Siebel

Configuring Siebel Business Applications

August 2026

Siebel
Configuring Siebel Business Applications

August 2026

F84302-12

Copyright © 1994, 2026, Oracle and/or its affiliates.

Author: Sukanya Mishra

Contents

Get Help i

1 What's New in This Release 1

What's New in Configuring Siebel Business Applications, Siebel CRM 26.4 Update 1
What's New in Configuring Siebel Business Applications, Siebel CRM 26.3 Update 1
What's New in Configuring Siebel Business Applications, Siebel CRM 25.3 Update 1
What's New in Configuring Siebel Business Applications, Siebel CRM 24.10 Update 2
What's New in Configuring Siebel Business Applications, Siebel CRM 24.8 Update 2
What's New in Configuring Siebel Business Applications, Siebel CRM 24.4 Update 2
What's New in Configuring Siebel Business Applications, Siebel CRM 23.10 Update 3
What's New in Configuring Siebel Business Applications, Siebel CRM 23.6 Update 3
What's New in Configuring Siebel Business Applications, Siebel CRM 23.3 Update 4
What's New in Configuring Siebel Business Applications, Siebel CRM 22.8 Update 4
What's New in Configuring Siebel Business Applications, Siebel CRM 22.6 Update 4
What's New in Configuring Siebel Business Applications, Siebel CRM 22.5 Update 5
What's New in Configuring Siebel Business Applications, Siebel CRM 21.5 Update 5

2 Architecture of Siebel CRM 7

Architecture of Siebel CRM 7
About Siebel Objects 7
About the Siebel Object Architecture 9
About the Siebel Operating Architecture 17
About Siebel Technologies That Configure Siebel CRM Behavior 22

3 About Tables and Columns 27

About Tables and Columns 27
About Siebel Tables 27
Options to Configure the Data Objects Layer 42
Guidelines for Configuring the Data Objects Layer 45
Limitations on Use of Direct SQL Against Siebel Databases 51

4 About Business Components, Fields, Joins, and Links 57
About Business Components, Fields, Joins, and Links 57
About Business Components 57
How a Business Component Gets Data from an External Database 60
About Business Component Fields 65
How Siebel CRM Defines Read-Only Behavior for a Business Component Field 70
About Joins 76
About Multi-Value Links 80
About Links 88
Creating a New Business Component using the Web Tools Wizard 89

5 About Business Objects 91
About Business Objects 91
Business Objects and Business Components, Views, and Screens 91
How Siebel CRM Creates a Business Object 93
Guidelines for Creating a Business Object 95

6 About Applets, Controls and List Columns 97
About Applets, Controls and List Columns 97
About the Form Applet and List Applet 97
About Applet Controls and List Columns 99
Options to Create an Applet 101
Guidelines for Creating an Applet 108
Guidelines for Creating a Control or List Column 111

7 About Views, Screens, and Applications 113
About Views, Screens, and Applications 113
About the Siebel Client Navigation Model 113
About Views 114
About Screens 116
Options to Create a View or Screen 121
About Applications 125

8 About Siebel Web Templates and Siebel Tags 127
About Siebel Web Templates and Siebel Tags 127
About Siebel Web Templates 127

About View Web Templates 133
About Applet Web Templates 134
About Siebel Tags 145
Guidelines for Configuring Siebel Web Templates and Siebel Tags 154

9 Configuring a Siebel Application 159
Configuring a Siebel Application 159
About Configuring a Siebel Application 159
Roadmap for Configuring a Siebel Application 160
Developing an Implementation Plan 160
Using Development Tools and Setting Up the Development Environment 163

10 Reusing Predefined Objects 175
Reusing Predefined Objects 175
Reasons to Reuse or Not Reuse a Predefined Object 175
Guidelines for Reusing a Predefined Object 176
Process of Determining Whether You Can Reuse a Predefined Object 183

11 Using the Entity Relationship Designer 191
Using the Entity Relationship Designer 191
About the Entity Relationship Designer 191
Process of Creating and Binding an Entity Relationship Diagram 193
Opening or Modifying an Entity Relationship Diagram 195
Modifying Shapes and Lines in the Entity Relationship Designer 197

12 Configuring Tables 203
Configuring Tables 203
Using the New Table Wizard to Create a New Table 203
Creating a Custom Index 205
Adding an Extension Column to a Base Table 206
Configuring Objects to Use a One-To-Many Extension Table 206
Configuring an Extension Table 207
Applying a Data Layer Customization to the Server Database 209

13 Configuring Business Components, Links, and Business Objects 213
Configuring Business Components, Links, and Business Objects 213

Configuring a Business Component 213
Configuring a Business Component Field 219
Configuring a Link 228
Creating a Business Object 232
Configuring a Searchable Virtual Business Component 233

14 Configuring Views, Screens, and Applications 239
Configuring Views, Screens, and Applications 239
Process of Creating a View 239
Configuring a View 242
Process of Creating a Screen 250
Process of Creating a Screen Home Page View 254
Creating and Deploying an Application 266

15 Configuring Applet Layouts 271
Configuring Applet Layouts 271
Process of Using the Applet Web Template Editor 271
Options for Configuring an Applet Layout 274
Using Grid Layout for an Applet 282

16 Configuring Applets 289
Configuring Applets 289
Creating an Applet 289
Using the Applet Method Menu Item Wizard in Web Tools 298
Configuring Pop-Up Applets and Windows 299
Configuring Applet Buttons, Controls, and List Columns 306
Configuring How Siebel CRM Displays Data in an Applet 316
Process of Configuring Drilldown from the Calendar Applet 323

17 Configuring Special-Purpose Applets 335
Configuring Special-Purpose Applets 335
Configuring a Chart Applet 335
Configuring a Tree Applet 358
Configuring a Hierarchical List Applet 366
Configuring a File Attachment Applet 372
Configuring an Organization Analysis Applet 377

18 Configuring Lists and Pick Applets 379
Configuring Lists and Pick Applets 379
About Lists and Pick Applets 379
List and Pick Applet Configuration 395
Creating a List of Values 406
Associating an Organization with a List of Values 409

19 Configuring Multi-Value Group, Association, and Shuttle Applets 413
Configuring Multi-Value Group, Association, and Shuttle Applets 413
Creating Multi-Value Groups and Multi-Value Group Applets 413
About Association Applets 423
About Shuttle Applets 429
Creating a Shuttle Applet 430

20 Configuring Menus, Toolbars, and Icons 437
Configuring Menus, Toolbars, and Icons 437
About Menus and Toolbars 437
Configuring Menus and Toolbars 441
Configuring Icons 447

21 Configuring Siebel Web Templates 457
Configuring Siebel Web Templates 457
Configuring Siebel Web Templates and Web Pages 457
Configuring Web Templates to Display Menus, Toolbars, and Thread Bars 462
Configuring an HTML Control Type 467

22 Improving the Performance of Siebel CRM 473
Improving the Performance of Siebel CRM 473
Using the Case Insensitivity Wizard to Improve Query Performance 473
Improving the Performance of a Siebel Application 485

23 Mapping a Custom Table to an Interface Table for Siebel EIM 493
Mapping a Custom Table to an Interface Table for Siebel EIM 493
Overview of Using Siebel EIM for Bulk Import and Export of Data 493
Mapping a Custom Table to an Interface Table 499

Mapping a Table to an EIM Interface Table in Siebel Web Tools 505

24 Configuring Dock Objects for Siebel Remote 507

Configuring Dock Objects for Siebel Remote 507
About Dock Objects 507
Configuring Dock Objects 512

25 Localizing Siebel CRM 521

Localizing Siebel CRM 521
Overview of Localizing a Siebel Application 521
Localizing a Multilingual List of Values 524
Converting Your Current Data for an MLOV 535
Configuring Certain Siebel Modules to Use MLOV Fields 540

26 Configuring Data Visualization 547

Configuring Data Visualization 547
About Data Visualization 547
About Data Visualization Components 548
About Sample Industry Dashboards 552
Configuring Data Visualization Components 595
Creating a New Data Visualization Dashboard 660

27 Configuring the Customer Dashboard 661

Configuring the Customer Dashboard 661
Overview of the Customer Dashboard 661
Enabling the Customer Dashboard 662
Process of Configuring the Customer Dashboard 663
Modifying the Appearance and Layout of the Customer Dashboard 667

28 Reference Materials for Configuring Siebel CRM 683

Reference Materials for Configuring Siebel CRM 683
Properties of Object Types 683
Types of Applet Controls and List Columns 699
Objects You Use with Enterprise Integration Manager 702
Types of Tables and Columns That CIAI Query Supports 710

Extensive Code Examples That This Book Uses 712

Get Help

Preface

This preface introduces information sources that can help you use the application and this guide.

Using Oracle Applications

To find guides for Oracle Applications, go to the Oracle Help Center at *https://docs.oracle.com/*.

Documentation Accessibility

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

Contacting Oracle

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit *My Oracle Support* or visit *Accessible Oracle Support* if you are hearing impaired.

Comments and Suggestions

Please give us feedback about Oracle Applications Help and guides! You can send an email to: *oracle_fusion_applications_help_ww_grp@oracle.com*.

1 What's New in This Release

What's New in Configuring Siebel Business Applications, Siebel CRM 26.4 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Types of Applet Controls and List Columns	Modified topic. Radio Button type added to the table. Explains the **SHOWNONE** control user property.

What's New in Configuring Siebel Business Applications, Siebel CRM 26.3 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Handling Browser Script in Siebel Versions Prior to 20.8	New topic. This topic describes how browser scripts are handled in Siebel CRM across versions and provides the steps required to generate and deploy browser scripts using the genbscript utility for releases prior to 20.8.
Using the Applet Method Menu Item Wizard in Web Tools	New topic. This topic describes how to use the Applet Method Menu Item Wizard in Web Tools to manage an applet's menu.

What's New in Configuring Siebel Business Applications, Siebel CRM 25.3 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Creating a Multi-Value Group (MVG) Applet in Web Tools Using a Wizard	New topics. New object wizards in Web Tools that allow developers to quickly create MVG, Pick and Tree Applets, improving productivity and overall developer experience.

Topic	Description
Creating a Pick Applet in Web Tools Using a Wizard	
Creating a Tree Applet in Web Tools Using a Wizard	

What's New in Configuring Siebel Business Applications, Siebel CRM 24.10 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Limitations on Use of Direct SQL Against Siebel Databases	New topic. This topic discusses what should and should not be allowed for SQL DDL and DML statements against Siebel Databases.

What's New in Configuring Siebel Business Applications, Siebel CRM 24.8 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Creating an Applet	Modified topic. Added topics describing steps in creating applets in Web Tools using a Wizard.

What's New in Configuring Siebel Business Applications, Siebel CRM 24.4 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Format Toolbar for Web Tools	New topic. Web Tools can format controls in Form applets. Formatting Applet Controls refers to their placement, alignment, size, and spacing.

What's New in Configuring Siebel Business Applications, Siebel CRM 23.10 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Using the Pick List Wizard in Web Tools	New topics. These are two new object wizards for Web Tools.
Using the Multi Value Group Wizard in Web Tools	

What's New in Configuring Siebel Business Applications, Siebel CRM 23.6 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Creating a New Command Object using the Web Tools Wizard *Creating a New Business Component using the Web Tools Wizard* *Creating a New Transient Business Component using the Web Tools Wizard*	New topics. As of Siebel CRM 23.6, Web Tools provide wizards that allow you to create new objects such as Business Components, Integration Objects, and perform various other Web Tools tasks.

What's New in Configuring Siebel Business Applications, Siebel CRM 23.3 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Mapping a Table to an EIM Interface Table in Siebel Web Tools	New topic. As of Siebel CRM 23.3 Update, mapping an EIM Interface Table to a Siebel Table starts with the EIM Table Mapping Wizard.

What's New in Configuring Siebel Business Applications, Siebel CRM 22.8 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Guidelines for Creating a Function-Based Index	New topic. As of Siebel CRM 22.8 Update, developers can work with their database administrators to create function-based indexes in Siebel Tools or Siebel Web Tools, where they are deemed appropriate for performance reasons in your deployment.

What's New in Configuring Siebel Business Applications, Siebel CRM 22.6 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Configuring Data Visualization	Modified topics. Additional configuration information is provided for data visualization dashboards that was omitted from this guide for Siebel CRM 22.5 Update.

What's New in Configuring Siebel Business Applications, Siebel CRM 22.5 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Configuring Data Visualization	New chapter. As of Siebel CRM 22.5 Update, new data visualization components and dashboards are provided that customers can configure for their own uses.

What's New in Configuring Siebel Business Applications, Siebel CRM 21.5 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Adding a Custom Control to the Customer Dashboard	Modified topic. Updated the code examples.
Multiple topics	Modified topics. As of Siebel CRM 21.2 Update, the **applicationcontainer** directory has been replaced by two directories, as follows: • **applicationcontainer_external** (for Siebel Application Interface) • **applicationcontainer_internal** (for all other Siebel Enterprise components) In the Siebel Application Interface installation, web artifacts for application configurations, which were formerly located in **applicationcontainer\webapps\siebel**, now map to **applicationcontainer_external\siebelwebroot**. This directory contains subdirectories such as **files**, **fonts**, **htmltemplates**, **images**, **migration**, **scripts**, and **smc**. For more information, see *Siebel Installation Guide* .
Multiple topics	Modified topics. Updated obsolete references to Siebel Web Format (SWF) files. Such files were previously migrated into the Siebel database and are now referred to as web templates containing SWF content or as format templates.

2 Architecture of Siebel CRM

Architecture of Siebel CRM

This chapter describes an overview of the architecture for Oracle's Siebel CRM applications. It includes the following topics:

- *About Siebel Objects*
- *About the Siebel Object Architecture*
- *About the Siebel Operating Architecture*
- *About Siebel Technologies That Configure Siebel CRM Behavior*

About Siebel Objects

A Siebel *object definition* is the metadata that defines a Siebel application. Siebel object definitions define user interface elements that Siebel CRM includes in the Siebel client, business entities, and the Siebel Database. The *Siebel Repository* is a set of database tables that stores these object definitions. Examples of types of objects include applets, views, business components, and tables. You can use Oracle's Siebel Tools to create or modify an object definition.

An object definition includes *properties* that are qualities of the software construct that the object defines. For example, the properties of a database column includes the name, data type, and length of the column.

This book uses the terms Siebel Business Applications and Siebel CRM interchangeably to mean the same thing.

Note: For more information about objects supported in Web Tools, see *Using Siebel Tools* .

How Siebel Tools Displays Relationships Between Objects

A *parent-child relationship* is a hierarchical relationship that defines a relationship between object definitions. If you expand an object type in the Object Explorer, then Siebel Tools displays the child objects of the *parent-child relationship*. For example, if you expand the Applet tree, then Siebel Tools displays the following child object types:

- Applet Method Menu Item
- Applet Browser Script
- Applet Server Script
- Applet Toggle

A parent-child relationship between object definitions implies that the child object definition is in, or belongs to, the parent object definition. It does not imply inheritance between object definitions. The properties of a parent object are not related to the properties of a child object.

Terms such as object, property, or class describe the Siebel CRM metadata. These terms do not describe corresponding terms in object-oriented programming.

For more information about the Object Explorer and the Object List Editor, see *Using Siebel Tools* .

How This Book Describes Objects

For brevity, this book describes how an object, such as a user property, does something. For example:

The Copy Contact user property copies contacts.

In reality, the Copy Contact user property only includes information that some other Siebel CRM component uses to copy contacts.

For brevity, this book typically only describes the property name to describe how Siebel CRM uses the value that a property contains. For example, assume Siebel CRM displays the value that the Display Name property contains. This is a property of a tree node object. This book only states the following:

Siebel CRM displays the Display Name property of the tree node.

In reality, Siebel CRM displays the value that the Display Name property contains.

How This Book Describes Relationships Between Objects

An object definition includes properties and a property includes a value. For example, the Business Object property of the Account Address view contains a value of Account. To describe this relationship, this book states the following:

The Account Address view references the Account business object.

Sometimes the relationship between objects occurs through multiple objects. For brevity, this book does not always describe the entire chain of relationships that exists between objects through the entire Siebel object hierarchy. For example, the Account business object references the Account business component and the Account Address view references the Account business object. So, this book states the following:

The Account Address view references the Account business component.

Overview for Using This Book

This book uses the following terms:

- A *user* is a person who uses a Siebel CRM client to access Siebel CRM data.
- The *client* is the client of one of the Siebel applications. Siebel Call Center is an example of a Siebel application.
- The *server* is the Siebel Server, unless noted otherwise.
- An *administrator* is anyone who uses an administrative screen in the client to configure Siebel CRM. The Administration - Server Configuration screen is an example of an administrative screen.

Computer font indicates a value you enter or text that Siebel CRM displays. For example:

```
This is computer font
```

Italic text indicates a variable value. For example, the n and the method_name in the following format description are variables:

```
Named Method n: method_name
```

The following is an example of this code:

```
Named Method 2: WriteRecord
```

A *predefined object* is an object that comes already defined with Siebel CRM. The objects that Siebel Tools displays in the Object List Editor immediately after you install Siebel Tools and the Siebel runtime repository but before you make any customization are predefined objects.

The term *focus* indicates the currently active object in the client. To indicate the object that is in focus, Siebel CRM typically sets the border of this object to a solid line.

Depending on the software configuration you purchase, your Siebel CRM applications might not include all the features that this book describes.

Getting Help From Oracle

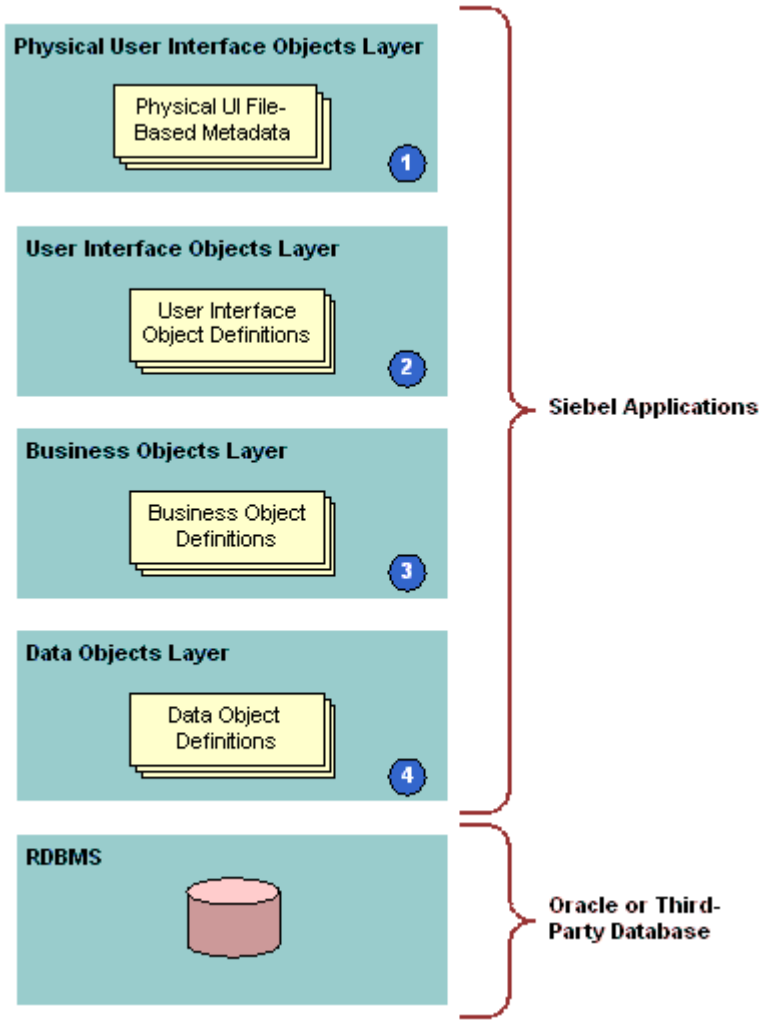
If you require help from Oracle for using object types, you can create a service request (SR) on My Oracle Support. Alternatively, you can phone Global Customer Support directly to create a service request or get a status update on your current SR. Support phone numbers are listed on My Oracle Support. You can also contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance.

About the Siebel Object Architecture

This topic describes the Siebel object architecture. It includes the following information:

- *Overview of the Physical User Interface Layer*
- *Overview of the Logical User Interface Object Layer*
- *Overview of the Business Object Layer*
- *Overview of the Data Objects Layer*
- *Hierarchy of Object Types and Relationships*
- *About Classes*

Siebel CRM separates the metadata that defines objects and files, such as web templates and style sheets, into several architectural layers. The following figure describes this Siebel object architecture.



As shown in this figure, there are four layers in Siebel object architecture as follows:

1. **Physical user interface objects layer.** Includes the physical files, templates, style sheets, and other metadata that reference files. Siebel CRM uses these files to render the user interface that it displays in the client. For more information, see *Overview of the Physical User Interface Layer*.
2. **Logical user interface object layer.** Includes object definitions for the user interface that Siebel CRM displays in the Siebel client. These objects define the visual elements that Siebel CRM displays and that the user uses in a web browser. A user interface object references a business object. For more information, see *Overview of the Logical User Interface Object Layer*.
3. **Business object layer.** Includes objects that define business logic and organize data from underlying tables into logical units. For more information, see *Overview of the Business Object Layer*.
4. **Data objects layer.** Includes object definitions that map the data structures from the underlying relational database to Siebel CRM. These objects provide access to these structures through object definitions that reside in the business object layer. For more information, see *Overview of the Data Objects Layer*.

