure 13–3 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.

Figure 13–3 Input UI Display Page



The function call used to create the Engine Type list box in Figure 13–3 looks like this:

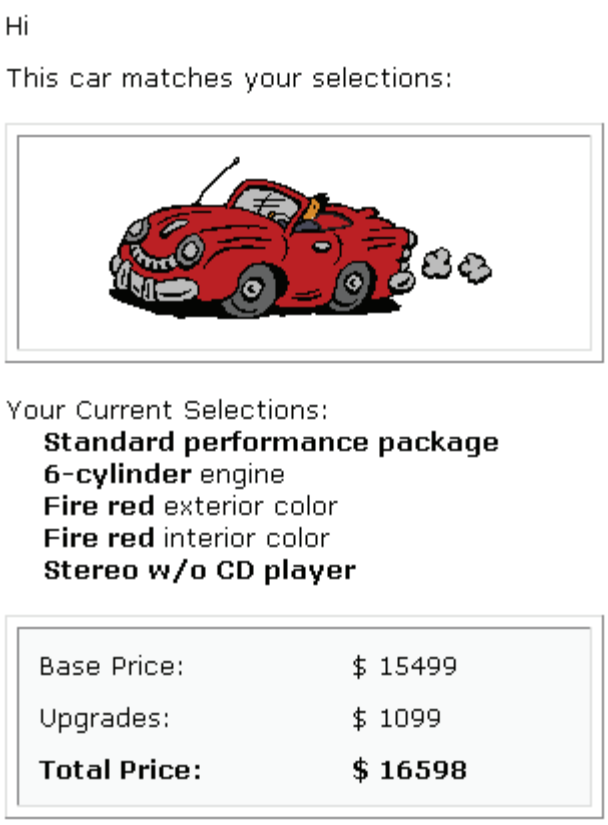
```
<script>document.write(ISS.BuildWidget("LISTBOX",window,"ENGINE",3,32,true));</scr
ipt>
```

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

Example 13–2 Display Page with Output UI Controls

Figure 13–4 shows the display page with output UI controls that result from a valid configuration.

Figure 13–4 Output UI Display Page



The output UI Control text and images on display pages are dynamically created using an Advisor 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 Advisor 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 an Advisor project

1. Navigate to Administration - Product, then Advisor Projects and create a project.

   For more information, see "To create a project" on page 5-2.

   Note: 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.

14

Referencing Other Siebel Data from Advisor

Advisor 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.

This chapter covers the following topics:

About Referencing Other Siebel Data from Advisor Applications

To reference other Siebel data, you:

- 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:
  - Siebel Pricer to provide pricing information in your application
  - Server-based Configurator
  - The Siebel shopping cart or quote
  - Product detail view

Binding Advisor Data to Siebel Business Components

Advisor 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 Advisor 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 does not change if the data changes in the Siebel database. You need to deploy your project to refresh the data.

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

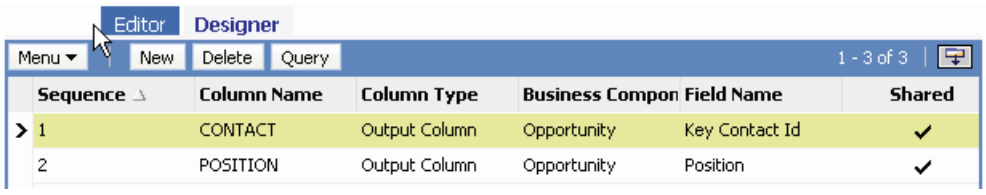
  If more than one column shares the same row of data in the business component, use this check box to specify those extra columns.

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

Configuration Table Designer Example

Figure 14–1 shows the Configuration Table Designer with two shared columns.