Siebel CRM insulates objects in each layer of the Siebel Object Architecture from the layers preceding and succeeding each layer, including the RDBMS (Relational Database Management System) that resides at the lowest level of the architecture. This configuration allows you to modify or configure Siebel objects in one layer without affecting objects that reside in other layers. Siebel CRM separates the database in the RDBMS from the object architecture, so you can make any necessary database modifications with only minimal effect on the layers of the Siebel CRM architecture.

Note: Siebel Tools and Web Tools are built as a Siebel application and are a way to configure your Siebel application. However, both the tools cannot be customized and are locked down to run as a shipped configuration. For more information on the related system preferences, see *Siebel Applications Administration Guide* .

Overview of the Physical User Interface Layer

The physical user interface layer includes physical files, such as templates, Siebel tags, style sheets, and other metadata. These files and metadata control the layout and the look and feel of the Siebel client. The physical user interface layer separates definitions for style and layout from user interface objects that reside in the Siebel repository. This separation allows you to define style and layout across multiple objects of the Siebel client. For more information, see *Siebel Developer's Reference* .

For more information about configuring Siebel CRM, see *Developing a Plan to Control File Versions for the Physical User Interface Layer*.

Siebel Web Template

A *Siebel web template* is code in HTML that defines the layout and format of elements that Siebel CRM displays in the Siebel client. Elements include views, applets, controls, and so on. The template provides this layout information to the Siebel Web Engine (SWE) when this engine renders Siebel repository objects into HTML files. Siebel CRM then displays these HTML files in the Siebel client. For more information, see *About Siebel Web Templates*.

Siebel Tag

A *Siebel tag* is a special tag that resides in a web template file. A tag specifies how Siebel CRM defines a Siebel object in the Siebel repository and how Siebel CRM must render it in the HTML page in the web browser. For more information, see *About Siebel Tags*.

Cascading Style Sheet

A *cascading style sheet* (CSS) is a style sheet document that defines how Siebel CRM displays HTML or XML elements and their contents in a web document. It includes typographical and format information on how Siebel CRM displays the web page, such as the text font. It controls the look and feel of user interface elements. A cascading style sheet allows you to control the appearance of the page. For more information, see *Configuring Web Templates to Display Menus, Toolbars, and Thread Bars*.

Overview of the Logical User Interface Object Layer

The logical user interface object layer includes object definitions that determine the visual interface that the user interacts with in a web browser. A user interface object displays data from the business object layer and allows the user to use controls to navigate, view, or modify data.

Applet

An *applet* is a user interface object that allows the user to view, enter, and modify data that the applet gets from a single business component. It includes the following qualities:

- Occupies a section of a view.
- Includes controls, such as buttons, fields, check boxes.
- Allows the user to view, enter, modify, and navigate through records.

- Can display as a form, list of records, chart, business graphics, or navigation tree.
- Allows data entry for a single record or through a scrolling table that displays multiple records.

For more information, see *About Applets, Controls and List Columns*.

A Siebel CRM applet is not equivalent to a Java applet.

View

A *view* is an object that contains a collection of related applets that Siebel CRM displays in the Siebel client. A view can contain lists, forms, charts, and other types of applets. Most views are a master-detail view or a list-form view. A view is associated with the data and relationships that exist in a single business object. For more information, see *About Views*.

Screen

A *screen* is an object that contains a collection of related views. A screen is associated with a major area of the enterprise, such as accounts, contacts, or opportunities. All views in a screen typically reference the same business object. For more information, see *About Screens*.

Application

An *application* is an object that contains a collection of screens. The user can access this application through the following clients:

- Siebel Web Client
- Siebel Mobile Web Client
- Siebel Developer Web Client
- Siebel Mobile applications (connected)
- Siebel Mobile applications (disconnected)

The user navigates to a screen from the tab bar or the Site Map that the object definition of the Siebel application defines. Your organization might possess licenses for more than one of the Siebel CRM applications, for example, Siebel Sales and Siebel Call Center. Different groups in your organization might use these applications, such as the sales team or the customer support team. For more information, see *About Applications*.

A Siebel application is not equivalent to an application executable, such as an .exe file.

Page Tab

A *page tab* is an object that associates a screen to a parent application. It is a child of a screen object and Siebel CRM includes it as a tab in the tab bar. The user clicks the tab to access the screen. For more information, see *Creating a Page Tab*.

Screen Menu Item

A *screen menu item* is an object that associates a screen with the application object. It is a child object of an application. Siebel CRM displays it and its child views as a link on the Site Map. It allows the user to navigate to a screen. For more information, see *Creating a Screen Menu Item*.

Control

A *control* is an object that is an element in the Siebel client, such as a field, text box, check box, button, and so on. It is a child of an applet. It allows the user to interact with Siebel CRM and with CRM data. For more information, see *About Applet Controls and List Columns*.

List Applet

A *list applet* is a type of applet that can display data from multiple records. The predefined list displays data fields in a multicolumn layout where Siebel CRM displays each record of data in a row. A list applet can include textual data, images in JPEG (Joint Photographic Experts Group) and GIF (Graphics Interchange Format) formats, and edit controls, such as check boxes, lists, multi-value group applets, and text fields.

The user can click in the column selection area of a list applet to choose a single row. If chosen, the fields in the row can activate input or edit controls. If the user clicks New in the applet, then Siebel CRM creates a new row that contains a series of empty fields where the user can enter information.

List

A *list* is an object that defines property values that pertain to a scrolling list in a list applet. It is a child of an applet. It provides a parent object definition for a set of list columns. For more information, see *How Siebel CRM Creates a List Applet*.

List Column

A *list column* is an object that corresponds to a column in a scrolling list in a list applet, and to fields in the business component. It is a child of a list. For more information, see *Creating a List Applet*.

Web Template, Applet Web Template, and View Web Template

A *web template*, an *applet web template*, and a *view web template* are objects that identify external HTML files, or other markup language files, that define the layout and Siebel Web Engine interactions for an applet or view. For more information, see the following topics:

- *About the Form Applet and List Applet*
- *About Views*
- *About Siebel Web Templates*

Applet Web Template Item

An *applet web template item* is an object that defines a control or list item in the web implementation of an applet. For more information, see *About the Form Applet and List Applet*.

View Web Template Item

A *view web template item* is an object that defines the inclusion of an applet in the web implementation of a view. For more information, see *About Views*.

Overview of the Business Object Layer

The business object layer includes objects that define business logic and organize data from underlying tables into logical units. This topic describes the more common objects that exist in the business object layer. For more information, see *Siebel Object Types Reference* .

Business Component

A *business component* is an object that associates columns from one or more tables into a single structure. A business component provides a layer of wrapping over tables. This wrapping allows an applet to reference a business component rather than the underlying table. For more information, see *About Business Components*.

Business Component Field

A *business component field* is a child object of a business component that provides data for controls and list columns in an applet. It typically associates a column to a business component. Siebel CRM can calculate a field from the values in other fields in the business component. A field supplies data to a control or list column in the web interface. For more information, see *About Business Component Fields*.

Join

A *join* is an object that creates a relationship between a business component and a table that is not the base table of the business component. A join allows the business component to use fields by using columns from the joined table. To get rows on a one-to-one basis from the joined table, the join uses a foreign key in the business component even though a one-to-one relationship between the two is not required. For more information, see *About Joins*.

Join Specification

A *join specification* is an object that provides details about how Siebel CRM defines the join in the business component. It is a child of a join. For more information, see *How Siebel CRM Uses the Join Specification*.

Business Object

A *business object* is an object that groups related business components together. It allows you to create a relationship among the business components that Siebel CRM uses in the context of a business object. For more information, see *Business Objects and Business Components, Views, and Screens*.

Business Object Component

A *business object component* is an object that references a business component and typically a link in the parent business object. The link specifies how Siebel CRM relates the business component to another business component in the context of the business object. For more information, see *Business Objects and Business Components, Views, and Screens*.

Link

A *link* is an object that defines a one-to-many relationship that exists between two business components. It makes master-detail views possible. A *master-detail* view is a type of view that displays one record of the parent business component with many records of the child business component that correspond to the parent, such as, for example, where a single account includes many contacts. You can use a pair of links to create a many-to-many relationship. For more information, see *About Links*.

Multi-Value Group, Multi-Value Link, and Multi-Value Group Applet

A multi-value group (MVG) is a set of child records that Siebel CRM associates with a parent record. A multi-value link (MVL) is an object that lets you create a MVG. A MVG applet is a dialog box that displays the records that constitute a MVG.

If the user opens the applet from a parent record in a list or form applet, then Siebel CRM uses a MVL to build, and then display a MVG in the MVG applet. For more information, see these topics:

- *Viewing an Example of a Multi-Value Group Applet*
- *About Multi-Value Links*
- *How Siebel CRM Creates a Multi-Value Group*

User Property

A *user property* is an object that allows you to configure behavior that goes beyond the configuration that exists in the properties of the parent object. At the business object layer, a user property exists as a child object of a business component, business service, integration component, integration object, or virtual business component. For more information, see *Siebel Developer's Reference* .

Business Service

A *business service* is an object that contains a set of methods. It allows you to call C++ or scripted methods of the business service from a script that you create, or in the object interface logic, through the mechanism that Siebel CRM uses to call the method. For more information, see *Integration Platform Technologies: Siebel Enterprise Application Integration* .

Overview of the Data Objects Layer

Object definitions that reside in the data objects layer provide a logical representation of the underlying physical database that includes tables, columns, indexes, and so on. These objects are independent of the installed RDBMS, and provide a layer of abstraction over the RDBMS that insulates the Siebel CRM application, so that you do not have to administer and restructure the database.

Siebel object definition layers allow you to manage relational databases in Siebel CRM. Siebel CRM creates queries in reply to a user action in combination with the context that the relevant object definitions create. The RDBMS contains the data and handles the queries that originate in Siebel CRM. Siebel CRM processes the query results that it returns from the RDBMS up through the relevant object definitions that the architecture contains, and then displays the results to the user.

Siebel CRM creates the physical tables in the RDBMS when you install Siebel CRM and create the Siebel database, as described in *Siebel Installation Guide* .

This topic describes some of the more common objects that reside in the data layer of the Siebel Object Architecture. For more information, see *Siebel Object Types Reference* .

Table

A *table* is an object that represents a database table in an RDBMS. It contains child column and index objects that represent the columns and indexes of the table. Siebel Tools displays all of the tables, columns, and indexes that the RDBMS contains. For more information, see *About Siebel Tables*.

Column

A *column* is an object that represents one column in a database table. For more information, see *About Columns and Indexes in a Siebel Table*.

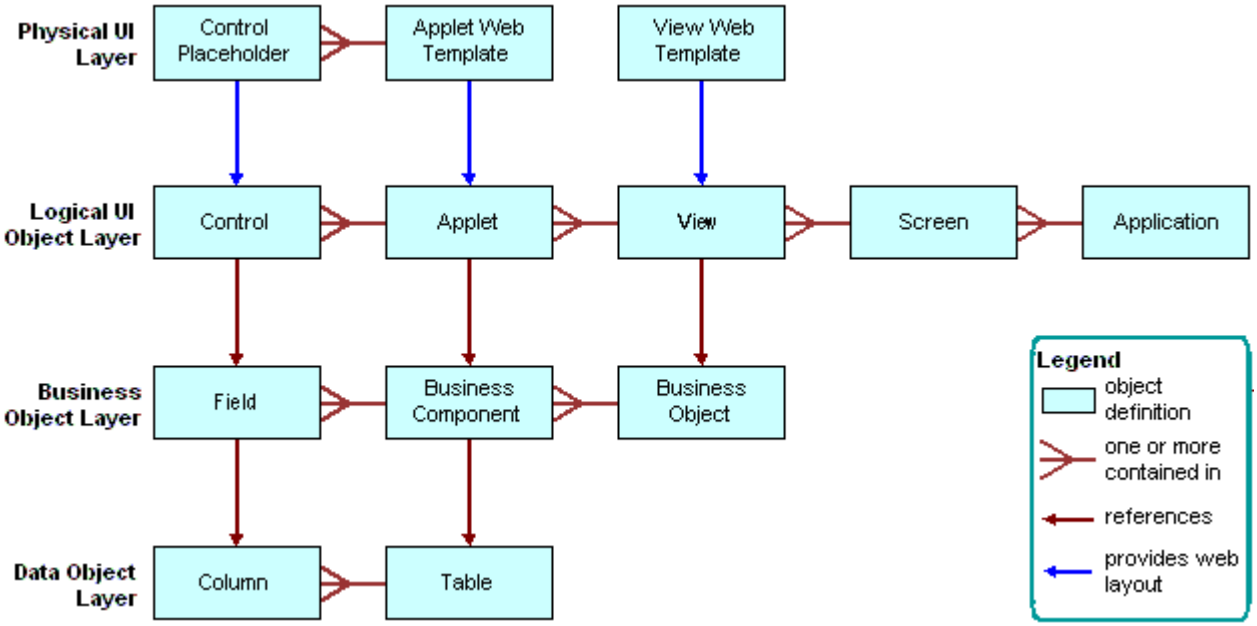
Index

An *index* is an object that identifies a physical index file in the RDBMS. For more information, see *Indexes of a Siebel Table*.

Hierarchy of Object Types and Relationships

Siebel CRM builds an object that resides in an upper layer at start of an object that resides in a lower layer. For example, an applet references a business component, a view references a business object, and a field references a column.

The following figure describes the relationships that exist between the major object types that Siebel CRM uses.



As shown in this figure, the object types are spread over four layers as follows:

1. **Physical UI Layer.** The object types in this layer are: Control Placeholder (provides Web layout for controls), Applet Web Template (provides Web layout for applets), and View Web Template (provides Web layout for views). There is a one-to-many relationship between Applet Web Template and Control Placeholder.
2. **Logical UI Object Layer.** The object types in this layer are Control (references fields), Applet (references business components), View (references business objects), Screen, and Application. There is a one-to-many relationship between Application and Screen, between Screen and View, between View and Applet and between Applet and Control.
3. **Business Object Layer.** The object types in this layer are Field (references column), Business Component (references table), and Business Object. There is a one-to-many relationship between Business Object and Business Component, and between Business Component and Field.
4. **Data Object Layer.** The object types in this layer are Column and Table. There is a one-to-many relationship between Table and Column.

About Classes

A *class* is a property of some objects, such as an applet or business component. The value of the class property assigns a set of behaviors to the object definition and distinguishes it from other categories of object definitions of the object type. For example, a value of CSSFrameList in the class property in the object definition of an applet makes the applet a list applet. To do this configuration, the Class property assigns a DLL to the object definition.

About the Siebel Operating Architecture

This topic describes the major components that the architecture uses to implement Siebel CRM on one or more Siebel Servers. It includes the following information:

- *Components That the Siebel Operating Architecture Uses*
- *Infrastructure That the Siebel Web Engine Uses*
- *How the Siebel Web Engine Creates a Siebel Application*
- *Integration with Java EE*

Components That the Siebel Operating Architecture Uses

This topic describes some of the major components of the Siebel operating architecture, including the following:

- *Object Manager*
- *Siebel Web Engine*
- *Data Manager*
- *Extensible and Nonextensible Objects*

Object Manager

The *object manager* hosts a Siebel application, providing the central processing for HTTP transactions, database data, and *metadata*, which are object definitions in the Siebel repository that are relevant to the current state of Siebel CRM. The Siebel Web Engine and data manager operate as facilities in the object manager.

The object manager handles object definitions for all levels of the object hierarchy. These objects include web interface definitions, business object definitions, and data object definitions. For run-time objects that reference the object definitions, Siebel CRM only directly instantiates the business layer objects. These objects include business objects, business components, and so on. The Siebel Web Engine instantiates interface objects. The data manager instantiates data objects.

For more information about Application Object Manager components that Siebel CRM uses on the Siebel Server, see *Siebel System Administration Guide* .

> **Note:** Siebel Tools and Web Tools are built as a Siebel application and are a way to configure your Siebel application. However, both the tools cannot be customized and are locked down to run as a shipped configuration. For more information on the related system preferences, see *Siebel Applications Administration Guide* .

Siebel Web Engine

The Siebel Web Engine (SWE), a part of the Application Object Manager component on the Siebel Server, creates the Siebel CRM user interface as HTML pages, combines them with data from the Siebel database, then forwards them to the Siebel Application Interface. The Siebel Application Interface includes static files such as those for cascading style sheets, images, fonts, and so on, and then sends the pages to a user's web browser through HTTP. Users can both view and modify data. A notification mechanism allows the Siebel architecture to notify all applets if a user modifies data in an applet, and then all applets can update their data.

Data Manager

The *data manager* is a facility that resides in the object manager. It sends SQL queries in reply to requests that the object manager sends, and it sends back the database result set to the object manager. The data manager includes one DLL connector for each type of database connection that Siebel CRM supports. The object manager dynamically loads the DLL that the data source requires.

For a description of the entities that a Siebel deployment uses, see *Siebel Deployment Planning Guide* . For more information about the Siebel environment, see *Siebel Installation Guide* .

Extensible and Nonextensible Objects

Extensible objects are objects that you can modify and customize according to your business needs, whereas *nonextensible objects* are objects that you cannot modify.

- All end-user facing objects are extensible objects, such as Account, Contact and Service Request, and their associated views, applets, business components, tables, and so on that support them.
- Siebel Tools and Siebel Web Tools are examples of nonextensible objects. Both are built as a Siebel application and are a way to configure your Siebel application. However, both of these tools cannot be customized and are locked down to run as a shipped configuration. For more information on the related system preferences, see *Siebel Applications Administration Guide* .

  Other examples of nonextensible objects include most Administration views, Workspace Framework objects, and objects related to migration behavior.

  All nonextensible objects accessible from the UI are identified by an `[NEO]` prefix or tag. For more information see the following procedure.

To determine whether an object is a nonextensible object

1. Log in to your Siebel business application and navigate to your desired location.
2. Click Help and then select About View.

   On the dialog that appears, the information in the following table appears. Notice that an `[NEO]` tag appears next to any object that cannot be modified.

Field	Description	Example 1	Example 2
Screen	Name of the screen.	[NEO]: ISS Unified Administration Screen	[NEO]: ISS Unified Administration Screen
View	Name of the screen view.	ISS Product Administration Screen	[NEO]: Configurator CE Cache Admin View

Field	Description	Example 1	Example 2
Business Object	Business object associated with the screen.	Admin ISS Product Definition	[NEO]:Configurator CE Cache
Applets	Applets associated with the screen.	Applet[0]: SIS Product List Admin Applet; Applet[1]: SIS Product Form Applet - ISS Admin; Applet[2]: SIS Product Details Form Applet;	Applet[0][NEO]: Configurator CE Cache Item Applet; Applet[1][NEO]: Configurator CE Cache Component Applet; Applet[2][NEO]: Configurator Cache CE Product Applet;
Business Component	Business components associated with the screen.	BusComp[0]: Internal Product - ISS Admin; BusComp[1]: Internal Product - ISS Admin; BusComp[2]: Internal Product - ISS Admin;	BusComp[0][NEO]: Configurator Aggregate CE Cache Item; BusComp[1][NEO]: Configurator CE Deployment Component; BusComp[2][NEO]: Configurator Cache Products;
Cache Mode	The cache mode.	Memory	Memory

About Nonextensible Objects and the Seamless Repository Framework

The seamless repository framework is a new method of updating the Siebel repository that allows customers to take advantage of new functionality (such as Siebel Runtime Repository Version Rollback and Auto Tile Visualization) without doing a repository merge or going through a conflict resolution process during the application of a montly update. The new framework allows Siebel CRM update releases to deliver new (add or update) Siebel repository objects, identified as nonextensible objects, without affecting any customer's existing custom repository objects or requiring a merge or resolution process. Available in Siebel CRM 20.3 Update or later, the seamless repository framework supports all operating system and database combinations and all languages. For more information about this framework, see *Siebel Installation Guide* .

For Object Types consisting of a parent object and child objects, the seamless repository framework merges the respective child objects from the nonextensible object (NEO) files and from the runtime repository definition. For example, if there are 4 views for Screen_A in the NEO file and 5 distinct views for Screen_A in the RR definition, then while loading the definition of Screen_A at runtime, a total of 9 views would be loaded and displayed. The following table shows the different types of (parent-child) Object Types.

Parent Object	Child Object
Application	Screen
Application	Page Tab
Screen	View
Business Object	Business Component

Parent Object	Child Object
View	Applet
Webpage	Web Page Item

Infrastructure That the Siebel Web Engine Uses

The Siebel Web Engine allows you to deploy Siebel CRM in HTML. The Siebel Web Engine is a service that is part of the Object Manager component on the Siebel Server, and communicates with the Siebel Application Interface.

Siebel Application Interface interfaces with the Siebel Web Engine, with most of the work occurring in the Siebel Web Engine. Siebel Application Interface maintains the session and works as a communication intermediary. Network communication between Siebel Application Interface and the object manager occurs through SISNAPI (Siebel Internet Session Network Application Programming Interface), which is a Siebel communication protocol that references TCP/IP. It provides security and compression.

The Siebel Web Engine runs as the Web Engine Interface Service object manager service. This service implements most components of the Siebel Web Engine, deploying an interface between Siebel Application Interface and the object manager. From the perspective of Siebel Application Interface, the Siebel Web Engine interface service does the following work:

- Handles incoming HTTP requests that include the Siebel Web Engine prefix
- Creates HTTP replies

From the perspective of the object manager, the Siebel Web Engine interface provides a user interface for interactions with the object manager.

Where Siebel CRM Hosts Components

If the user accesses Siebel CRM through a web client, then Siebel CRM hosts no components on this client. The client interacts through a web browser. The user accesses a URL that navigates the user to a Siebel application that Siebel CRM hosts on a web server. This application comes predefined with HTML or equivalent pages that the Siebel Web Engine service creates in the object manager.