Figure 14–1 Shared Columns in a Configuration Table



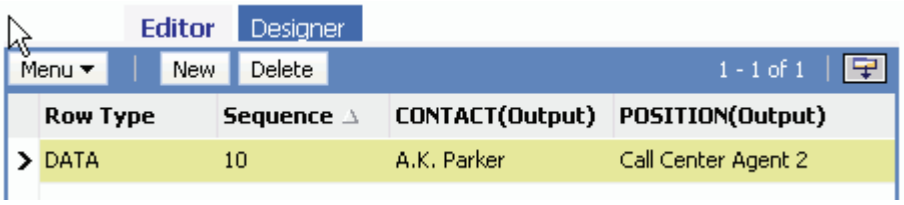
The CONTACT column is mapped to the Key Contact ID field in the Siebel business component named Opportunity. The POSITION column is mapped to the POSITION

field in the Siebel business component named OPPORTUNITY. The two columns, CONTACT and POSITION, share the same target, that is, they both get their values from the same row of data in the Opportunity business component.

You 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 14–2, 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.

Figure 14–2 Shared Columns in a Configuration Table



Adding Access from Advisor to Other Business Components

When designing a feature table or configuration table, select a business component and field to populate a column with values. Advisor 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. Navigate to Administration - Data, then List of Values.
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 appears in the list.

   For example, if you enter 2, the new business component appears second in the list of business components.
8. Check the Active check box.

To create a new Picklist Name record

1. Navigate to Administration - Data, then List of Values.
2. In the List of Values screen, click New.
3. From the Type drop-down list, select ISSCDA_DESIGNER_PICKLIST_TYPE.
4. In the Display Value text box, enter the name for the new picklist, for example, PickList Project.
5. In the Language Independent Code text box, enter the same Picklist name you entered in Step 4.
6. Click Language Name to open a picklist and select a language.
7. Check the Active check box.

To create a new Pick Applet Name record

1. Navigate to Administration - Data, then List of Values.
2. In the List of Values screen, click New.
3. From the Type drop-down list, select ISSCDA_DESIGNER_PICKAPP_TYPE.
4. In the Display Value text box, enter the name for the new pick applet, for example, Project Pick Applet.
5. In the Language Independent Code text box, enter the Pick Applet list name you entered in Step 4.
6. Click Language Name to open a picklist and select English-American.
7. Check the Active check box.

About Modeling for Customizable Products in Advisor

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, Configurator, or Pricer, you 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 Configurator.

- 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, see *Siebel Product Administration Guide*.

Data Evaluation in Advisor 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 for Modeling Customizable Products" on page 14-9 for more information)

Data Evaluation in Advisor 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 in Advisor Applications

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 Advisor 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 Advisor 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 14–3.

Figure 14–3 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

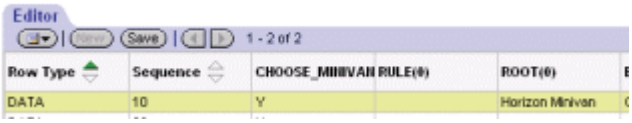
1. Navigate to Administration - Product, then Advisor Pagesets and Configuration Tables.
2. Select a Configuration table to map the root product and its attributes.

   Depending on your modeling needs, you may create a Configuration table specifically for the purpose of mapping products.
3. Switch to the Designer view for the Configuration table.
4. Create a type (0) column and give it any name other than a reserved name.
5. Map the column to the business component "Internal Product" by selecting it from the picklist.
6. 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.
7. Switch to the Editor view.

   A Picklist icon appears in the column you created and mapped in 6.
8. Click the Picklist icon to open a list of Internal Product business components.
9. 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.



10. 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 Feature Tables

Use the following procedure to map root product attributes in Feature tables.

To map root product attributes in Feature tables

1. Navigate to Administration - Product, then Advisor Pagesets and Feature Tables.
2. Select the Feature table that you want to map the attribute to.
3. Switch to the Designer view.
4. Create a Standard column and give it any name other than a reserved name.
5. Map the column to a class by selecting a class from the Class picklist.
6. Map the column to a specific attribute of the class by selecting it from the Attribute picklist.
7. Switch to the Editor view.
8. Enter the exact value for the attribute you want the root product to have when the Feature table row is active.

9. Repeat Step 8 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

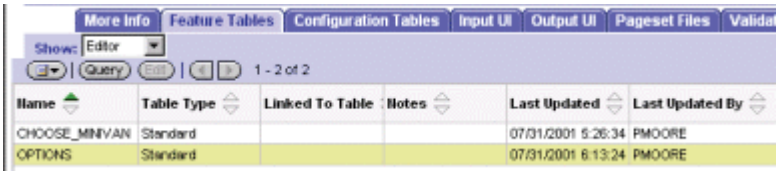
Note that you must select a product for the root, but you can choose not to select a product for a child. You may choose not to add a child by:

- 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. Navigate to Administration - Product, then Advisor Pagesets and Feature Tables.
2. Select the Feature table you want to map to the child product.

   This procedure uses the following feature table as an example.

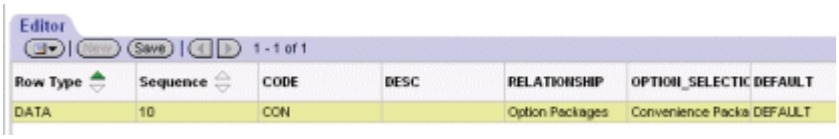


3. Switch to the Designer view.
4. Create a column and give it any name other than a reserved name.
5. Map the column to the Internal Product business component by selecting it from the picklist.
6. Select a field from the list of fields in the Internal Product business component.

   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.
7. Create another column and give it the reserved name RELATIONSHIP and do not map it to anything. RELATIONSHIP is a reserved word.
8. Switch to the Editor view.
9. 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.
10. 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.



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
- Mapping Root Product Attributes in Feature Tables

Best Practices for Modeling Customizable Products

Before you start modeling in Advisor 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 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, see ISS.BuldProductStr in *Siebel Advisor 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 reports this conflict. However, if you pass incomplete information, the resulting product may not be consistent with the rules defined in the server-based Configurator and incomplete products may 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 of Advisor Applications with the Shopping Cart or Quote

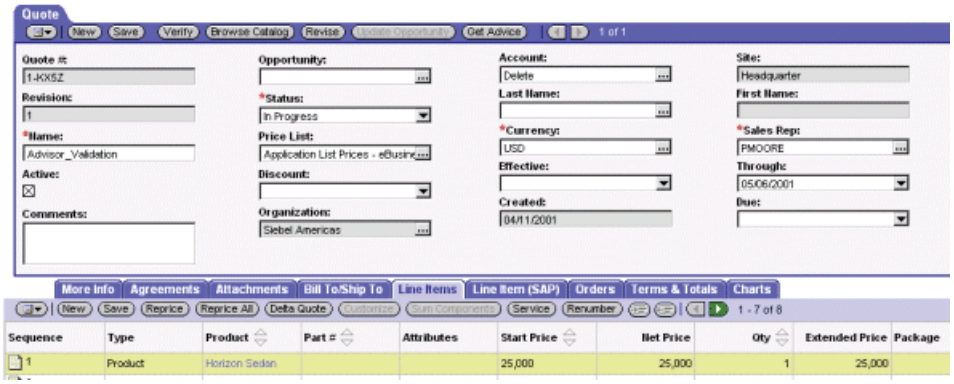
By default, a link to the Siebel shopping cart or quote is included on the default UI Output page of your application. You can also add more links to the Siebel shopping cart or quote in your application.

Figure 14–4 shows an application that includes an Add to Cart button.

Figure 14–4 Application That Links to a Quote

In the example shown in Figure 14–5, when a user clicks the Add to Cart button, the sedan is added to the user's quote.

Figure 14–5 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.

   For more information about adding buttons see, "To create three buttons that display different applets" on page 11-3.

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, see *Siebel Advisor API Reference*.