How You Can Use Siebel Tools to Build a View

You can use Siebel Tools to associate a set of HTML templates with an applet and view to make the applet and view available to the web. When Siebel CRM displays an applet in the client, the Siebel Web Engine gets the information that defines the applet, the data for the various applet controls or list columns, and the HTML template. To create the final web page that Siebel CRM sends to the browser, the engine then combines definition information and data.

To create an applet web template in Siebel Tools, you can use the Web Applet Designer and the following object types:

- Applet Web Template
- Applet Web Template Item

To create a view web template in Siebel Tools, you can use the following object types:

- View Web Template

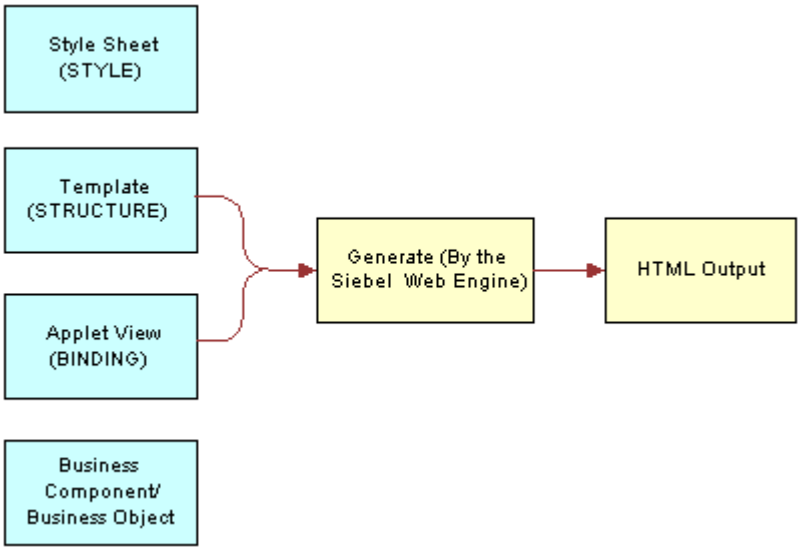
- View Web Template Item

How the Siebel Web Engine Creates a Siebel Application

The user interacts with Siebel CRM through a web browser. The Siebel client includes a set of web pages that the Siebel Web Engine dynamically creates. To do this work, the engine matches the Siebel repository definition of the Siebel application to Siebel web templates.

When the user interacts with Siebel CRM, by clicking a button or link for example, the Siebel Web Engine does the following, as illustrated in the following figure:

1. Reads the object definitions from the Siebel runtime repository.
2. Chooses the necessary web templates.
3. Combines the object definitions and templates.
4. Gets data from the underlying business objects and business components.
5. References the data, applet, and view information to the corresponding placeholders in the Siebel web template.
6. Displays the HTML output to the user.



Integration with Java EE

An enterprise might develop and implement Java applications to meet a variety of business requirements. Typically, these applications combine existing enterprise information systems with new business operations to deliver services to a broad range of users. These services are typically architected as a distributed application that includes the following tiers:

- Clients
- Data sources
- The middle tier between clients and data sources

The middle tier is where you typically find transports and interfaces that receive messages that travel between applications that reside in and out of the enterprise. These transports and interfaces can include HTTP, MQSeries, Java servlets, Enterprise Java Beans (EJBs) that are typically in XML format, and so on.

To simplify integration, Siebel CRM uses Java and XML to receive XML requests that it sends through HTTP or MQSeries. Java and XML provide a uniform way to receive and process requests from Siebel CRM in a Java EE environment. Siebel CRM uses Oracle's Siebel EAI integration infrastructure to transmit requests that Siebel CRM starts to the Java EE Application Server. Java and XML includes a servlet that receives HTTP requests and an MQSeries base server that gets messages from an MQSeries queue.

To use Java and XML, you must implement the ProcessRequest interface that understands the contents of the incoming request and dispatches it to the Java component.

> **CAUTION:** You can use Java and XML only to receive XML requests from Siebel CRM. You can create custom code only for use in object code form and only to integrate a Siebel application with a non-Siebel application. Any modification or extension of this code is not in the scope of Maintenance Services and will void all applicable warranties. For more information, see *Getting Help From Oracle*.

Java Beans Can Represent Siebel Integration Objects or Business Services

You can use the Siebel Code Generator Business Service to create JavaBeans that represent Siebel integration objects or business services. The JavaBeans that the Siebel Code Generator creates a strong interface for integration objects, business services, and their related components. This capability allows you to identify and use the Java code that you require for Siebel CRM.

> **CAUTION:** You can only use the source code that the Siebel Code Generator Business Service creates only in object code form and only to integrate a Siebel application with a non-Siebel application. Any modification or extension of code that the Siebel Code Generator Business Service creates is not in the scope of Maintenance Services and will void all applicable warranties. For more information, see *Getting Help From Oracle*.

For more information about Java, XML, and the Siebel Code Generator Business Service, see *Transports and Interfaces: Siebel Enterprise Application Integration* .

About Siebel Technologies That Configure Siebel CRM Behavior

This topic describes solutions other than Siebel Tools that you can use to configure Siebel CRM. It includes the following information:

- *Siebel Personalization*
- *Task UI*
- *Siebel Workflow*
- *Siebel SmartScript*
- *Siebel Assignment Manager*
- *State Model*
- *Siebel Pricer*

To configure these solutions, you use the administration views in the Siebel client rather than Siebel Tools.

Siebel Personalization

Siebel Personalization is a solution that allows you to filter content in an applet that Siebel CRM displays for a user according to the requirements of the preference or profile of the user. For example, you can include a salutation applet that does the following work:

- Greets the user by name
- Indicates how much time has elapsed since the user last visited the site
- Displays information about products or services

Note the following key points about personalization:

- Personalization is available on any applet in Siebel CRM.
- Personalization uses rules and rule sets to determine the records that the user can view in an applet according to the user profile. These rules evaluate the profile to determine the records and applications to display. A *rule set* is a group of rules. You can define multiple rule sets so that if the criteria in one rule set is not met, then Siebel CRM evaluates the next rule set.
- The user profile references any attribute that belongs to one of the following items:
  - If the user is a contact, a contact record and the account of the contact record
  - If the user is an employee, an employee record and the division of the employee record
- Personalization uses the User Profile Attributes object to contain and get elements of a user profile. You can display these attributes in the client and in rules that determine the content that Siebel CRM displays in the client.
- Siebel Personalization can track events that occur in the context of the Siebel application, business component, or applet. When an event occurs, it starts a Personalization Action that modifies user profile attributes.
- A rule or an event can call an action. Siebel CRM uses an action to set a predefined profile attribute or a profile attribute that it creates dynamically in the client. A dynamic profile exists only for the duration of the user session. You can use a profile attribute that you configure in Siebel Tools or in the client to store state information in much the same way that Siebel CRM stores a variable in a cookie or a persistent frame. Where possible, you must use a profile attribute instead of a cookie.
- A rule or action can call a business component method or a business service method. Typically, you use the method to return values that Siebel CRM uses as criteria for a rule or for setting a profile attribute.

For more information, see *Siebel Personalization Administration Guide* .

Task UI

You can use the task-based user interface (Task UI) to create a wizard-like user interface that Siebel CRM displays in the client. A *task UI* is a multiple-step, interactive operation that can include branching and decision logic. Task UI guides the user through task UI execution, allows forward and backward navigation in this task UI, and allows the user to pause and resume this task UI. These features guide the user through performing an unfamiliar task. It can help to increase the efficiency of a novice or intermittent user. Task UI can increase the efficiency of a busy veteran user especially if the user works in an environment that is prone to interruption. Task UI can help the user to switch between multiple tasks during the work day. For more information, see *Siebel Business Process Framework: Task UI Guide* .

Siebel Workflow

Siebel Workflow is a customizable business application that allows you to manage and enforce business processes, such as response time objectives, creating review policies, or monitoring service requests or opportunities over time. It can use the same basic processes that your organization uses in your sales, marketing, and service departments. You can use Siebel Workflow to enforce business policies and procedures. For more information, see *Siebel Business Process Framework: Workflow Guide* .

Siebel SmartScript

Siebel SmartScript allows you to deploy an interactive guide in question and answer format in a web page that helps the user find information. The interactive guide asks the user to answer questions to refine a search. Depending on the answers, the guide pursues branching paths to locate the correct answer. You can use a single administrative user interface to define scripts, and then deploy these scripts to call center agents or to users through the web.

You can deploy a predefined SmartScript with little or no more configuration. You need only display the SmartScript view, and then Siebel CRM dynamically creates the remaining views, applets, and so on.

Siebel SmartScript can make applications be increasingly driven by data, which simplifies web configuration. For more information, see *Siebel SmartScript Administration Guide* .

Siebel Assignment Manager

Siebel Assignment Manager allows you to assign the most qualified person to a task. To do this, it matches candidates to assignment objects. To assign the most qualified candidate to each object, it applies assignment rules that you define.

You can specify how Siebel CRM uses Assignment Manager to evaluate a record. You can run Assignment Manager to process assignments interactively in real time, dynamically when the user does something that causes Siebel CRM to modify the database, or periodically in batches. For more information, see *Siebel Assignment Manager Administration Guide* .

State Model

State model allows you to configure workflow control according to the status of an object, such as a service request or a product defect. A state represents the status of an object, such as Open, Closed, or Pending. The state represents where the object is in the lifetime of the object. The state can determine whether the user can or cannot modify the data of that object. For example, a service request that is in a Closed state might be considered frozen and the user cannot modify the object.

A *state transition* defines how the user can modify an object from one state to the next. For example, state model can allow a user to modify the state for a service request from Closed to Open, from Open to Pending, but not directly from Closed to Pending. Modifying a service request from Closed to Open or Open to Pending represents a state transition. For more information, see the content about State Model in *Siebel Applications Administration Guide* .

Siebel Pricer

Siebel Pricer is a solution that allows you to define, assess, administer, and deploy a flexible pricing strategy. It includes the following:

- A set of administration views that allow you to define pricing adjustments and the conditions that Siebel CRM uses to apply them.
- An engine that evaluates the condition statements and determines the pricing adjustments that Siebel CRM applies.
- A testing area that allows assessment of the pricing adjustments.
- Integration with user interfaces, such as Quotes, Orders, Siebel Product Configurator, Siebel PRM, and Siebel eSales.

Siebel Pricer includes the following items:

- **Price lists.** Contain base prices.
- **Pricing models.** Management tool to control a set of related pricing factors.
- **Pricing factors.** Statements that define conditions and pricing adjustments.
- **Scripting.** Allows you to use business services with a pricing factor to configure the pricing calculation and to access external data.
- **Pricing validation.** Allows you to test pricing factors and the pricing model before releasing to users.
- **Reports.** Allows you to print reports of pricing factors.
- **Pricer Engine.** Evaluates conditional statements and applies pricing adjustments.

For more information, see *Siebel Pricing Administration Guide* .

3 About Tables and Columns

About Tables and Columns

This chapter describes tables and columns. It includes the following topics:

- *About Siebel Tables*
- *Options to Configure the Data Objects Layer*
- *Guidelines for Configuring the Data Objects Layer*
- *Limitations on Use of Direct SQL Against Siebel Databases*

About Siebel Tables

This topic describes Siebel tables. It includes the following information:

- *Overview of Siebel Tables*
- *Naming Format for a Siebel Table*
- *How an Extension Table Stores Custom Data*
- *How an Intersection Table Defines a Many-To-Many Relationship*
- *About Columns and Indexes in a Siebel Table*
- *How a User Key Creates a Unique Set of Values*
- *How the S_Party Table Controls Access*

For more information, see *Overview of the Data Objects Layer*.

Overview of Siebel Tables

The object definition of a Siebel table is a logical representation of the physical table that resides in the underlying RDBMS. Note the following:

- You can use an extension table to configure the data objects layer.
- You can use an extension column on a base table.
- You cannot add a new base table, delete a base table or column, or modify the properties of a base column.

Siebel CRM uses the term *base table* to describe the following object definitions:

- The table that an extension table configures, as defined in the Base Table property of the extension table
- The table that a business component references, as defined in the Table property of the business component

For more information, see *Guidelines for Naming an Object*.

Naming Format for a Siebel Table

A Siebel table in the Siebel database uses the following three part naming format:

PREFIX_NAME_SUFFIX

The following table describes the naming format.

Part	Description
PREFIX	A one-letter to three-letter prefix that distinguishes the table from other tables. Example prefixes include EIM_, S_, W_, and so on.
NAME	A unique table name that is typically an abbreviation of the name of the entity supertype. For example, the table name for the event supertype is EVT.
SUFFIX	The subtype of the entity. For example, the EVT supertype includes the activity subtype that the ACT suffix represents. For example, S_EVT_ACT.

The following table describes some of the prefixes that Siebel CRM commonly uses. Each prefix indicates the part of the Siebel schema that contains the table.

Prefix	Description
EIM_	Interface table for Enterprise Integration Manager.
S_	Siebel base table. In some situations, a table might contain a name of the form S_*name*_IF. This format indicates an obsolete interface table.
W_	Siebel Business Data Warehouse table.

The following table describes some of the suffixes that Siebel CRM commonly uses. Each suffix indicates a table type.

Suffix	Description
_ATT	File attachment table.
_REL	A table that supports a many-to-many relationship from an entity back to itself.
_SS	A table that stores Siebel-to-Siebel integration information.

Suffix	Description
_X	A one-to-one extension table that you can use to add custom data to the Siebel database.
_XA	A table that stores custom data that Siebel CRM associates with an object class.
_XM	A one-to-many extension table that you can use to add custom data to the Siebel database.

How an Extension Table Stores Custom Data

This topic describes the extension table. It includes the following information:

- *Overview of an Extension Table*
- *How a One-To-One Extension Table Extends Data Storage for a Single Business Component*
- *How an Implicit Join Creates a Relationship Between a Base Table and a Business Component*
- *An Explicit Join Creates a Relationship Between an Extension Table and a Business Component*
- *A One-To-Many Extension Table Stores Data From Multiple Business Components*
- *Summary of Support for Extension Tables and Extension Columns*

For more information, see the following topics:

- *Options to Use a Predefined One-to-One Extension Table*
- *Options to Use a Predefined One-to-Many Extension Table*
- *Manually Creating a One-to-One Extension Table*
- *Using the New Table Wizard to Create a New Table*

Overview of an Extension Table

An *extension table* is a type of table that includes columns that you can use to store custom data. It includes an implicit one-to-one or a one-to-many relationship with a base table. Siebel CRM includes a set of predefined extension tables that you can use. Each of these tables includes generic ATTRIB_ columns that you can use to store custom data. These tables are part of the data objects layer, so you are not required to update the database if you use them.

Siebel CRM uses some ATTRIB_ columns in an extension table. You must not modify or delete an ATTRIB_ column that a predefined Siebel application uses.

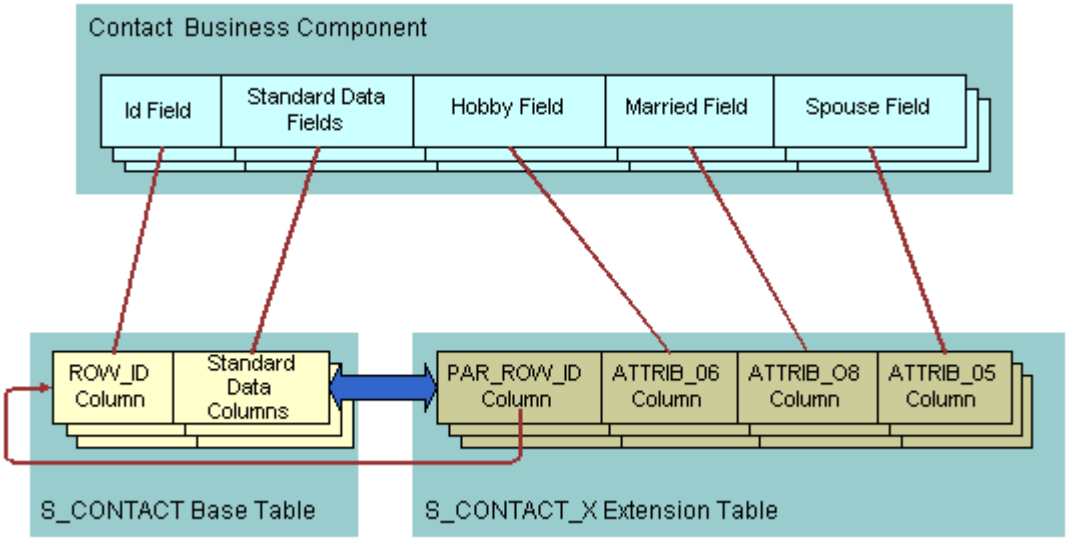
You can use the New Table Wizard to create your own extension table. An extension table that you create requires a modification to the logical schema, so you must apply it to the physical database.

When Siebel CRM updates a column in a base table it does not update the timestamps of the extension tables of this base table unless it also updates the columns in the extension tables. If Siebel CRM modifies a record in an extension table, then it updates the system columns that exist in the parent table. Siebel CRM does this work because the object manager treats the associated record in an extension table as logically part of the parent record.

How a One-To-One Extension Table Extends Data Storage for a Single Business Component

The name of a one-to-one extension table includes an _X suffix. A row in an extension table contains a one-to-one relationship with the corresponding row in the base table. This row is an extension of the base table record. The value of the Type property of a one-to-one extension table is Extension.

The following figure describes an example of how a one-to-many extension table uses new business component fields that reference the base table and maps them to columns that are available in the one-to-one extension table. It adds the Hobby, Married, and Spouse fields to the Contact business component. These fields reference columns that reside in the S_CONTACT_X extension table.



How an Implicit Join Creates a Relationship Between a Base Table and a Business Component

An *implicit join* is a relationship that creates a one-to-one relationship between the extension table, the base table, and the business component. It does the following:

- Creates a relationship between the following objects:
  - Between a one-to-one (_X) extension table and an intersection table.
  - Between an extension table of the S_PARTY table and the S_USER table. S_ORG_EXT, S_CONTACT, and S_POSTN are examples of these extension tables. These implicit joins map data to party business components. For example, if you add a field to the Account business component, and then choose the Join property, then Siebel Tools displays several implicit joins that it does not display in the Joins list, including joins that contain an S_ORG_EXT or S_USER alias.
- Makes the rows of the extension table available on a one-to-one basis to the business component that references the extension table.
- Is part of the Siebel object architecture. You do not use Siebel Tools to explicitly create an implicit join.
- Typically uses the table name as the Join Alias. The name of the implicit join is the same name as the extension table. If a business component field references a column in the extension table, then:
  - The Column property of the Field object contains the name of the column.
  - The Join property contains the name of the extension table.

For example, the Column property for the Industry field in the Contact business component contains ATTRIB_48 and the Join property contains S_CONTACT_X.

- Is sometimes referred to as an *implied join*.

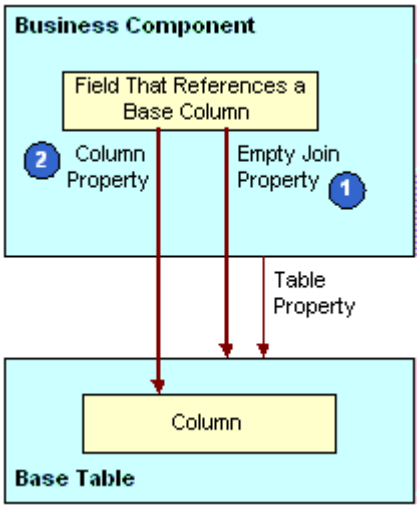
Unlike an explicit join, Siebel CRM can update the columns of an implicit join.

If you create an extension table, then Siebel Tools creates an implicit join. For more information, see *How an Extension Table Stores Custom Data*.

How Siebel CRM Creates an Implicit Join

The following figure illustrates how Siebel CRM creates an explicit join. As shown in this figure, Siebel CRM uses the following objects and properties to create an implicit join:

1. **Empty join property.** If the Join property is empty, then Siebel CRM gets the column from the base table that the business component references.
2. **Column property.** Identifies the table column.



An Explicit Join Creates a Relationship Between an Extension Table and a Business Component

An *explicit join* is a join that is different from an implicit join in the following ways:

- In the Siebel client, the user cannot typically edit a field that references a column from a joined table. You typically use this field only to display information.
- You do not create an implicit join. With an implicit join, the column in the extension table is available for you to use.

You use Siebel Tools to explicitly create the join for other tables. For more information, see *About Joins*.