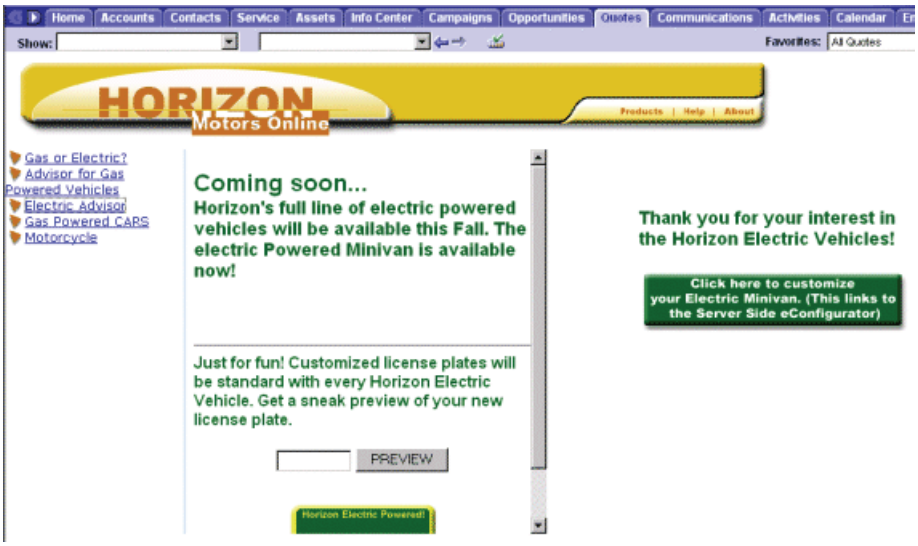
Note: To add any items to the cart or quote from applications, the recommendation list or product must exist in the Siebel product master. The product master runs off the Siebel Internal Product business component.

Runtime Interaction of Advisor Applications with Server-Based Configurator

You can integrate your Advisor applications with the server-based Configurator. For example, you may want to use an Advisor 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 Configurator. For information on the difference between browser-based and server-based applications, see "Siebel Application Deployment Methods" on page 2-5.

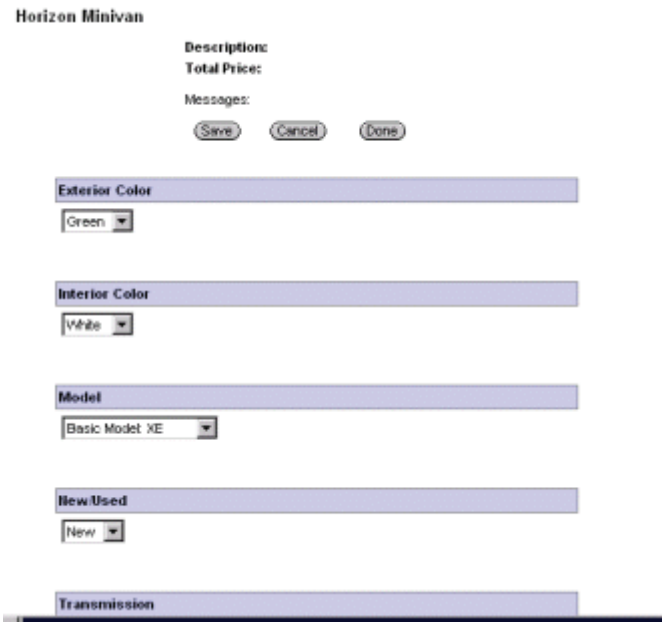
Figure 14–6 shows an Advisor application that links to the server-based Configurator.

Figure 14–6 An Advisor Application That Links to Server-Based Configurator



In this example, when a user clicks the Configuration button shown in Figure 14–7, the server-based Configurator opens to allow the user to further customize their minivan.

Figure 14–7 An Configurator Session Started from an Advisor Application



For more information about the functions referred to in the following procedure, see *Siebel Advisor API Reference*.

To integrate with the server-based Configurator

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 in Advisor" on page 14-4.

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 Configurator, replacing the browser-based view in the browser with the server-based Configurator view. If a string is specified, it is used to describe the customizable product, which is used to start up the Configurator. 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 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, see ISS.BuldProductStr in the Siebel-Specific Functions chapter of *Siebel Advisor 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 Pricer.