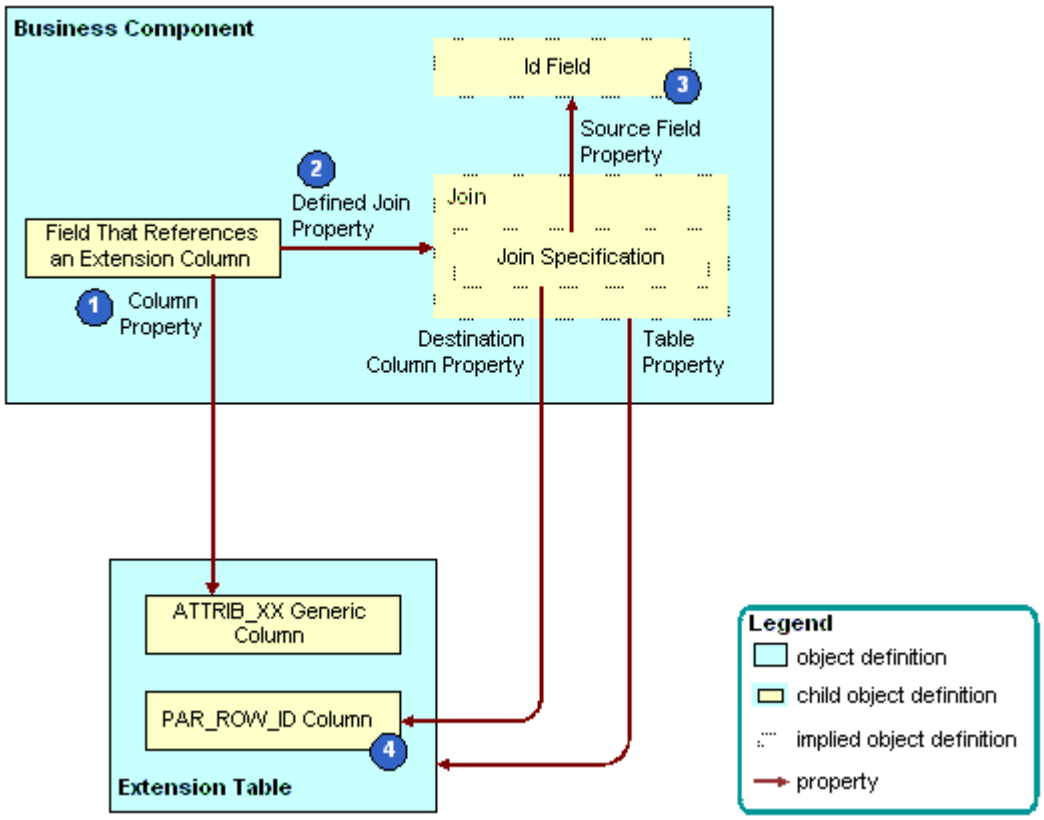
How Siebel CRM Creates an Explicit Join

The following figure illustrates how Siebel CRM creates an explicit join. As shown in this figure, Siebel CRM uses the following objects and properties to create an explicit join:

1. **Column property.** Identifies the table column.
2. **Defined join property.** If the Join property is not empty, then the Join property identifies the join that supplies data from an extension table or other joined table.

3. **Id field.** A system field in the business component. It represents the ROW_ID column in the base table. You can use it in a join that involves an extension table and other joined tables. For more information, see *System Fields of a Business Component*.
4. **PAR_ROW_ID (parent row ID) column.** A column that is a foreign key to the base table that the extension table extends. Every extension table includes a column for parent row ID. Every row in an extension table contains a value in the PAR_ROW_ID column.

For more information, see *Options to Use a Predefined One-to-One Extension Table*.



A One-To-Many Extension Table Stores Data From Multiple Business Components

A *one-to-many extension table* is a table that you can use to track an entity that includes a one-to-many relationship with a parent business component but that a predefined business component does not represent. Note the following:

- You can store data for multiple business components in a one-to-many extension table.
- You use the Type column to group records in a one-to-many extension table.
- You configure each business component to get only the rows of a single type.
- A one-to-many extension table can contain multiple rows for a single row in the base table.
- The name of a one-to-many extension table includes an _XM suffix.
- Similar to a one-to-one extension table, a one-to-many extension table includes a set of generic ATTRIB_nn columns that you can use to store custom data.
- Unlike a one-to-one extension table, the value in the Type property of a one-to-many extension table is Data (Public) rather than Extension.

For more information, see the following topics:

- *Options to Use a Predefined One-to-Many Extension Table*

- *Configuring Objects to Use a One-To-Many Extension Table*
- *Configuring a Business Component*
- *Configuring a Link That Creates a One-to-Many Relationship*
- *Creating a Business Object*

Summary of Support for Extension Tables and Extension Columns

The following table summarizes support for extension tables and extension columns.

Object	Description
Public data table	Can be extended by using an extension table and extension columns.
Private data table	The following support is available for the private data table: • Cannot contain an extension column • Cannot add an extension column to a private data table A *private data table* is a table with the Type property set to Data (Private). Some interface tables are private, but most are public.
Intersection table	The following support is available for an intersection table: • Can be extended with an extension column • Cannot be extended with a custom extension table
LOV Bounded, LOV Type property of a table column	Read-only for a predefined column in Siebel CRM but is editable for a custom extension column. MLOV (multilingual list of values) is allowed with a custom extension column.
Predefined one-to-one extension column	It is recommended that you do not modify or delete a predefined one-to-one extension column.
Predefined extension column	Similar to a data column in a base table, you must not modify or delete a predefined extension column that a predefined Siebel application uses.
Custom extension column	The following support is available for a custom extension column: • You can use the Database Designer to add a custom extension column to a base table. The Database Designer is available in the Tables list in Siebel Tools. The relational database that you use with Siebel CRM determines if you can or cannot create a custom extension column on a base table. • You can add a custom extension column to one of several types of tables. For more information, see *Adding an Extension Column to a Base Table*.
Custom extension table	The following is available for a custom extension table: • You can use the Database Designer to create a new one-to-one extension table. • Several types of custom extension tables are available. For more information, see the table in *Options to Use a Predefined One-to-One Extension Table*.
EIM mapping	The following support is available for Enterprise Integration Manager (EIM) mapping:

Object	Description
	• The EIM Table Mapping Wizard allows you to create or associate a new table with an interface table that uses EIM: o You can create EIM Table Mapping objects to import data to a table that you define. o You can automate the creation of an EIM attribute map on an extension column that Siebel CRM adds to a base table. • You cannot add an EIM mapping for a foreign key relationship to a table that does not contain a user key. For more information, see *Mapping a Custom Table to an Interface Table*.
Custom extension to a dock object	The Dock Object Mapping Wizard allows you to associate a new table with a predefined or a new custom dock object. This support allows Siebel CRM to synchronize data that resides in the dock object of a Remote user.

How an Intersection Table Defines a Many-To-Many Relationship

This topic describes the intersection table. It includes the following information:

- *Overview of an Intersection Table*
- *How Siebel CRM Creates an Intersection Between Tables*
- *How Siebel CRM Creates a Many-To-Many Relationship*
- *Intersection Data in an Intersection Table*
- *How Siebel CRM Uses an Implicit Join With an Intersection Table*

Overview of an Intersection Table

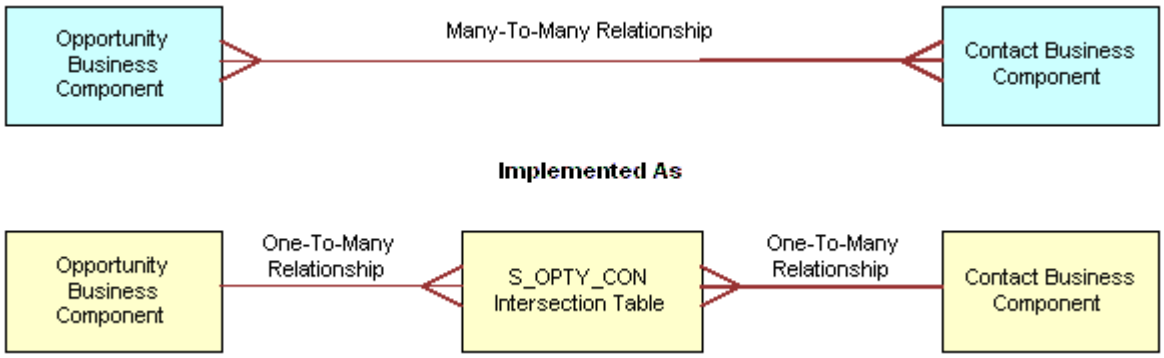
An *intersection table* is a table that defines a many-to-many relationship. It includes an intersection between two business components. A *many-to-many relationship* includes a one-to-many relationship from either direction. For example, a many-to-many relationship exists between Accounts and Contacts. You can view this relationship in the Siebel client:

- The Account Detail - Contacts View displays one account with multiple detail contacts.
- The Contact Detail - Accounts View displays one contact with multiple detail accounts.

Siebel CRM can include the two different views in different business objects. The business objects associate the two business components in opposite directions.

No database construct directly creates a many-to-many relationship. Instead, the Siebel schema uses two links and an *intersection table* to create a many-to-many relationship.

The following figure gives an example of how an intersection table defines a many-to-many relationship.



This figure shows that the many-to-many relationship between Opportunity Business Component and Contact Business Component can be implemented as follows:

- Define a one-to-many relationship between Opportunity Business Component and S_OPTY_CON Intersection Table.
- Define a many-to-one relationship between S_OPTY_CON Intersection Table and Contact Business Component.

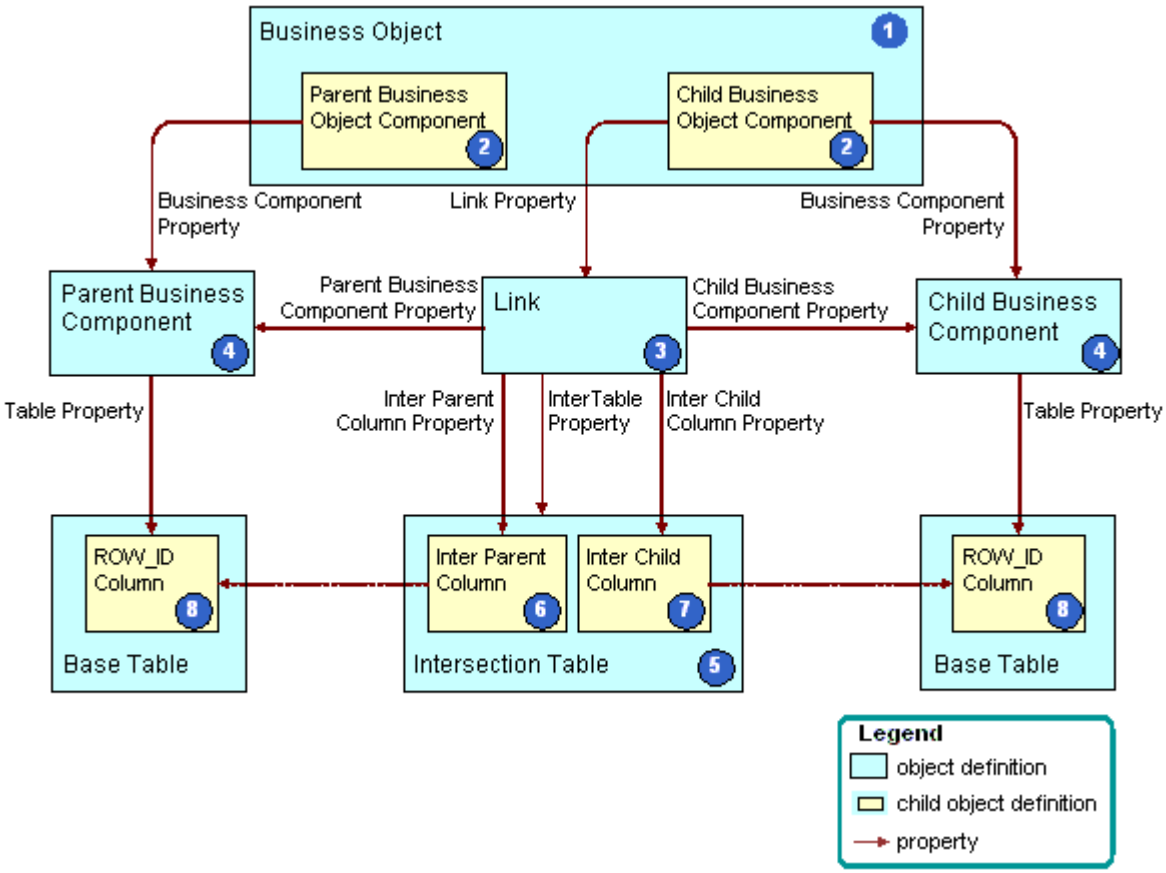
The Type property of an intersection table contains Data (Intersection).

You can add custom extension columns to an intersection table. You cannot use custom extension tables to configure an intersection table. For more information, see *About Links*.

How Siebel CRM Creates an Intersection Between Tables

An *association* is a pair of ROW_ID values, where each value references a row in the base table of a business component. An intersection table contains one row for each association that exists between the row in the base table of one business component and a row in the base table of another business component. The association row in the intersection table stores the ROW_ID values of the row that resides in the base table of each business component.

The following figure describes how Siebel CRM creates an intersection. The associations in the intersection table serve the Opportunity/Contact and the Contact/Opportunity links and their corresponding views. The figure describes how the set of object definitions and relationships pertain to one of two links. The other link uses the same set of object types but with different relationships. Siebel CRM can display one association in both views. For example, the association between Cynthia Smith and Smith Dry Goods.



As shown in this figure, Siebel CRM uses the following objects to create an intersection:

1. **Business object.** References the link that uses the intersection table. It contains the two business components that the link contains. The business object makes this reference indirectly through the child business object component of the business object.
2. **Parent and child business object components.** The Siebel schema uses the business object component to include business components in the business object. The business object component is a child of the business object. The detail business object component references the child business component through the Business Component property. It references the link through the Link property. The parent business object component only references the corresponding business component.
3. **Link.** Creates a one-to-many relationship between the two business components in a specific direction. The properties of the link define one business component as the parent and the other business component as the child in the parent-child relationship.
4. **Parent and child business components.** The Siebel schema specifies two business components in the link. They provide data to the objects that Siebel CRM displays in the parent-child relationship in the client. The base table of each business component contains the ROW_ID column that the Inter Child Column and Inter Parent Column properties of the link reference.
5. **Intersection table.** Contains the associations between rows in the base tables of the parent and child business components. Each row in the intersection table represents one association that exists between the two business components. Two columns in the intersection table serve as foreign keys to the base tables of the two business components. The Inter Parent Column and Inter Child Column properties of the link identify these columns.
6. **Inter Parent column.** Contains the reference to the associated row that resides in the base table of the parent business component. It is identified in the Inter Parent Column property of the link.

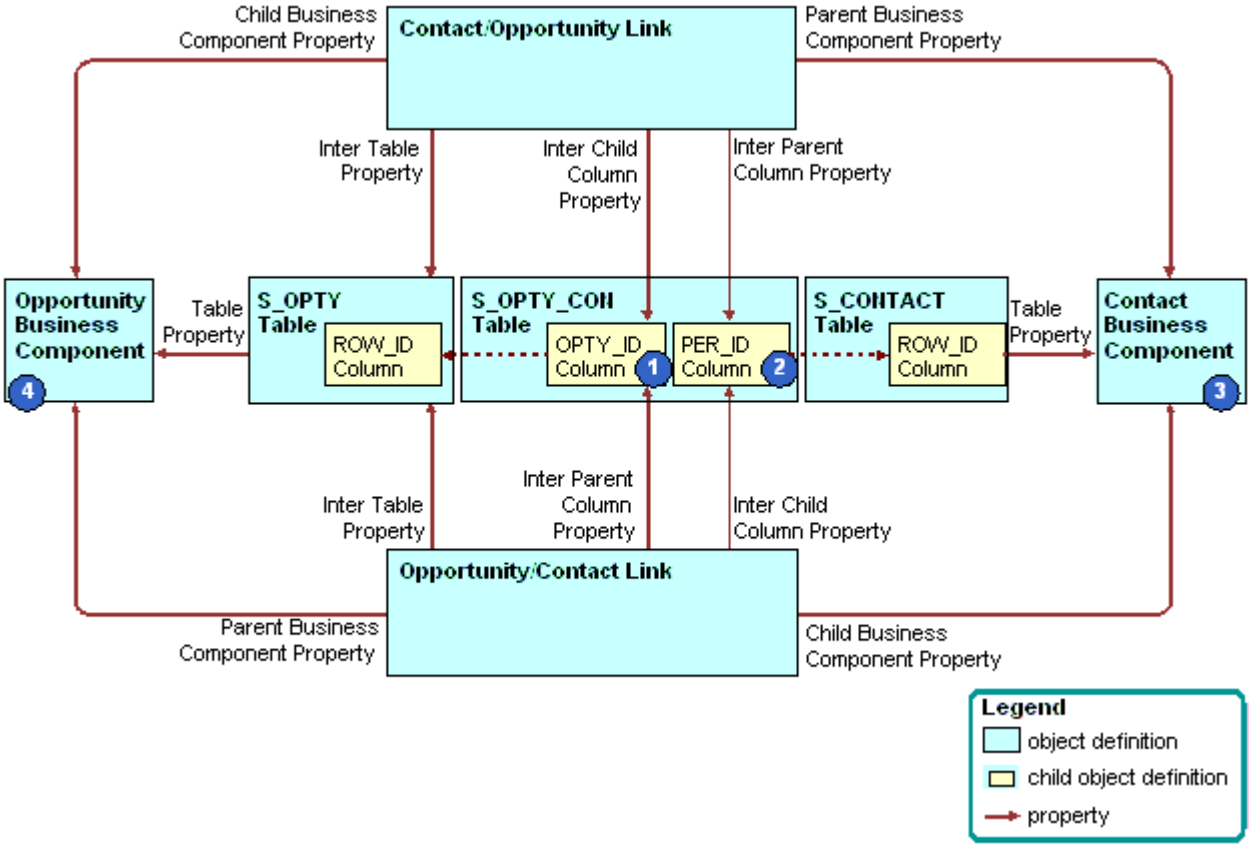
7. **Inter Child column.** Contains the reference to the associated row that resides in the base table of the child business component. It is identified in the Inter Child Column property of the link.
8. **ROW_ID columns.** A unique identifier column for each row that resides in the base table of each business component.

The Siebel schema uses the following properties of the link to create a many-to-many relationship. These properties are empty for a link that does not create a many-to-many relationship:

- Inter Table
- Inter Parent Column
- Inter Child Column

How Siebel CRM Creates a Many-To-Many Relationship

The following figure describes how Siebel CRM uses properties in two links to create a many-to-many relationship. In this example, the relationship is between opportunities and contacts.



Explanation of Callouts

Siebel CRM uses the following objects to create a many-to-many relationship:

1. **OPTY_ID column.** The following properties reference the OPTY_ID column in the S_OPTY_CON table:
    - The Inter Child Column property of the Contact/Opportunity link
    - The Inter Parent Column property of the Opportunity/Contact link

2. **PER_ID column.** The following properties reference the PER_ID column in the S_OPTY_CON table:
    - The Inter Parent Column property of the Contact/Opportunity link
    - The Inter Child Column property of the Opportunity/Contact link
3. **Contact business component.** The following properties reference the Contact business component:
    - The Parent Business Component property of the Contact/Opportunity link
    - The Child Business Component property of the Opportunity/Contact link
4. **Opportunity business component.** The following properties reference the Opportunity business component:
    - The Child Business Component property of the Contact/Opportunity link
    - The Parent Business Component property of the Opportunity/Contact link

Intersection Data in an Intersection Table

An intersection table contains two foreign key columns that create a relationship between the records of two business components. It contains *intersection data columns*, which are columns that contain data that are specific to the intersection.

For example, the S_OPTY_CON table defines the many-to-many relationship that exists between opportunities and contacts. It includes several data columns in addition to OPTY_ID and PER_ID. These data columns contain information about the combination of a opportunity and a contact. Some of these columns include the following:

- **ROLE_CD.** The role that the contact in the opportunity plays.
- **TIME_SPENT_CD.** The time that the contact spends working on the opportunity.
- **COMMENTS.** Comment that is specific to this combination of opportunity and contact.

Some intersection data columns are useful to one parent-child relationship, some are useful to the other parent-child relationship, and some are useful to both of these relationships. For example:

- The ROLE_CD column is useful only in the context of a parent-child relationship that includes an opportunity that is the parent record that includes multiple detail contact records.
- The TIME_SPENT_CD column is useful in the context of either parent-child relationship. Each contact fulfills a unique role in the opportunity. The time spent can be useful if viewed from one of the following perspectives:
    - Time spent with each contact of an opportunity
    - Time spent with each opportunity of a contact

How Siebel CRM Uses an Implicit Join With an Intersection Table

To access an intersection data column, the Siebel schema uses a business component field that uses a join. An implicit join exists for any intersection table. It includes the same name as the intersection table. It exists for the child business component. If Siebel CRM creates a link that uses an intersection table, then it creates the implicit join. For example:

- The schema references the ROLE_CD column of the S_OPTY_CON table to the Role field that resides in the Contact business component.
- The Join property of the Role field contains S_OPTY_CON.
- The Contact business component does not contain a child S_OPTY_CON join object definition.

The Siebel schema includes the join. This join is not visible in the Object Explorer. This situation is similar to the implicit join that exists for a one-to-one extension table. You can use an implicit join to update data.

About Columns and Indexes in a Siebel Table

A *column* is a representation of the physical column that resides in the underlying database management system. The Siebel schema records the name, data type, length, primary key status, foreign key status, alias, and other properties of the database column as properties in the corresponding object definition of the column. The schema includes other properties that are internal to Siebel CRM in the object definition, such as the Changed status, Inactive status, and the Type. For more information, see the following topics:

- *Properties of a Table Column*
- *How a CIAI Index Can Improve a Query*.

Data Columns of a Siebel Table

A *data column* is a column that provides data for a field. It can serve as a foreign key that references a row in another table. Most columns in Siebel CRM are data columns. A data column is sometimes referred to as a base column. A data column can be public or private. You cannot modify the properties of a data column.

Extension Columns of a Siebel Table

An *extension column* is a column that stores custom data. Siebel CRM supports the following types of extension columns:

- **Predefined extension column.** Included in a predefined extension table. Siebel CRM names these columns ATTRIB_*nn*, where *nn* is a value between 01 and 47. For example, ATTRIB_13. It is recommended that you do not modify or delete a predefined extension column.
- **Custom extension column in an extension table.** Added by a developer to an extension table. Siebel CRM names these with an X_ prefix.
- **Custom extension columns in a base table.** Added by a developer to a base table. The relational database system that you use with Siebel CRM determines if this configuration is allowed or not allowed. If the database system supports a custom extension column in a base table, it might be preferable for performance reasons to add it to the base table rather than to add it to an extension table. Performance might be affected if you add an extension column to an extension table because Siebel CRM creates extra SQL to join the extension table.

System Columns of a Siebel Table

A *system column* is a column that Siebel CRM displays in all tables, but it does not include the same set of system columns in every table. You can use the data in a system column for various reasons. For example, you can use the ROW_ID column to create a join. Most system columns are read-only. You typically must not modify the data in a system column. Some exceptions exist, such as using certain system columns in an interface table. For more information, see *System Fields of a Business Component*.

The following table describes some of the system columns that Siebel CRM commonly uses.

Column	Description
ROW_ID	Stores a unique, base 36 alphanumeric identifier for the rows in the table. ROW_ID is present in all tables. It is the typical destination column of a foreign key relationship from another table. In a predefined data table, the Id field often represents ROW_ID for use in a join or link. For example, the Id field in the Account business component represents the ROW_ID column in the S_ORG_EXT table. For more information, see *Relationship Between a System Field and a System Column*.

Column	Description
CREATED	Stores the creation date and time of each record.
CREATED_BY	Stores the ROW_ID of the S_USER record of the person who created the record. This is not the user name that the user enters when the user logs in to Siebel CRM.
LAST_UPD	Stores the date of the last update that Siebel CRM performed for the record.
LAST_UPD_BY	Stores the ROW_ID of the S_USER record of the person who last updated the record. This is not the user name that the user enters when the user logs in to Siebel CRM.
DB_LAST_UPD	Stores the date of each record that Siebel CRM updates in the database. DB_LAST_UPD is different than LAST_UPD. For example, if the user updates a record, then Siebel CRM updates the LAST_UPD and DB_LAST_UPD columns in the local database. If the user synchronizes with a Server database, then Siebel CRM only updates the DB_LAST_UPD column.
PAR_ROW_ID	Stores a foreign key to the ROW_ID column of the base table. Siebel CRM includes the PAR_ROW_ID column in extension tables, file attachment tables, and tables whose name contains a _T suffix.

Siebel CRM updates the following columns:

- CREATED
- CREATED_BY
- LAST_UPD
- LAST_UPD_BY
- ROW_ID

The following columns store the date, time, and user values for the client. They do not store the date, time, and user values for the Siebel database:

- CREATED
- CREATED_BY
- LAST_UPD
- LAST_UPD_BY

Indexes of a Siebel Table

An *index* is a logical representation of a physical index that resides in the underlying database management system. Siebel CRM includes a set of predefined indexes. The name for each index contains an S_ prefix. You must not modify or delete a predefined index. You can create a custom index. For more information, see *Properties of an Index of a Siebel Table* and *Creating a Custom Index*.

Index Columns of an Index

An *index column* is a child object of the index object. The object definition for an index column associates one column to the parent index. For more information, see *Properties of an Index Column* and *Creating a Custom Index*.

How a User Key Creates a Unique Set of Values

A *user key* is a key that specifies columns that must contain unique sets of values. The purpose of a user key is to prevent the user from entering duplicate records. You can use it to determine the uniqueness of records during a data import operation in Enterprise Integration Manager.

The name of the parent table of the user key that contains an _U*n* suffix designates the user key. For example, S_PROD_INT_U1. Each user key includes User Key Column child objects that define the table columns that must include unique values. For example, BU_ID, NAME, and VENDR_OU_ID in the S_PROD_INT_U1 user key.

A predefined index exists for each predefined user key. This index uses the following format:

```
S_TABLE_NAME_Un
```

You cannot add or modify a user key that resides in a predefined Siebel table or an EIM base table. For help with remapping data to meet your business requirements, see *Getting Help From Oracle*.

For more information, see *About Interface Tables*.

How the S_Party Table Controls Access

The party model organizes entities such as Person, Organization, Position, and Household. A party always represents a single person or a group that Siebel CRM can translate to a set of people, such as a company or a household. Siebel data access technology uses this party model. Some parts of the data objects layer use the party model to abstract the difference between people, companies, households, and other legal entities. This model covers the relationships that exist between your company and people, such as contacts, employees, partner employees, and users, and other businesses, such as accounts, divisions, organizations, and partners. Siebel CRM uses the S_PARTY table as the base table for this access. The Siebel schema implicitly joins related tables as extension tables.

The following information lists the extension tables and their corresponding EIM interface tables. A *party table* is a table that holds party data. Some example party tables include S_CONTACT, S_ORG_EXT, S_USER, and S_POSTN.

Data Type	Extension Table to S_PARTY	EIM Interface Table
Accounts	S_ORG_EXT	EIM_ACCOUNT
Business Units	S_BU	EIM_BU
Contacts	S_CONTACT	EIM_CONTACT
Employees	S_CONTACT	EIM_EMPLOYEE
Households	S_ORG_GROUP	EIM_GROUP
Positions	S_POSTN	EIM_POSITION

Data Type	Extension Table to S_PARTY	EIM Interface Table
Users	S_USER	EIM_USER

The Siebel schema implicitly joins these extension tables to the S_PARTY table, so they are available through the S_PARTY table. The PARTY_TYPE_CD column of the S_PARTY table supports the following types:

- AccessGroup
- Household
- Organization
- Person
- Position
- UserList

Guidelines for Using the S_PARTY_PER and S_PARTY_REL Tables

The predefined S_PARTY_PER and S_PARTY_REL intersection tables create a many-to-many relationship between party business components, such as Account and Contact. The table you use depends on whether you must or must not enforce access control.

You can use the S_PARTY_PER table to create a many-to-many relationship between two party business components where you must create access control. Records in the S_PARTY_PER table provide data access rights from the parent to the child parties. To maintain a good response time with a query that constrains visibility, you must minimize the number of rows that the S_PARTY_PER table contains. If you create a many-to-many relationship where you do not require access control, such as if you create a recursive many-to-many relationship between a party business component and itself, then it is recommended that you use the S_PARTY_REL table.

For example, you can use the S_PARTY_PER table to create a relationship between the following items:

- Access groups and members
- Accounts and contacts
- Employees and positions
- User lists and users

If you must configure tables in the party model, then you must create an extension table from the S_PARTY table. For example, S_CONTACT is an extension table of the S_PARTY table. The S_CONTACT table is an Extension (Siebel) type, so you cannot use it as a base table for an extension table. You must create an extension table and use the S_PARTY table as the base table. To display data from the new extension table, you can create an explicit join that brings data from the new extension table to the business component you are using.

For more information about the party model, see *Siebel Security Guide* .

Options to Configure the Data Objects Layer

This topic describes options to configure the data objects layer. It includes the following information:

- *Options to Configure Predefined Objects and Perform Advanced Configuration*

- *Options to Use a Predefined One-to-One Extension Table*
- *Options to Use a Predefined One-to-Many Extension Table*

For more information, see the following topics:

- *Configuring Tables*
- *Properties of a Siebel Table*

Options to Configure Predefined Objects and Perform Advanced Configuration

This topic describes options that are available to you to configure predefined objects and to do advanced configuration.

Options to Configure a Predefined Database Object

You can configure a predefined extension table or column that is available for you to use for your own purposes. These tables and columns provide the easiest option to store more entities because they are already part of the data objects layer. Using them does not require you to modify the logical schema. The following predefined extensions are available:

- Extension columns
- One-to-one extension tables
- One-to-many extension tables

For more information, see *Options to Use a Predefined One-to-One Extension Table* and *Options to Use a Predefined One-to-Many Extension Table*.

You can use the Database Designer to add an extension column to a base table or to create a new one-to-one extension table. For more information, see *Adding an Extension Column to a Base Table*.

Options to Perform Advanced Configuration of Database Objects

You can use the New Table Wizard to create the following types of tables:

- Stand-alone table
- One-to-one extension table
- One-to-many extension table
- Intersection table

For more information, see *Using the New Table Wizard to Create a New Table*.

You can use the EIM Table Mapping Wizard to map an extension to an interface table. This wizard allows you to create or associate the new table to the interface table that uses EIM. You can create EIM table mapping objects that import data to tables you define, and you can automate how Siebel Tools creates an EIM attribute map on an extension column that you add to a base table. For more information, see *Mapping a Custom Table to an Interface Table for Siebel EIM*.

You can use the Dock Object Mapping Wizard to map an extension to a dock object. To support data synchronization to Remote users, this wizard allows you to associate the new table with a predefined or custom dock object. For more information, see *Configuring Dock Objects for Siebel Remote*.

Options to Use a Predefined One-to-One Extension Table

Siebel CRM uses one-to-one predefined extension tables for many of the predefined data tables. The predefined extension table contains columns of various types that possess a predefined one-to-one relationship with a base table. This base table uses more columns in the extension table for new functionality without modifying the base table or database schema. For more information, see *Guidelines for Modifying a Predefined One-to-One Extension Table*.

A one-to-one predefined extension table does not require you to create a new business component because Siebel CRM implicitly defines this type of table as a join. For more information about implicit joins, see *How an Extension Table Stores Custom Data*.

A one-to-one predefined extension table includes an _X suffix, such as S_PROD_INT_X. Siebel CRM names the columns that these tables contain with ATTRIB_*nn*, where *nn* is a value from 01 to 47.

The following table lists the different data types in a Siebel extension table and the number of columns of each data type.

Data Type	Number of Columns
Number	12
Date	10
Varchar(255)	1
Varchar(100)	5
Varchar(50)	10
Varchar(30)	5
Char(1)	4

Determining Availability of a Predefined Extension Column

You must determine whether a predefined Siebel application uses or does not use the table column before you use a predefined extension table. To do this, you search the Siebel repository for fields that Siebel CRM associates with the column.

CAUTION: If a predefined field references a column, then do not deactivate the field.

To determine availability of a predefined extension column

1. In Siebel Tools, in the Object Explorer, click the Flat tab.
2. In the Object Explorer, click Field.
3. In the Fields list, create a query using values from the following table.

Property	Value
Column	Name of the column you must use.
Join	Name of the extension table you must use.

4. If the query does not return any Field object definitions, then Siebel CRM does not use the column in the extension table and it is available.
5. If the query returns one or more object definitions, then you must find another extension column in this table. To identify the extension columns that Siebel CRM currently uses, do the query again using values from the following table.

Property	Value
Column	ATTRIB*
Join	Name of the extension table you must use.

Options to Use a Predefined One-to-Many Extension Table

More than 20 predefined tables exist that contain a one-to-many relationship with a base table. These tables include the _XM suffix. They include generic columns that you can use to store more data. They allow you to track entities that do not exist in a predefined Siebel application, and they include a one-to-many relationship to a predefined base table. The extension tables themselves are already part of the data objects layer, so you are not required to modify the database schema. For more information, see *How an Extension Table Stores Custom Data*.

Guidelines for Configuring the Data Objects Layer

This topic describes guidelines to configure the data objects layer. It includes the following information:

- *Overview of Guidelines for Configuring the Data Objects Layer*
- *Guidelines for Creating a New Table*
- *Guidelines for Adding an Extension Column to a Base Table*
- *Guidelines for Creating a Custom Index*
- *Guidelines for Creating a Function-Based Index*
- *Guidelines for Creating a LONG Column*
- *Guidelines for Modifying a Predefined One-to-One Extension Table*
- *Guidelines for Creating a Custom One-to-One Extension Table*

- *Guidelines for Configuring a Base Table or Configuring a One-To-Many Extension Table*
- *Guidelines for Configuring a Foreign Key That Affects Enterprise Integration Manager*
- *Guidelines for Creating a Custom Docking Rule*

For more information, see *Guidelines for Reusing a Predefined Table*.

Overview of Guidelines for Configuring the Data Objects Layer

If you configure the data objects layer, then use the following guidelines:

- Do not modify a predefined base table or the columns of a predefined base table.
- Do not modify a predefined one-to-one extension table or the column of a predefined one-to-one extension table. For more information, see *Options to Use a Predefined One-to-One Extension Table*.
- The predefined user interface that Siebel CRM displays in the Siebel client does not use all of the relationships that are available in the underlying data objects layer. Most entity relationships are available for you to use. It is recommended that you use predefined objects in the data objects layer, if possible.
- To minimize the effect of your modifications on other developers, make any bulk modifications to the Siebel schema at the beginning of each project phase. If you make modifications during a project phase, then you must distribute these modifications to all other remote users. You can use Siebel Anywhere to distribute a schema modification. Otherwise, you must create a new database extract for each remote user before you can progress to the next phase.
- If your deployment runs in a DB2 environment, then do not create a column that contains a name that is longer than 18 characters.
- The data objects layer includes over 2,000 database tables. Each of these tables uses a consistent naming format to help you identify each individual table. For information on naming formats for tables, see *About Siebel Tables*.

Guidelines for Creating a New Table

If you create a new table, then use the following guidelines:

- You can only create the following types of tables:
  - Data (Public)
  - Data (Intersection)
  - Extension
- You must explicitly grant permissions on any table that you define.
- Create a new table only after you explore other ways of meeting your business requirements, such as using a predefined extension table.

Guidelines for Adding an Extension Column to a Base Table

You can add an extension column to a predefined base table. Adding an extension column avoids having to add another join to an extension table to store custom data. You can add an extension column to any of the following table types:

- Data table

- Intersection table
- Interface table
- Predefined extension table
- Custom extension table
- Extension (Siebel) table

You cannot add an extension column to a private data table that contains a value of Data (Private) in the Type property. Some interface tables are private, but most are public. Use the following guidelines if you add a column to a table:

- Any column you add must conform to the data type limitations of all the RDBMS types that your enterprise uses. Consider your server database and any regional or remote databases.
- If you add a new column to a predefined table with one or more rows of data, then the RDBMS does not allow you to add the column unless you include a default value.
- You cannot remove a column after you add it to a table. For more information, see the documentation for your database technology.
- If you add a column to a table, then do not use a column name that includes a word that is reserved on your server or client database. If you use an underscore (_) at the beginning and end of the reserved word, then you can use a reserved word. For more information, see *Naming Format for a Siebel Table*.
- If you create a new extension column in the Siebel schema, then padding problems might occur with Siebel Remote. For more information, see *Siebel Remote and Replication Manager Administration Guide* .

Guidelines for Creating a Custom Index

You can create a custom index to improve performance. If you create a custom index, then use the following guidelines:

- If you create a custom table, then the custom table typically requires new indexes.
- Use caution if you create an index. A custom index can result in a situation where objects reference the custom index instead of the predefined indexes. This situation can result in poor performance.
- If at some point you no longer require a custom index that you have defined, then do not delete it from the Siebel repository. Instead, you can deactivate it. Make sure the Inactive property of the index contains a check mark.
- You must thoroughly test any custom index in a test environment before you implement it in a production environment.
- In a DB2 environment, do not create an index that contains a name that is longer than 18 characters.
- For Oracle Database and DB2, you can create function-based indexes. For more information, see *Guidelines for Creating a Function-Based Index*.

Guidelines for Creating a Function-Based Index

As of Siebel CRM 22.8 Update, developers can work with their database administrators to create function-based indexes in Siebel Tools where they are deemed appropriate for performance reasons in your deployment. Examples where function-based indexes can improve query performance include:

- Operations on multiple columns

- Case insensitivity
- Partial content searches (for example, the last four digits of a U.S. Social Security Number)

Note: Function-based indexes are currently supported for Oracle Database and IBM DB2 databases only.

Function-based indexes are created in the Siebel repository. The user interface for creating a function-based index for a table is the same as that for creating a standard index in Siebel Tools. Once a function-based index has been created, all of the Siebel CRM utilities and other functions that manage the database schema will respect these indexes, such as PostInstallDBSetup, Apply DDL, Generate DDL, the Siebel Migration application and related REST APIs, DDLImp, DDLExp, and other command-line utilities.

CAUTION: It is not supported to manually create your own function-based indexes outside of the Siebel repository or using methods other than those described here. All changes to the Siebel CRM database schema must be made through the Siebel repository.

Note: Like other indexes that you create for Siebel CRM, function-based indexes are intended to be created only as result of a specific performance issue for some common query that users encounter. Your Siebel CRM development team creates function-based indexes only with the participation of your database administrator.

After the initial creation of a function-based index, this index would not need to be recreated to stay up to date, because when each database record is subsequently created or updated, the query function (expression) is calculated and the result is stored in the index at that time.

More specifically, when a query performance issue is identified, the developer contacts the database administrator to analyze existing indexes on a given table and determine whether it might be appropriate to address the performance issue using a function-based index. Once the database administrator has defined the desired solution, the developer defines the index in the Siebel repository based on the information provided by the database administrator. The developer then applies the data definition language (DDL) directly or generates the DDL and provides the DDL file to the database administrator to apply.

In defining a function-based index, the developer adds one or more index column objects. In addition to those index columns containing expressions (functions), other index columns can be added that specify physical table columns that might be part of the same query. Note the following:

- For index columns in support of a function-based index, the Expression Flag property must be checked and a value must be specified for the Expression property.
- For index columns based on physical columns, leave Expression Flag unchecked and do not specify an expression.

In an index column containing an expression, a valid Column Name value must be specified (due to existing unique key constraints on the table), but the actual column is ignored when the index is generated. Where Expression Flag is checked and the Expression property is set to the database expression or function you require for this index, when the index is generated, this expression is used instead of the column specified in Column Name. See also *Siebel Object Types Reference* .

Also consider the following information:

- The information provided by the database administrator representing the solution to the query problem might need to be divided into separate parts, to be mapped to different index column objects when the index is defined (where, for example, some index column records might include expressions and some physical columns).
- Each expression in a function-based index must not exceed 255 characters.

- Function-based index support is added through installation of Siebel CRM 22.8 Update or later. When PostInstallDBSetup runs, it makes the changes to the Siebel repository necessary to store the Expression and Expression Flag data in the repository table S_INDEX_COLUMN, which stores all index columns. No further customer action is required to enable this feature.

Example of a Function-Based Index

The following is a hypothetical example scenario of a customer determining a need and creating a function-based index to improve the query performance. Assume a base table `CONSULTANT_USAGE` that is used to store usage information about consultants working on projects. For each consultant in the table working on a particular project, the Days column stores the number of days the consultant is expected to spend on the project. The Rate column stores the daily rate for this consultant. The table stores 1,000 such consultant records for this large project.

In this example, project managers frequently query this table to identify high-value consultants for whom the number of project days multiplied by the consultant's rate is greater than $10,000. Without a function-based index created to query for this information, the database engine must load all 1,000 records in the table and the expression of Days multiplied by Rate must be calculated before the engine can filter out the records that do not qualify, as follows:

```
SELECT * FROM CONSULTANT_USAGE WHERE DAYS*RATE > 10000 WHERE PROJECT_ID = 'x'
```

As the size of this table increases, the performance of this query might worsen and it might be deemed a candidate for an index. Upon examination, the database administrator would note that a calculation is being performed in the `where` clause and might conclude that this is the cause of the performance problem and recommend a function-based index including the expression `DAYS*RATE`. If such a function-based index is created, then the query can be created to directly load only those records where `DAYS*RATE > 10000`, which is a much more efficient and better-performing query.

Guidelines for Creating a LONG Column

“LONG” columns are obsolete on all database platforms.

Selecting the “Long” datatype when defining a Column object will be treated as a CLOB or NVARCHAR(max) as appropriate for your database platform.

Guidelines for Modifying a Predefined One-to-One Extension Table

It is strongly recommended that you add custom extension columns to the base table to store your data instead of storing frequently accessed data in columns in a one-to-one extension column. It is strongly recommended that you do not modify a predefined one-to-one extension table or the column of a predefined one-to-one extension table for the following reasons:

- Some of the columns that reside in a predefined extension table are not available to use because Siebel CRM uses them. You must not modify or delete an extension column that Siebel CRM uses.
- A C++ class might use the extension table in a reserved way. Modifying this table might cause behavior that you cannot predict.
- An upgrade effort might use the extension table, so you cannot predict future use of this table.
- Docking rules use some extension columns, so these columns are reserved for use with Siebel Remote. For more information, see *Configuring Dock Objects for Siebel Remote*.

- Use of an extension table affects performance because Siebel CRM must include the table in all queries that use the field that Siebel CRM uses to run the query. This situation can become a problem if Siebel CRM joins the table to multiple business components, specifically if a number of extension tables are in use.

It is permissible to use a predefined one-to-one extension table in the following situations:

- When a custom column is only used in very specialized situations.
- If the implementation of a database constraint is beneficial. For example, to realize the improved performance that results when maximum bytes in a row are used before record chaining occurs.

Guidelines for Creating a Custom One-to-One Extension Table

If you create a custom one-to-one extension table, then use the following guidelines:

- If you must configure a table whose type is Extension or Extension (Siebel), then you must extend from the base table of the table, not from the extension table. The Base Table property of the extension table describes the base table to extend. For example, the S_CONTACT table is an extension table of the S_PARTY table. The S_CONTACT table is an Extension (Siebel) table, so you cannot use it as the parent table for an extension table. Instead, you can extend the S_PARTY table and use an implicit join to display the data from the extension table.
- A custom one-to-one extension table does not require new docking rules because the Siebel schema implicitly routes the data that this table contains according to the docking rules that the parent table specifies.