   – 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, see ShowProductDetails in the Siebel-Specific Functions chapter of *Siebel Advisor 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 navigates to at runtime.
   - ISSCDAProdDetBusCompName: defines the Business Component Name that a product detail view uses.
   - ISSCDAProdDetBusObjName: defines the Business Object Name that a product detail view uses.
   - ISSCDAProdDetViewName: defines the product detail view name.

Example 14–1 Creating a Cross-Sell

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 Configurator. 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 Configurator.

For more information on the following steps, see *Siebel Advisor 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 Pricing Information in Advisor Applications

There are two ways of displaying pricing information in browser-based applications:

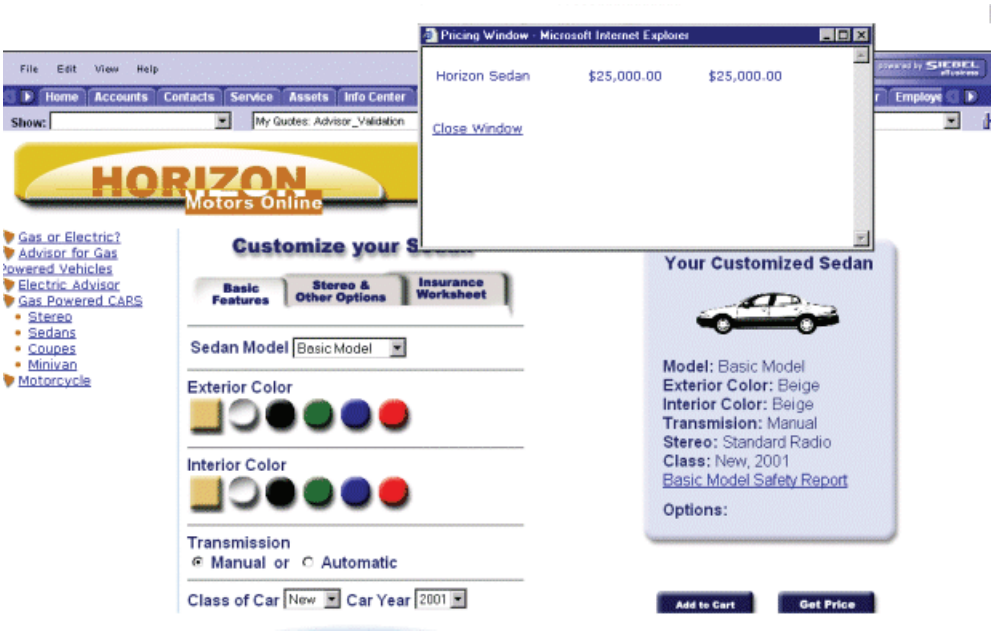
- Publishing static pricing in static pagesets

  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 14–8.

Figure 14–8 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:

start price * adjustment = 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 Advisor, you select a price list at the project level. For more information, see "Associating a Price List with a Browser-Based Model" on page 14-15.

Adjustment is defined in Pricer. To apply adjustments (such as promotional prices, volume discounts, or context-specific adjustments) to the static list price, you need to use functionality provided by Siebel Pricer.

Using Pricer, 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, see *Siebel Pricing Administration Guide*.

To provide access to your pricing information

1. Associate your price list with the Browser-Based model in Advisor.
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, see *Siebel Advisor 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 a 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, then List of Values and search for Master Price List. The Display Value field displays the price list ID of the default price list.

Whenever the price list or prices for price list line items are updated, you 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 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, see *Siebel 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, are 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 appears, 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 variable are identified using session profile attributes that were declared in Siebel Tools.

  For more information on personalization, see *Using Siebel Tools*.

- 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, see the Browser-Based Application File Reference chapter in *Siebel Advisor 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 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, see the Browser-Based Application File Reference chapter in *Siebel Advisor API Reference*.

Adding Rules Based Pricing

At runtime, users can click Get My Price to access a personalized price. This price is generated by Siebel Pricer. These rules are created in the Pricing Administration screen.

For information about Siebel Pricer, see *Siebel Pricing Administration Guide*.