Guidelines for Configuring a Base Table or Configuring a One-To-Many Extension Table

You can use the following guidelines to help you decide to add an extension column to a base table or to use columns in a one-to-many extension table:

- Try to use a predefined one-to-many extension table or column to meet design requirements. They are predefined and already part of the data objects layer so they do not require you to modify the Siebel schema or the physical database. If a predefined extension table or column is not available, then explore other options, such as creating a new extension table.
- Add an extension column to a base table if the data you must store almost always exists for a base record, and if Siebel CRM does not regularly access it. This configuration often results in better performance because it avoids the join that an extension table uses. It can result in slower access to the base table if a lot of data exists where numerous large fields are added and where these fields always contain data. In this situation, fewer rows fit on one page.

  If a user query regularly includes an extension column, then it is likely that an index is required on the column that Siebel CRM must include on another base table column. You must add it to the base table.
- If one-to-many extension fields are required, and if the user only infrequently accesses the view that displays this data, then you can use columns in a one-to-many extension table. In this situation, Siebel CRM uses the join for the extension table, but only if the user accesses this view.

Guidelines for Configuring a Foreign Key That Affects Enterprise Integration Manager

Use caution if you configure an extension column to contain a foreign key. An extension column that contains a foreign key might be appropriate if it references a business object that is visible to the enterprise. You must avoid an extension column that contains a foreign key if it references a business object whose visibility is limited, such as Opportunity, Contact, Account, or Service Request. Using an extension column as a foreign key column can cause problems if Siebel CRM creates an EIM mapping or if it routes data to a remote user.

You cannot configure EIM to import data to a foreign key column because you cannot configure the required EIM object types.

You cannot add an EIM mapping for a foreign key relationship to a table that does not include a user key.

Guidelines for Creating a Custom Docking Rule

If your enterprise uses the Siebel Mobile Web Client, then note that Dock Object Visibility rules determine how Siebel CRM downloads data to the local database. These rules use predefined relationships to identify the data that Siebel CRM uses from the tables to help it route data to the local database that the remote user uses.

If you create a new relationship, then no Dock Object Visibility rules exist that allow Siebel CRM to download relevant data to the local database. This situation might result in a user who cannot view data. To resolve this problem, you can use the Docking Wizard to create custom docking rules for custom foreign keys. To avoid performance problems with the Transaction Processor and Transaction Router, you must analyze how your configuration affects performance before you create a new Dock Object Visibility rule or object.

If you add a rule, then you might inadvertently add a significant number of database records for Remote users. This configuration might affect initialization and synchronization performance. An increased number of records in the Remote database might affect performance.

For more information, see *Configuring Dock Objects for Siebel Remote*.

Limitations on Use of Direct SQL Against Siebel Databases

This topic explains when it's allowable to perform "Direct SQL" against Siebel Databases. It should be noted immediately that most direct SQL is forbidden and should not be considered an option. Following are some reasons:

- Siebel Remote, Replication Manager, and Mobile require that all changes to data are captured in the Docking Transaction log (S_DOCK_TXN* tables). While the Object Managers and EIM ensure that this happens, a direct SQL statement against the database will not.
- The Siebel ERD is massive, with over 5,000 tables and 20,000 foreign keys ready-to-use.
    - Due to the schema objects defined in the Repository, the Object Managers and EIM can ensure referential integrity for these foreign keys, which would be incredibly difficult to do manually.

    - Data is denormalized into various base and intersection tables–it would be very easy to miss (or do incorrectly) and cause data corruption.
- Many of the validation rules (such as "numeric value between 0 and 100") in Siebel CRM are maintained at the Business Object Layer–issuing direct SQL would bypass those validation rules, allowing invalid data into the database.
- The Siebel utilities that manage the coordination of the physical database schema and the logical Siebel Repository schema are intended to be "self healing"–for example, if there are columns, indexes, or other items in the physical layer that aren't in the logical layer, they will be removed, so all schema changes must be made in the Siebel Repository.
- Oracle Support can't trace what happened to produce unusual data or diagnose why unwanted behavior is occurring when data was modified through direct SQL.

For more information, see the following topics:

- *Definitions*
- *Allowed and Forbidden SQL Usage*
- *Exception Process*

Definitions

This topic explains some important definitions.

- **Siebel Repository**- The metadata where the Siebel configuration is stored (including schema object definitions).
- **Logical Schema**- The description of the schema objects as defined in the Siebel Repository.
- **Physical Schema**- The actual schema as defined in the underlying database.
- **Table Types**- Siebel CRM tables can be divided into a few categories.

Type	Description	"Type" as defined in the Siebel Repository
EIM	Staging tables for getting data into or out of the Siebel base tables. There is typically at least one EIM table for every frequently-used base table.	Interface
Base	Tables in which "end-user" data would appear. Examples include customer accounts, opportunities, or service requests.	Data (Public) and Data (Intersection)
System	Tables that support the infrastructure of the behavior of the product, such as the Server Request tables, Siebel Remote Transaction Logs, etc.	Data (Private)
Repository	The tables in which the Repository metadata is stored.	Repository

- **Data Definition Language (DDL)**- This refers to SQL statements that can change the physical schema of the data, such as creating new tables or indexes, adding columns to tables, etc. DDL statements are issued at specific times, such as immediately after making a change in the logical schema, or during an Upgrade, Update, or Migration.
- **Data Manipulation Language (DML)**- This refers to SQL statements that change the data within the database, such as creating, modifying, or deleting a new account, service request, or opportunity record. DML is constantly being executed as users use the system and are updating their data.
- **CRUD**- This is an acronym for the four types of DML statements: Create, Read, Update, and Delete.

Allowed and Forbidden SQL Usage

Allowed SQL

Data Definition Language (DDL)

Use Case	Description/Comments
Test an Index	When a database performance issue is identified, a customer may request that their DBA investigate the cause, and this is often found to be something that can be fixed with a new index. In these cases, the DBA will typically create a *temporary* index in the physical schema to see if it solves the problem and does not introduce new ones. This may take several iterations and the DBA is permitted to do this with direct DDL. NOTE: Once the correct indexes are identified, the DBA must provide the final definition to the Siebel team to formalize in the Siebel Repository. Failure to do this will result in the index being removed automatically the next time a DDLIMP process executes.
Pre-execute DDL changes from an Oracle-provided Siebel utility.	When a process requires making changes to the database schema, such as Migration or utilities associated with Monthly Updates, the DBA may apply such changes in advance, for example: • To avoid potential downtime due to table locks for high usage tables by making the changes during low usage or scheduled downtime. • For many customers–either for security or other reasons–the "Siebel Team" does not have tableowner access required to execute DDL–only a DBA has that privilege (and is not negotiable with the customer). All Siebel utilities provide basic DDL for schema changes to enable this option. • DBAs *may* modify the DDL for database storage requirements–such as specifying alternate tablespaces, extents, etc., but may not make changes to table or index structure. • Note that in the cases where DBAs will be applying DDL, those DDL changes must be applied before attempting to continue to execute the remaining steps in a given process--For example, it is not acceptable to run the non-schema portions of PostInstallDBSetup *before* the schema portions.
View creation	The DBA may create read-only database views against any table to facilitate with integration with other systems, External Business Components, Virtual Business Components, or any other reason.
Temporary Tables	The DBA may create *temporary* tables as required for a temporary purpose. Examples:

Use Case	Description/Comments
	• Assist in getting data into an EIM table, for example, SELECT from a base table into a temporary table, then manipulate that data with the eventual goal being to get it into the corresponding EIM table. • Facilitating execution of an Oracle-provided Siebel utility. Note that this is a very unusual corner case, but an example is that "WFCleanup" needs a temporary table to execute and there is a known bug where that table is sometimes not automatically created.

Data Manipulation (DML)

Operation	Description	Example	Allowed
C	Create	INSERT INTO...	Inserts into EIM tables (only) are allowed. Note that since Reads are always allowed (even against base tables), it is acceptable to have a statement of the form "INSERT INTO EIM_x SELECT ... FROM S_..."
R	Read	SELECT ... FROM ...	Selects from *any* table are allowed (including base tables), as SELECTs do not affect data.NOTE: Too many SELECT statements may impact performance, but from an "allowed/not allowed" perspective, they are allowed.
U	Update	UPDATE ...	Updates to EIM tables (only) are allowed.
D	Delete	DELETE FROM...	Deletes from EIM tables (only) are allowed.

Forbidden SQL Usage

Data Definition Language (DDL)

Use Case	Description/Comments
Ad hoc tables, columns, etc. not defined in the Repository.	NOTE: Failure to formalize the objects in the Repository *will result in the objects being dropped during execution of certain Siebel utilities.*
Attempt to work around functionality not available in Siebel Tools/Web Tools	The DBA cannot create (nor modify) schema objects in a way that cannot be done in Siebel Tools/Web Tools. Rephrasing: If you cannot make a change in Siebel Tools/Web Tools, your DBA cannot do it at the physical layer via DDL. Examples: • Changing the size of an OOTB column directly at the database layer. It is not possible to change the length via Tools, therefore it is not supported via DDL. • Modifying an OOTB index (such as adding index columns, sort order, etc.) • As of August 2024, there is no support for Function-Based Indexes on the Microsoft SQL Server platform–an attempt to work around this at the physical database layer is not supported. • Modification of OOTB Unique Indexes ← this is particularly dangerous • Deleting OOTB Indexes

Use Case	Description/Comments
Delete OOTB objects	An attempt to delete or modify an out-of-the-box object directly at the database level (without making a change in the Repository) will result in it being put back the next time a Siebel DB utility runs. Note: If an Index is causing a performance problem, it should be inactivated in the Repository once testing is complete; if not, it will be restored the next time a Siebel DB utility runs.
Index creation/modification at physical layer *without eventually being reflected in the Repository.*	This is not permitted–as noted above, it is fine to *test* indexes using DDL, but they must be formalized in the Repository. NOTE: Failure to formalize the index in the Repository *will result in the index being dropped*.

Data Manipulation (DML)

Operation	Description	Example	Forbidden
C	Create	INSERT INTO...	Inserts into non-EIM tables are never permitted.
R	Read	SELECT ... FROM ...	There are no restrictions on SELECT statements, but note that too many SELECT statements may impact performance, but from an "allowed/not allowed" perspective, there is no restriction.
U	Update	UPDATE ...	Updates to non-EIMs table are never permitted.
D	Delete	DELETE FROM...	Deletes from non-EIM tables are never permitted.Note that this includes "cleaning up" tables that tend to grow "forever", such as the Audit Log, Transaction Log, etc. Rather than delete, it is important to find out *why* such things are happening and eliminate the cause.

Exception Process

There are two primary options available for exceptions to the above.

Proof of Concept (POC)

A "POC" is a "hotfix" for a given Siebel CRM Monthly Update. They provide one-off, binary-packaged solutions for issues where data needs to be cleaned up due to a product defect. For example, if an Upgrade causes an issue in a customer environment that is unique to that customer, Oracle may provide an executable that will adjust data in the customer's database or Repository to unblock the customer. This is usually customer-specific, but could be widely distributed.

> **Note:** If a POC is provided, the customer must run the POC (not attempt to make manual, direct SQL statements) following the instructions provided with the POC release documentation.

A POC will only be built in response to a data corruption problem caused by a Siebel CRM product defect–a POC will not be provided to clean up data in the case where a customer caused the data corruption, such as the case where the customer used direct SQL to update data.

Non-Standard Change Request (NSCR)

The remaining option for an exception to the policies above is a "Non-Standard Change Request" or "NSCR". To get an NSCR approved, the customer must conduct a paid engagement with the Oracle CSS Team (formally known as "Advance Customer Support" and "Expert Services"). A qualified specialist from that team will review the request and determine if the only option would require deviating from the rules above.

At that point, the specialist will submit the request to Oracle's Siebel CRM Data Modelling Group for analysis and approval. The decision to approve (or not) will be based on several criteria:

- Is there a normally supported path to do this (e.g., through Tools/Web Tools)?
- Any obvious issues, such as possible data corruption/integrity, failure to comply with unique indexes/user keys, and so on.
- Possible performance impacts–may not be a blocker, but may require performance testing results.
- Ability to Upgrade/Update–for example, changing an Index that we could anticipate changing on the Oracle side or an attempt to change a Column Type or Length that could cause issues later.
- Logic (is this the right way to do this?)

For information on engaging a specialist for a NSCR, please contact your *Oracle account team*.

4 About Business Components, Fields, Joins, and Links

About Business Components, Fields, Joins, and Links

This chapter describes business components, business component fields, and joins. It includes the following topics:

- *About Business Components*
- *About Business Component Fields*
- *About Joins*
- *About Multi-Value Links*
- *About Links*

About Business Components

This topic describes business components. It includes the following information:

- *Overview of Business Components*
- *How a Business Component Gets Data from an External Database*
- *Business Components That Hold Temporary Data for a Task UI*
- *Class Property of a Business Component*
- *How a Business Component Sorts Records*
- *Guidelines for Creating a Business Component*

For more information, see the following topics:

- *Overview of the Business Object Layer*
- *Configuring a Business Component*
- *Properties of a Business Component*

Overview of Business Components

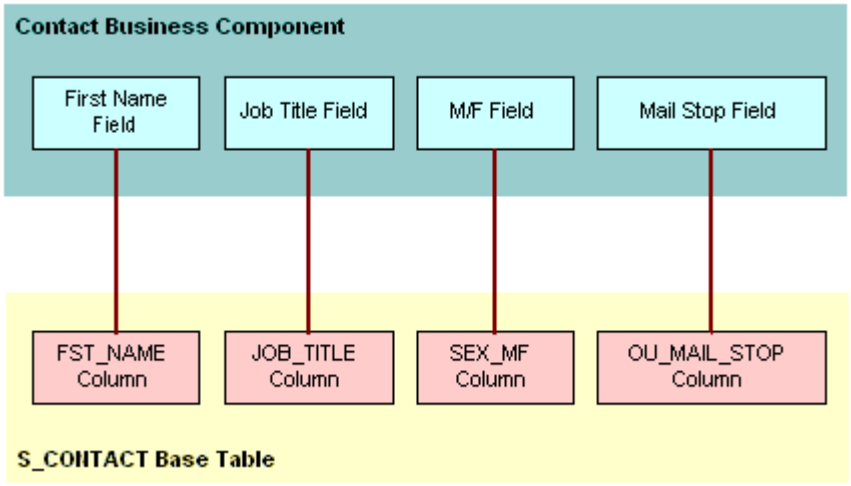
A *business component* provides the foundation for controlling how Siebel CRM chooses, inserts, and updates data in underlying tables. The information that Siebel CRM stores in a business component is typically specific to an area, such as a product, a contact, or an account. This information might or might not depend on other business components. A business component can exist in one or more business objects. It can include a default sort specification or search specification that allows you to display records in the Siebel client in a predetermined sort order and according to a set of selection criteria. Multiple users can instantiate copies of the same business component. Siebel CRM includes the data modifications that one user makes in all instances of the business component. For more information, see *Business Component* and *Options to Filter Data That Siebel CRM Displays in an Applet*.

How Business Component Fields Reference Base Table Columns

Siebel CRM gets the main data for a business component from a base table and one or more joined extension tables. For example, the Account business component references the S_PARTY table, but the S_ORG_EXT joined extension table stores most of the data that the Account business component gets.

Siebel CRM assigns a base table to each predefined business component. The *base table for* a party business component includes the most important columns that provide data to fields that reside in the business component. The Table property of the business component references the base table. A single business component field references a single base table column.

The following figure shows an example of how fields (First Name, Job Title, M/F, Mail Stop) in the Contact business component reference columns (FST_NAME, JOB_TITLE, SEX_MF, OU_MAIL_STOP) in the S_CONTACT base table. As shown in this figure, the First Name field references the FST_NAME column, the Job Title field references the JOB_TITLE column, and so on.



A business component does not always reference all columns that reside in the base table, but typically it does reference most of them. Implied fields in the business component represent system columns in the base table, such as ROW_ID, CREATED_BY, and LAST_UPD_BY. A system column does not require a field object definition in the business component.

For more information, see *How an Implicit Join Creates a Relationship Between a Base Table and a Business Component*.

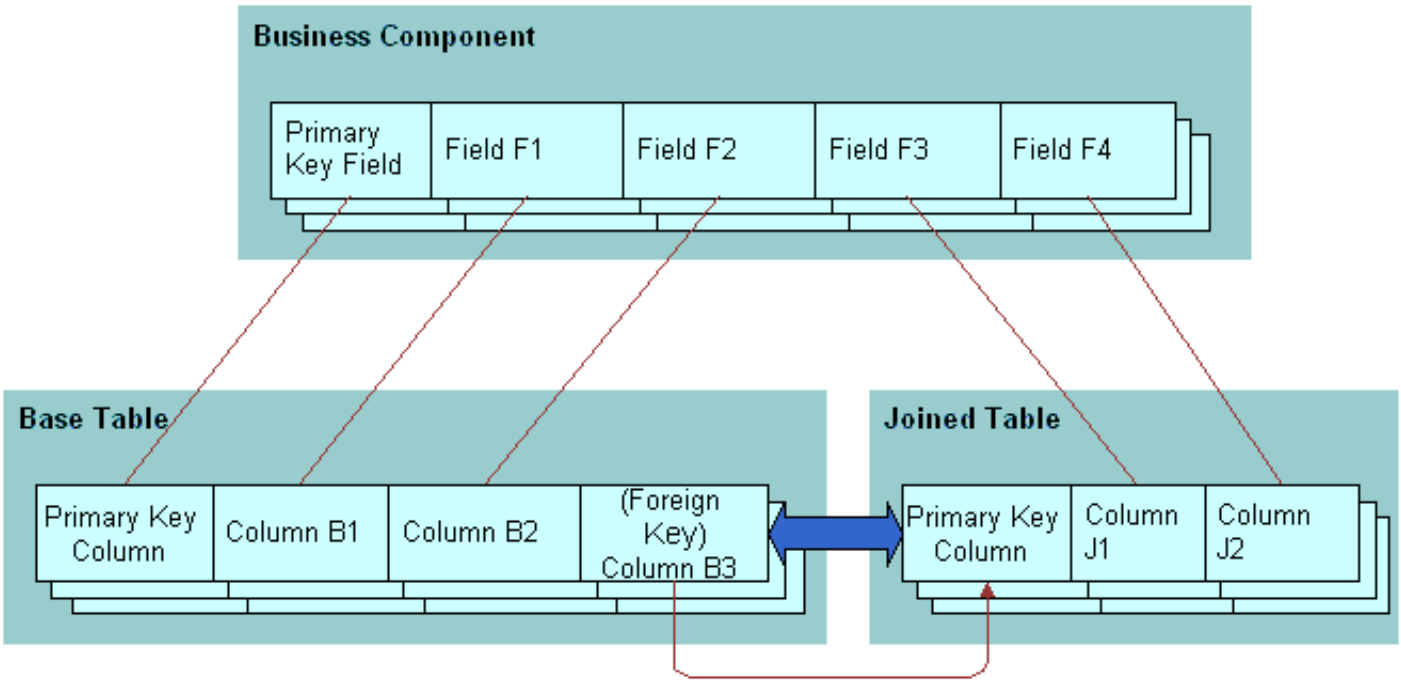
A Business Component Can Reference Data from a Joined Table

A business component can reference data from an extension table and a joined table. A *party business component* is a business component that references the S_PARTY table as the base table. The main data for a party business component comes from a joined table. A join defines the relationship that exists between the business component and the additional table. For more information, see *How the S_Party Table Controls Access*.

A *joined table* provides rows on a one-to-one basis to the business component. The foreign key relationship that exists between the joined table and the base table of the business component creates this basis. For every record in the business component that corresponds to a row in the base table, a corresponding row can exist in the joined table. Every record in the base table does not include a record in the joined table.

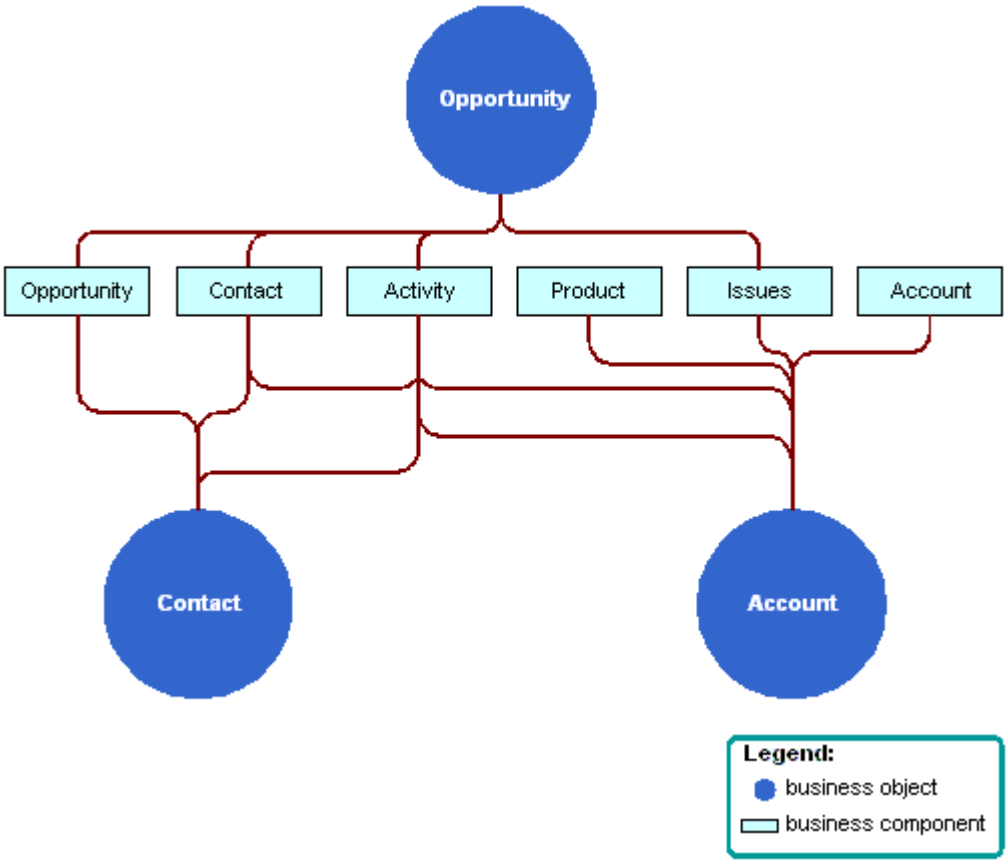
The following figure shows how fields in a business component (Primary Key, Field F1, Field F2, Field F3, Field F4) can reference columns in a base table (Primary Key Column, Column B1, Column B2, Column B3 Foreign Key) and a joined table (Primary Key Columns, Column J1, Column J2). In this figure:

- Primary Key references Primary Key Column, Field F1 references Column B1, and Field F2 references Column B2.
- Field F3 references Column J1 and Field F4 references Column J2.
- Column B3 (Foreign Key) in Base table references Primary Key Column in Joined table.



You Can Reuse a Business Component

The following figure shows how you can create a business component (such as Opportunity, Contact, Activity, Product, Account, and so on) once in terms of a logical collection of columns from one or more tables, and then use it in multiple business object contexts (such as Opportunity, Contact, Account). For more information, see *Business Objects and Business Components, Views, and Screens*.



How a Business Component Gets Data from an External Database

A *virtual business component* is a type of business component that references external data. This data is typically real-time information that Siebel CRM gets from an external database, but a virtual business component can reference any source that can supply data in reply to a structured query. You can use a virtual business component if you must get data from a location other than a table in the Siebel database.

You can use an *external business component* (EBC), which is a type of business component that uses ODBC and SQL to supply data. For more information, see *Integration Platform Technologies: Siebel Enterprise Application Integration* .

A virtual business component allows you to do the following:

- Represent external data as a virtual business component in Siebel CRM. The business component specifies the library that Siebel CRM uses to access the data.
- Use a business service to transfer data.

A virtual business component includes the following qualities:

- Supports single-value field.
- Supports field-level validation.
- Supports a predefined business component event model. For example, PreNewRecord, PreDelete, and so on.

- Supports insert, delete, query, and update operations.
- Can be a stand-alone business component or a child business component in a business object.
- Supports dynamic applet toggles. For more information, see *Options to Toggle Between Applets in a View*.
- Can work as the parent of a link in a one-to-many relationship with another business component:
    - Siebel CRM creates a Siebel row ID in a virtual business component in the same way that it creates this ID in a predefined business component.
    - Supports a many-to-many relationship to an external system that works similarly to a one-to-many relationship.
    - Does not support a many-to-many relationship to another virtual business component.
- Can be the basis for an applet.
- Can be accessed through an object interface.
- Can access all business component events for scripting.
- Cannot be docked.

For more information, see *Overview: Siebel Enterprise Application Integration* .

Business Components That Hold Temporary Data for a Task UI

A *transient business component* (TBC) is a type of business component that allows Siebel CRM to create data that the user can edit in a task-based user interface (Task UI). The data that Siebel CRM stores in a TBC is temporary. Siebel CRM uses this data during the life of the task UI typically to control the flow or logic, and then discards it when the task UI is complete. In some situations, Siebel CRM can store data in a TBC to long-term storage in the Siebel database. For more information, see *Siebel Business Process Framework: Task UI Guide* .

Creating a New Transient Business Component using the Web Tools Wizard

To create a new transient business component using the web tools wizard

1. Log into the Web Tools client.
2. Open an editable Development Workspace.
3. Click the **New Object Wizard** button. It has a magic wand icon.
4. Choose the *Transient BusComp* icon and click Start.
5. The first view shows three fields.
    a. **Project** (Required): This is the Project in which to put your new Transient Business Component.
    b. **Name** (Required): The unique name for this Transient Business Component in the Repository.
    c. **Multirecord TBC** : Check this box if the Transient Business Component will hold more than one record.
6. After configuring these fields click the Next button.
7. The next view is a simple summary. Click the Finish button to create the *Transient BusComp icon*.

Class Property of a Business Component

Siebel CRM contains a hierarchy of business component classes. The CSSBusComp class resides at the top (highest level) of the hierarchy. Siebel CRM gets all other specialized business component classes from this CSSBusComp class. You must set the Class property of any new business component that you create to CSSBusComp or CSSBCBase. The following functionality is common for a business component:

- Move through the set of records that Siebel CRM returns from the Siebel database
- Get or set field values in records
- Create and delete records
- Commit modifications
- Undo and redo
- Set a bookmark
- Do a search
- Do a sort

Caution About Using Specialized Classes

A *specialized business component class* is a type of class that Siebel CRM derives from a generalized business component class. It is recommended that you use a specialized business component class only if necessary. It is recommended that you do not use a specialized business component class with a typical business component. These specialized classes often use functionality that references other objects, such as fields, other business components, or other classes. You must not configure Siebel CRM in such a way that it modifies the values that these objects contain.

You must not modify the Class property of a predefined business component. If you copy a predefined business component, then you must not modify the Class property of the copy.

> **CAUTION:** Using a specialized business component class or applet class improperly might cause an unpredictable problem that can be difficult to fix. For example, Siebel CRM might add or delete a child record, or modify an associate record. A run-time error might occur. It is recommended that you configure the class property with extreme care and thoroughly test any modification you make.

Oracle only supports methods that *Siebel Object Interfaces Reference* describes for use in scripting. Modifying method logic before or after Siebel CRM calls this method can cause unpredictable behavior.

For more information, see *Siebel Developer's Reference* .

How a Business Component Sorts Records

A *sort specification* is a property of a business component that imposes a sort order on the records that Siebel CRM returns to an applet that references this business component. For example, the Sort Specification property of the predefined Account business component includes the following value:

```
Name(ASCENDING), Location
```

This value configures Siebel CRM to do the following:

- Sort account records according to the account name in ascending order.
- If the name is the same for multiple accounts, then it sorts the records that contain the same name according to the Account Location.

A sort specification includes the following qualities:

- If the Sort Specification property is empty, then Siebel CRM returns the records in the order that they occur in the table.
- If a predefined query exists, then it might override a sort specification that is defined on a business component.
- If Siebel CRM runs a sort on one of the following fields, then a sort specification can result in a negative effect on performance:
  - A field that references a join
  - A field that references a new extension column that is not indexed

  For more information, see *Siebel Performance Tuning Guide* .
- Siebel CRM displays empty records at the beginning of the result set.
- Siebel CRM cannot sort a calculated field.
- You can sort values in a static list or a pick applet differently than how the default sort for the underlying business component does the sort. For more information, see *Creating a Sort Specification for a Static List*.

For more information, see the following topics:

- *Determining How a Business Component Sorts Records*
- *Guidelines for Configuring How a Business Component Sorts Records*

How Siebel CRM Sorts a Multi-Value Field

If Siebel CRM references a multi-value field in a sort specification, then it does the following sort:

- Sorts on the initial value of the multi-value field. You must use this configuration only if the multi-value group references a primary foreign key.
- Does not sort the records that reside in the underlying multi-value group. To sort these records, you must create a sort specification in the child business component of the multi-value link.

For more information, see *About the Multi-Value Field*.

How Siebel CRM Sorts a Check Box Field

If a sort specification references a check box field, then Siebel CRM sorts the following values:

- Y
- N
- NULL

If a sort specification references a check box field, and if you define the sort in descending order, then Siebel CRM returns the records in the following order:

- NULL
- Y
- N

How the Visibility Mode Affects a Sort Specification

Siebel CRM forces the sort that the All visibility mode uses to be on the primary key. The sort in Manager mode occurs on a column in the denormalized reporting relationship table. Siebel CRM can still sort records after the initial query. For better performance, you must configure Siebel CRM to sort records after it filters for a small record set.

You can use the All Mode Sort business component user property to force Siebel CRM to use a custom sort specification or to ignore all sort specifications. For more information, see *Siebel Developer's Reference* .

Guidelines for Creating a Business Component

This topic describes guidelines for creating a business component. For more information, see the following topics:

- *Guidelines for Naming an Object*
- *Guidelines for Reusing a Predefined Business Component*
- *Guidelines for Reusing a Predefined Business Object*

Guidelines for Naming a Business Component

Do not use the parent entity in the name of a business component that represents child entities. For example, use ABC Subsegment instead of ABC Account Subsegment. Similarly, you can include only the name of the business component in an applet that references these child business components. For example, ABC Subsegment List Applet instead of ABC Account Subsegment List Applet.

An exception to this requirement occurs if you must use multiple variations of the same business component or applet. A multiple variation might be necessary if Siebel CRM must display an entity as the highest level applet and as a child applet on other views, and if these two applets are not the same applet. In this situation, you can place the name of the parent entity at the beginning of the name of the child applet. For example, the ABC Account Contact List Applet is a contact list that Siebel CRM displays as the child of an account. It uses the word Account to distinguish it from the predefined ABC Contact List Applet.

Guidelines for Creating a Business Component That References a Specialized Class

If you must create a new business component, then you must avoid copying a business component that references a specialized class unless you do the following:

- Create a true copy of the original business component that contains the same functionality.
- Apply only minimal modifications.

For example, you can create a Locked Service Requests business component that displays only the service requests that are locked. To do this, you use a business component user property:

- Copy the Service Request business component, and then reference the CSSBCServiceRequest specialized class from this new business component.
- Create the Lock Field business component user property.
- Create the conditions that Siebel CRM must use to lock a service request.
- Create a search specification for the business component that gets only the service requests that meet the conditions. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.

The underlying behavior of the new business component remains the same as the original business component. You must avoid copying a specialized business component to reproduce an isolated feature that is associated with that business component.

If you set the Class property of a business component to CSSBCServiceRequest, then you must add the Abstract field to this business component. If you do not add this field, and if an applet references a business component that is a child of the business component that you add, then Siebel CRM might disable the New button in this applet.

For more information, see *Caution About Using Specialized Classes*.

Guidelines for Configuring How a Business Component Sorts Records

If you define the Sort Specification property of a business component, then use the following guidelines:

- The fields that your sort specification references must be child objects of the business component.
- Use a comma to separate field names.
- Include (DESCENDING) or (DESC) after the field name to indicate that Siebel CRM sorts a field in the list in descending order. For example, Start Date (DESCENDING). If you do not add a sort order, then Siebel CRM uses ascending order.
- Do not enclose the field name in square brackets. For example, [Account Name]. Siebel CRM accepts brackets in a search specification but not in a sort specification. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.
- Do not exceed 255 characters in the sort specification.

Guidelines for Modifying a Predefined Business Component That Siebel CRM Does Not Use

If you modify a predefined business component that Siebel CRM does not use, then use the following guidelines:

- You must not delete, deactivate, rename, or modify any predefined object that Siebel CRM does not use. Do not delete these objects because other objects might reference them.
- You can delete any custom business component that you create that Siebel CRM does not use and that does not reference any other object that Siebel CRM uses, such as an applet.

About Business Component Fields

This topic describes business component fields. It includes the following information:

- *Overview of Business Component Fields*
- *How a Business Component Field Identifies the Type of Data*
- *How a Business Component Field Calculates a Value*
- *How a Business Component Field Sequences Records*
- *How Siebel CRM Defines Read-Only Behavior for a Business Component Field*
- *System Fields of a Business Component*
- *Guidelines for Defining the Name of a Business Component Field*

Overview of Business Component Fields

A business component field typically represents the following values:

- Information from a database column that the field gets from a table column. The column can reside in the base table, an extension table, or a joined table of the business component.

- A calculated value that Siebel CRM gets from the values in other fields but that it does not store in the Siebel database. For more information, see *How a Business Component Field Calculates a Value*.

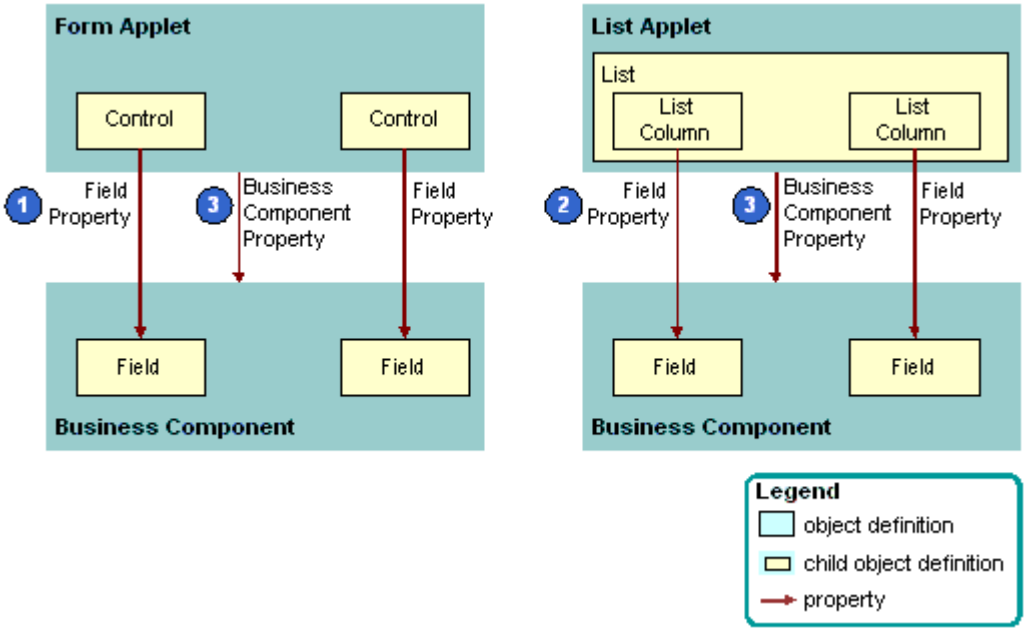
For more information, see *Business Component Field*.

How a Business Component Field Provides Data to the Control or List Column of an Applet

The following table describes several examples of how a business component field provides data to the control or list column of an applet.

Data in This Business Component Field	Provides Data to This Applet Control or List Column
Name field of the Opportunity business component.	Name control of the Opportunity Form Applet - Child applet.
Account field of the Opportunity business component.	Account control of the Opportunity Form Applet - Child applet.
Primary Revenue Amount field of the Opportunity business component.	Revenue control of the Opportunity Form Applet - Child applet.
Name field of the Account business component.	Name list column of the Account List Applet.
Main Phone Number field of the Account business component.	Main Phone Number list column of the Account List Applet.

The following figure describes how properties of a business component field and an applet reference each other when referenced from a form applet in comparison to a list applet. If a business component field is not a calculated field, then the Join and Column properties together define the table and column that Siebel CRM uses to get the data for the field. For more information, see *How an Implicit Join Creates a Relationship Between a Base Table and a Business Component*.



Explanation of Callouts

The following properties of a business component field and an applet reference each other:

1. **Field property of a control.** The Field property of a control references a business component field.
2. **Field property of a list column.** The Field property of a list column references a business component field.
3. **Business Component property.** The Business Component property of an applet references the business component.

How a Business Component Field Identifies the Type of Data

The Type property of a business component field identifies the type of data that Siebel CRM gets from and sends to the Siebel database. Siebel CRM does not map these data types to the physical data types that it defines for the database. The data type of the field is typically more specific than the data type of the underlying column. For example, the DTYPE_NUMBER (decimal) and DTYPE_INTEGER field data types each reference the Number physical data type in the column. For more information, see *Type Property of a Business Component Field*.

Just as the data type of the underlying table column restricts the set of field data types that work correctly, the data type of a business component field restricts the set of format options that Siebel CRM can use in the control or list column that reference the field.

It is recommended that you do not map a field to a table column that is not the same type as the field. For example, do not map a DTYPE_NUMBER business component field to a Varchar table column.

Siebel CRM gets most default values for formats from the Microsoft Windows Control Panel. Overriding the default format in the Siebel repository but might cause confusion. For example, overriding a number format to display more or fewer decimal places is useful, but overriding a date format to DD/MM/YYYY is confusing to a user who set the date format to MM/DD/YYYY in the Control Panel. For more information, see *How Siebel CRM Handles Certain Date Formats*.

The Type property of a multi-value field is empty because Siebel CRM defines the data type of the field in the child business component that enters data in the multi-value field. For more information, see *About the Multi-Value Field*.

How a Business Component Field Calculates a Value

A *calculated field* is a type of business component field that gets values from other fields that reside in the same business component or from the parent business component in an active link where the current business component is the child business component. The Calculated property of a calculated field contains a check mark and the Calculated Value property contains a value that is not empty.

The Calculated Value property contains an expression that Siebel CRM builds from field names, predefined functions, and string, numeric, and logical operators. For example, the Calculated Value property of the Full Name field that resides in the Contact business component includes the following value:

```
IIf (Language () = "JPN", [Last Name] + ' ' + [First Name],
[First Name] + ' ' + [Last Name])
```

This expression does the following:

- If the active client language is Japanese, then create the Full Name from the Last Name, an empty space, and the First Name.
- If the active client language is not Japanese, then create the Full Name from the First Name, an empty space, and the Last Name.

If you create a calculated field, then consider the following:

- If Siebel CRM modifies the calculated value of a field, then it does not refresh this calculated field, by default. It only refreshes a calculated field after it commits the record. To refresh the field immediately after Siebel CRM modifies the value in this field, you can make sure the Immediate Post Changes property of this field contains a check mark.
- A calculated field cannot reference itself in the Calculated Value property. For example, you cannot use Last Name in a calculation expression for the Last Name field.
- If the Cache Data property of the business component contains a check mark, then Siebel CRM does not support a query on a calculated field in this business component.
- You cannot use a script on a calculated field.

For more information, see *Siebel Developer's Reference* .

How a Business Component Field Sequences Records

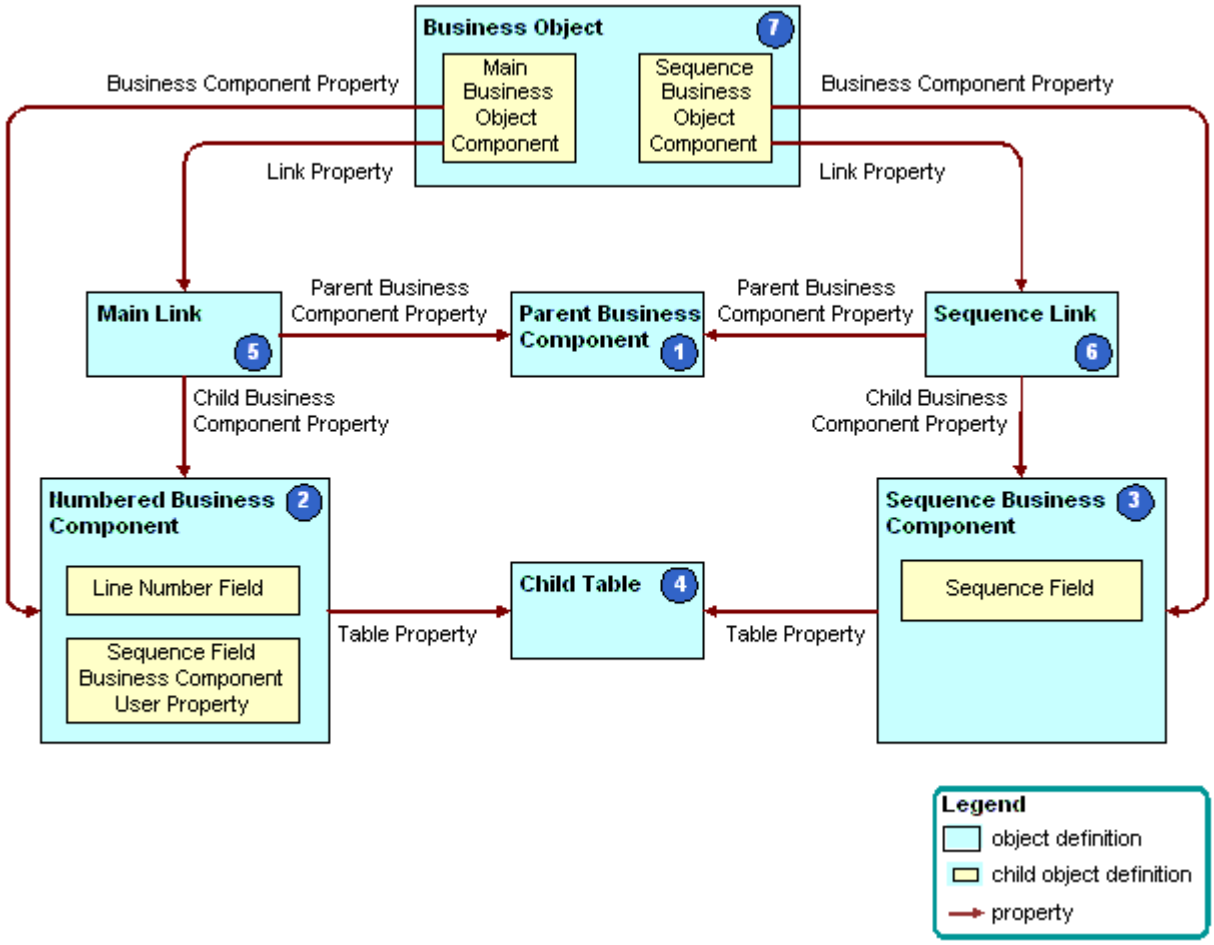
A situation might occur where you must create a field that provides sequential numbering for the parent business component. For example, you might need to number line items in an order or products in an opportunity. The sequence field behaves as follows:

- It is editable and can be set to any number.
- When Siebel CRM creates a new record, the initial sequence number is the maximum sequence number of the existing child records, incremented by one.
- Siebel CRM does not renumber records to resolve the gap that results when it deletes a record. The user must do this work manually.