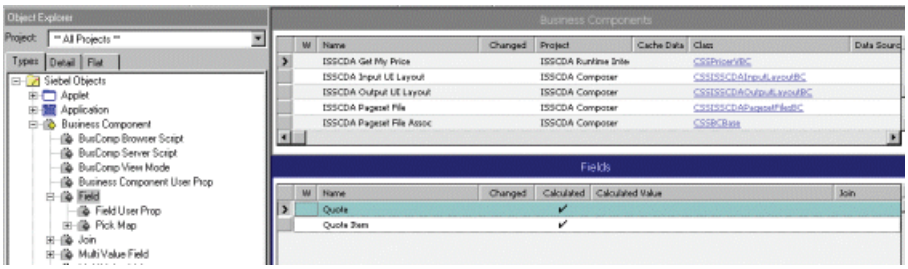
For information about enabling pricer related fields in Advisor, see the rest of this topic.

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

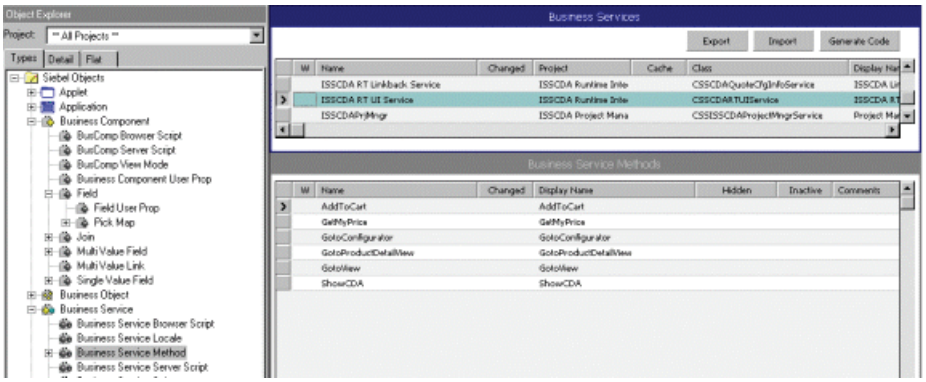
prices. In Figure 14–9, quote and quote item are used because the default integration object for Add to Cart is quotes.

Figure 14–9 The ISSCDA Get My Price Business Component



When the user clicks Get MyPrice, ISSCDA RT UI Services business service generates session specific information, as shown in Figure 14–10.

Figure 14–10 The ISSCDA RT UI Service

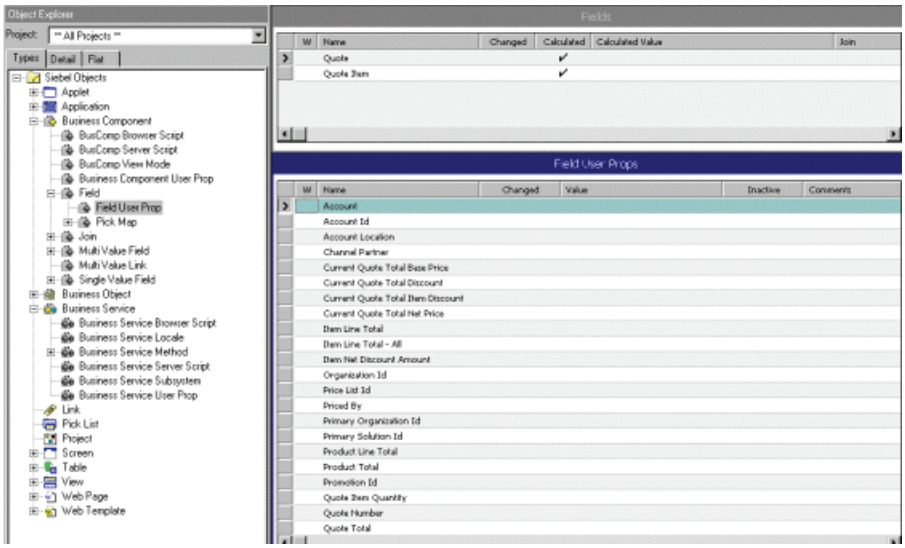


The ISSCDA RT UI Services business service generates session specific information in the following order:

1. The configurable product information for the current pageset is generated.

   See "Entering Data in the Advisor Feature Table" on page 7-5 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.



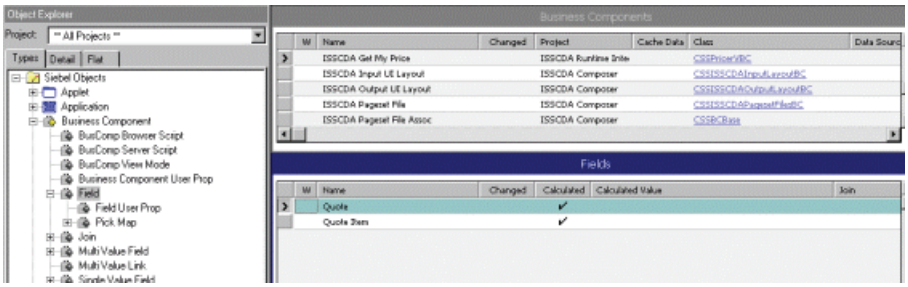
In this example, the business service tries to look up values for Account, Account ID, Account Location, and so on from the quote.

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 ISSCDA Get My Price virtual business component.

To add variables in the ISSCDA 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, then Field and 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 appears 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 ISSCDA Get My Price Virtual Business component. For more information, see "About Configuring the Get My Price Virtual Business Component" on page 14-17.

15

Working with Advisor Project Files

This chapter describes the Advisor files that make up your application, and the methods for working with them.

This chapter covers the following topics:

About the Advisor Project Files

Advisor 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 Oracle's 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 Advisor files, see *Siebel Advisor API Reference*.

Advisor provides a structured approach to creating and managing the files that make up an application. An Advisor 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 Advisor 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 Advisor. For example, a Contents file in an application is represented in Advisor as a contents list table.

Certain application files are represented by a collection of Advisor project objects, as opposed to a single object. For example, a single Configuration Data file in an application is represented in Advisor as a collection of Configuration tables.

Advisor generates application files from Advisor project objects when you preview or deploy your project.

Advisor Project Structure

The organization in which Advisor 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. Advisor 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 Advisor 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 directories and files listed in this section.

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 Advisor 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 Advisor application. These files apply to the application in general and not to any specific pageset.

Viewing the Advisor Project Files

Use the following procedure to view the project files.

To view the project files of an Advisor project

1. Navigate to Administration - Product, then Advisor Projects.
2. Select a project.
3. Select the Project Files tab.

   All files that make up the project are listed.

To open and view a project file of an Advisor project

1. Navigate to Administration - Product, then Advisor Projects.
2. Select a project.
3. Select the Project Files tab.

   All files that make up the project are listed.
4. Select a Project File

   A dialog opens asking if you would like to request the file from the server.
5. Click OK and 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, see *Siebel Advisor API Reference*.

Creating a File Attachment to an Advisor Project File

Use the following procedure to create a file attachment.

> **Note:** Attached files increase the amount of time some Advisor 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. Navigate to Administration - Product, then Advisor Projects.
2. Select a project.

3. On the Project Files tab, click New.

   A new File Attachment record appears.

4. In the Attachment Name field, click the button to select an attachment name.

5. Enter the directory name and any notes.

   Note: The combination of the attachment name and the directory name must be unique.

   The file is added as an attachment to the list of files displayed in the Project Files tab.

Editing a File Attachment to an Advisor Project File

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.

   For more information, see "To open and view a project file of an Advisor project" on page 15-3.

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. Navigate to Administration - Product, then Advisor Projects.

2. Select a project.

3. Select the Project Files tab.

   All files that make up the project are listed.

4. Select a project file.

5. Update the fields for the project file.

A

Implementation of Multi-Variable and Cascading Triggers

In order to use the multi-variable triggers and cascading triggers features of Advisor applications, use the following installation and implementation instructions.

> **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 Advisor.

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);
}
```

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

B

C

D

E

F

H

I

K

L

M

O

P

R

S

T

U

V

W