For more information, see *Determining How a Business Component Sequences Records*.

How Siebel CRM Creates a Sequence Field

The following figure describes how Siebel CRM creates a sequence field.



Explanation of Callouts

As shown in this figure, Siebel CRM uses the following objects to create a sequence field:

1. **Parent business component.** The business component that contains the parent records in the parent-child relationship where the child records are numbered. For example, the Opportunity business component is the parent in the parent-child relationship with Opportunity Product.
2. **Numbered business component.** The business component that contains the child records in the parent-child relationship. For example, the Opportunity Product business component is the detail in the parent-child relationship with Opportunity. The numbered business component includes the following child object definitions:
    - **Sequence value field.** A DTYPE_NUMBER field that contains the resulting sequence value. Line Number and Order are examples of a sequence value field.
    - **Business component user property.** The Sequence Field business component user property must be present, with the Value property set to the name of the sequence value field.

3. **Sequence business component.** This business component is named *xx.yy.*

   where:

   - *xx* is the name of the numbered business component.
   - *yy* is the name of the sequence value field.

   It references the CSSSequence specialized class, and it contains the following fields:

   - **Sequence field.** This field is named Sequence and is a DTYPE_NUMBER field.
   - **Foreign key field.** A foreign key field that references a foreign key column that resides in the detail table. The foreign key column references the primary key of the base table that the parent business component references. You can use it to create the link that resides between the parent business component and the sequence business component.

   For more information, see *Caution About Using Specialized Classes*.
4. **Detail table.** The base table that the numbered business component and the sequence business component references.
5. **Main link.** The parent-child relationship that exists between the parent business component and the numbered business component. This link is typically predefined, such as Opportunity or Opportunity Product.
6. **Sequence link.** The parent-child relationship that exists between the parent business component and the sequence business component. You must typically add this link, except when a predefined Siebel application includes the sequence configuration. Opportunity and Opportunity Product.Line Number (Sequence) are examples of a link to a sequence business component.
7. **Business Object.** Includes the main link and the sequence link.

You can view an example configuration of a sequence field in a predefined Siebel application in Siebel Tools. For example, examine the Sales Assessment Attribute numbered business component, and the Sales Assessment Attribute Value.Order (Sequence) sequence business component.

How Siebel CRM Defines Read-Only Behavior for a Business Component Field

Field	Description
Column	The name of the table's column. Default table is the business component table. This property is required unless it is a calculated field.
Name	User-defined name for the field. It must be unique within the business component. This property is required.
No Copy.	If TRUE, the field's value is not copied into the newly created record during a Copy Record operation. The No Copy property of a Multi Value Link will override the No Copy property of a Field in the child Business Component.
Read Only	If TRUE, the field value cannot be changed by the user.

Field	Description
Required	If TRUE, a value must be entered before the record can be written to the database.
Text Length	Siebel Tools gets the text length from the database and, for columns with a physical type of varchar, sets the Text Length property for a business component field to this value. If the physical type is character(1), the Text Length is set to 1 and the Type is set to DTYPE_BOOL. For number, date, and datetime fields, Siebel Tools does not put a value in Text Length. Exceptions are fields mapped to foreign key columns (these columns have names that end in ID and have a physical type of varchar and length of 15). These fields get a Siebel Type of DTYPE_ID and a Text Length of 15. If you edit the value in the Text Length property, it is ignored unless the value in a picklist is longer than what is specified on the business component field. In this case you get an error.
Type	The field data type. For information about the data types for the field object type, see *How a Business Component Field Identifies the Type of Data*.

This topic describes how Siebel CRM defines read-only behavior for a business component. It includes the following information:

- *How the BC Read Only Field User Property Works*
- *How the Field Read Only Field User Property Works*
- *How the Parent Read Only Field User Property Works*
- *How the Parent Read Only Field: Business Component Name User Property Works*
- *Guidelines for Using a Business Component User Property With the Admin Mode Property*

You can turn on or turn off the read-only status of a business component or business component field while Siebel CRM is running, depending on the value that the field of the current record contains. For more information, see *Defining Read-Only Behavior for a Business Component*.

The following table describes the business component user properties that you can use to define read-only behavior. For more information about user properties, see *Siebel Developer's Reference* .

Business Component User Prop	Description
BC Read Only Field	Defines a TRUE or FALSE field in the record. If TRUE, then the current record is read-only. For more information, see *How the BC Read Only Field User Property Works*.
Field Read Only Field: fieldname	Defines a TRUE or FALSE test field and a target field that reside in the same business component. If TRUE, then the target field is read-only. The format for FieldName works if FieldName is not a join field. If FieldName is a join field to another table, then this format does not update the field that uses this format in the Pre Default Value property of the field. For more information, see *How the Field Read Only Field User Property Works*.
Parent Read Only Field	Defines a TRUE or FALSE test on a field in the parent business component. If TRUE, then the target business component is read-only. For more information, see *How the Parent Read Only Field User Property Works*.

Business Component User Prop	Description
Parent Read Only Field: business component name	Defines a TRUE or FALSE test on a field that resides in the parent business component. This configuration is similar to the Parent Read Only Field business component user property, except the name of the user property rather than the value specifies the parent business component. For more information, see *How the Parent Read Only Field: Business Component Name User Property Works*.

How the BC Read Only Field User Property Works

The BC Read Only Field user property specifies a Boolean field that, if TRUE, modifies all fields that reside in the current record to read-only. It prevents the user from updating or deleting the record but it does not prevent the user from adding a new record to the business component.

The BC Read Only Field user property includes the following properties:

- **Name.** Contains the following value:

  ```
  BC Read Only Field
  ```

- **Value.** Contains the name of a field that resides in the parent business component of this business component user property. This field must be contain a TRUE or FALSE value.

Example of Using the BC Read Only Field User Property

Assume Siebel CRM must prevent the user from updating an inactive account. The Inactive Account field in an account record is a TRUE or FALSE field that, if TRUE, indicates that the account is inactive. To configure dynamic read-only behavior for the Account business component, you can add a business component user property to the Account business component. This example business component user property contains the following properties:

- **Name.** BC Read Only Field.
- **Value.** Inactive Account.

How the Field Read Only Field User Property Works

The Field Read Only Field user property is similar to the BC Read Only Field user property because it tests the field that you define in the Value property, and it enforces a read-only restriction if the value of the test field for the current record is TRUE. Unlike the BC Read Only Field user property, the Field Read Only Field user property restricts only one field in the business component record rather than restricting all fields in the entire business component record.

The Field Read Only Field user property includes the following properties:

- **Name.** Contains an expression in the following format:

  ```
  Field Read Only Field: fieldname
  ```

  where:

  - fieldname is the name of the field where Siebel CRM applies a read-only restriction.

  For example:

```
Field Read Only Field: Account Status
```

You must include only a single space between the colon and the field name.

- **Value.** Contains the name of the test field. This is a TRUE or FALSE field that resides in the parent business component of the user property.

You must create one Field Read Only Field user property for each field that you must make conditionally read-only.

How the Parent Read Only Field User Property Works

The Parent Read Only Field user property, like the BC Read Only Field user property, places a read-only restriction on an entire business component rather than on a single field. This restriction occurs if a TRUE or FALSE test field includes a TRUE value. Unlike the BC Read Only Field and Field Read Only Field user properties, the Parent Read Only Field user property places a restriction on a child business component of the business component that contains the test field. In the other user properties, Siebel CRM places the read-only restriction on the business component that contains the test field, or on another field in the same business component.

You can use the Parent Read Only Field user property to do the following:

- Restrict the child records that Siebel CRM includes in a multi-value group.
- Restrict the child records that Siebel CRM includes in a master-detail view. You must make sure that Siebel CRM does not use the restricted business component in the context of some other business object.

The Parent Read Only Field user property includes the following properties:

- **Name.** Contains Parent Read Only Field.
- **Value.** Contains an expression in the following format:

```
business component name.field name
```

where:

  - business component name is the name of the business component where the test field resides. For example, Account.Inactive Account.
  - field name is the name of the test field. This field is the TRUE or FALSE field that Siebel CRM evaluates.

You add the user property as a child of the business component that Siebel CRM restricts according to a condition. The business component that contains the test field must be a parent of the restricted business component through a link or through a series of link relationships.

If you use the Parent Read Only Field user property, then the value of the Link Specification property of the test field must be TRUE. If it is not TRUE, then the dynamic read-only functionality does not work. If Siebel CRM displays the child record in the multi-value field in the parent business component, then the Link Specification property of the field does not have to equal TRUE.

Example of Using the Parent Read Only Field User Property

Assume that if the account record includes a Type of Competitor, then Siebel CRM must not update the Account Address Mvg Applet. To do configure this requirement, you add the same calculated field that the *How the BC Read Only*

Field User Property Works topic describes. You then add a user property to the Business Address business component with the following values:

- **Name.** Parent Read Only Field.
- **Value.** Account.Competitor Calc.

This configuration causes the Account Address Mvg Applet to be read-only if the account record is for a competitor.

How the Parent Read Only Field: Business Component Name User Property Works

The Parent Read Only Field: *business component name* user property allows a child business component to do a TRUE or FALSE test on multiple parent business components. The behavior of the Parent Read Only Field: *business component name* user property is similar to the behavior of the Parent Read Only Field user property. The name rather than the value specifies the parent business component. If the calculated value of the field is TRUE or Y, then the child business component is read-only. For more information, see *Siebel Developer's Reference* .

Guidelines for Using a Business Component User Property With the Admin Mode Property

Do not use a business component user property with an applet that resides in a view if the Admin Mode Flag property of this view contains a check mark. If this Admin Mode Flag contains a check mark, then Siebel CRM does the following:

- Turns off all insert and update restrictions for the business components that the view uses, including the restrictions that the business component user property defines.
- Ignores the Sales Rep and Personal visibility modes of the business component.
- Makes records that do not include a primary team member visible.
- Does not override pop-up visibility.

You can include a check mark in the Admin Mode Flag property only if the view is part of a screen that contains only administration views. You can create a list view where the Admin Mode Flag property contains a check mark if this list view drills down to a detail view that Siebel CRM does not mark as an administration view. This configuration allows you to share a detail view with a list view that is not an administration view.

> **CAUTION:** All views and drilldowns in a screen that is granted Admin Mode will behave according to the Admin Mode due to their subordinate relationship to the screen. If a view is a child of a screen that is in Admin Mode, and if the Admin Mode Flag property of the view does not contain a check mark, then Siebel CRM still displays the view in Admin Mode.

System Fields of a Business Component

A *system field* is a business component field that represents the data that Siebel CRM gets from a system column. All business components in Siebel CRM include system fields. You are not required to do any special configuration to

display or manipulate a system field. You do not need to define it as a business component field. For example, you can reference a system field in the Field property of a control, list column, or in another object.

You must not modify a system field. For example, by renaming it. Siebel CRM does not support modifying a system field.

For more information, see *System Columns of a Siebel Table* and *Displaying a System Field in an Applet*.

Relationship Between a System Field and a System Column

The Id field that represents the ROW_ID column in a business component is an implicit field, and Siebel Tools does not display it in the Object Explorer as a child field of a business component. Every business component includes an Id field that represents the ROW_ID column of the base table of the business component, as defined in the Table property of the business component. The Siebel schema references the Id field in various properties throughout Siebel CRM. For example, in the Source Field property of a link where an empty value indicates the Id field.

You must not explicitly create a system field for a business component. If you create a business component field that references a system column, then Siebel CRM attempts to write a value to the column twice in the insert statement, which causes a duplicate column SQL error.

The following table describes the relationship that exists between a system field and a system column. Each field is predefined. You do not explicitly define it. You can configure Siebel CRM to reference a system field in the Field property of a control, list column, or other object, even though Siebel Tools does not display the field in the Business Components list.

System Field Name	System Column Name	Description
Id (or empty)	ROW_ID	Stores the primary key for the table.
Created	CREATED	Stores the date and time of when Siebel CRM created the row.
Created By	CREATED_BY	Stores the ROW_ID of the row from the S_USER table that references the person who created the record.
Updated	LAST_UPD	Stores the date of the most recent update that Siebel CRM performed on the row.
Updated By	LAST_UPD_BY	Stores the ROW_ID of the row from the S_USER table that references the person who last updated the record. In some situations, Siebel CRM updates this field even though the user does not actively update the record. For example, if Siebel CRM configures a multi-value link with a primary join. For more information, see *Defining the Primary ID Field of a Multi-Value Link*.
(varies)	DB_LAST_UPD	Stores the date of the most recent update that Siebel CRM performed on the row in the Siebel database. The system field name varies.

Guidelines for Defining the Name of a Business Component Field

If you define the name of a business component field, then use the following guidelines:

- Do not use parentheses in a field name. If you do a query on a field name that contains parentheses, then you might receive an SQL error because SQL expects a valid SQL expression in the parentheses.
- Apply the following requirements if each field is the only such field in the business component:
  - If you name a currency code, then name it Currency Code.
  - If you name a currency date, then name it Exchange Date.
- If multiple instances of a similar field exist, then it is recommended that you prefix each field with the name of the corresponding Amount column. For example, Revenue Currency Code for revenue or Budget Currency Code for budgets. The reason for this configuration is that other fields reference these fields when you define the Currency Code Field property and the Exchange Date field. This configuration makes sure Siebel CRM can understand the reference.
- For a link, you must name the URL field URL and you must set the Class property of the business component to CSSBCBase.

For more information, see the following topics:

- *Guidelines for Naming an Object*
- *Reusing Predefined Objects*
- *Guidelines for Reusing a Predefined Object*

About Joins

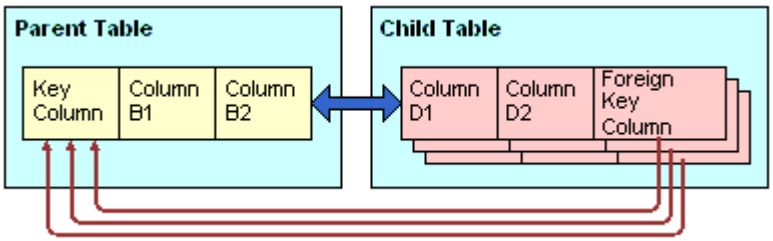
This topic describes joins. It includes the following information:

- *How Siebel CRM Creates a Join*
- *Guidelines for Creating a Join*

A join creates a relationship between a business component and a table that is not the base table of the business component. For more information, see *Join*.

To observe how a join works, in the Siebel client, you can navigate to the Service Request screen, and then examine the Service Request List Applet. This applet includes the Account field. A join brings the Account field to the Service Request business component, and then the Service Request List Applet displays the data in the Siebel client.

The following figure describes how a foreign key column that resides in the detail table defines the parent-child relationship. Multiple rows in the detail table include the same foreign key value that references back to the same row in the parent table. After you create a join, you can define more fields in the business component that reference columns in the joined table.

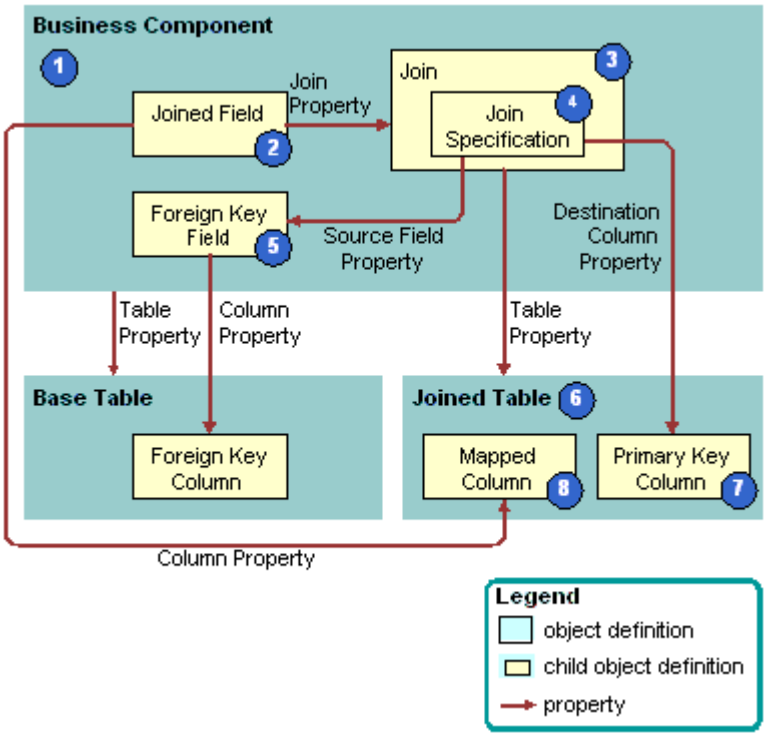


You can use a joined field as the Source Field on the join specification. For example, if you must join grandparent data through the parent ID field on the parent business component.

For more information about implicit joins, see *How an Extension Table Stores Custom Data*.

How Siebel CRM Creates a Join

The following figure describes how How Siebel CRM creates a a join.



Explanation of Callouts

As shown in this figure, Siebel CRM uses the following objects to create a join:

1. **Business component.** The business component is the parent of the join. Because of the join, a field in the business component that Siebel CRM joins can represent a column from the joined table.
2. **Joined field.** A *joined field* is a business component field that represents a column from a table other than the base table of the business component. For more information, see *How Siebel CRM Uses a Joined Field*.
3. **Join.** A join is a child of the business component. The join uniquely identifies a join relationship for the parent business component and provides the name of the joined table. The Table property of the join identifies the joined table. The join includes a child *join constraint*, which is an object that contains a constant value search

specification that Siebel CRM applies to a column during a join. Siebel CRM uses it with an outer join. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.

If Siebel CRM must get all records in the business component even if the joined fields are empty, then you must make sure the Outer Join Flag property contains a check mark.

4. **Join Specification.** The join specification is a child of the join. It identifies the foreign key field in the business component and the primary key column in the joined table. For more information, see *How Siebel CRM Uses the Join Specification*.
5. **Foreign key field and foreign key column.** The Source Field property of the join specification identifies the foreign key field. It represents a foreign key column that resides in the base table and it references the rows in the table that Siebel CRM uses in a join. For example, the foreign key field to the join on accounts data in the Contact business component is the Account Id field. This Account Id field references the PR_DEPT_OU_ID column in the base table.
6. **Joined table.** The joined table is the parent table in the parent-child relationship. It provides columns to the business component through the join. The Table property of the join identifies the joined table.
7. **Primary key column.** The Destination Column property of the join specification identifies the primary key column in the joined table. Each table in Siebel CRM includes a ROW_ID column that uniquely identifies the rows that the table contains. ROW_ID is the destination in most joins.
8. **Mapped column.** Columns in the joined table are available for use in fields in the business component.

How Siebel CRM Uses a Joined Field

A joined field gets values through a join. Siebel CRM includes the name of the join in the Join property of the field. The Join property and Column property together identify the column and how to access it. If you create a joined field in a business component, then you can modify the Type property from the default DTYPE_TEXT to a more appropriate type. For example, if you join a table column that contains phone numbers, then you can modify the Type field to DTYPE_PHONE.

How Siebel CRM Uses the Join Specification

The Source Field property of the join specification identifies the foreign key field that resides in the business component. If left empty, then the Source Field is the Id field that indicates a one-to-one relationship between the business component and the joined table. Siebel CRM sometimes defines a system field as the foreign key field in the Source Field property. The Created By and Updated By fields are examples of system fields. For more information, see *System Fields of a Business Component*.

The Destination Column property identifies the primary key column in the joined table. If the join occurs on a column other than ROW_ID, then the Destination Column property must not be empty. An empty value in the Destination Column property indicates that the destination column is ROW_ID that is typically the primary key in a table.

In rare situations, multiple join specifications can exist in a single join. For example, the Sub Campaign business component includes a join to the S_LANG table with two join specifications. In this situation, the source fields in the join specifications must reference the same table. For more information, see *Join Specification*.

How Siebel CRM Filters Duplicate Records From a Join In an Applet

A join between two business components can return one or more records. For example, if the joined table is an intersection table. In the applet, Siebel CRM displays only the first record in the result set. An applet that references a business component cannot display duplicate records from the base table of the business component.

For example, a many-to-many relationship exists between the Service Request and Organization business components. The link between these business components is Service Request/Organization. This link uses the S_SRV_REQ_BU table as the intersection table. In the Service Request business component, you can add a join to the S_SRV_REQ_BU table and a related joined field. If the user queries the business component to get a service request, and then the SELECT statement gets all the organizations that Siebel CRM associates with the service request. Siebel CRM displays only one

service request record in the Siebel client. To view all the organizations that Siebel CRM associates with the service request, the user can open the multi-value group applet that references the Organization business component.

For more information, see *Guidelines for Naming an Object*.

Guidelines for Creating a Join

If you create a join, then use the following guidelines:

- Use a join only if it gets no records or only one record. For example, use a join to get the primary account for an opportunity.
- Create a join only if the business component does not already include a join to a table that includes the data that your configuration requires, and only if a foreign key value exists between the base table of the business component and the joined table.
- Create a join only if Siebel CRM stores the foreign key value in a field that it does not already define as a source in a predefined join.
- If you use the Alias property to create an alias for each join, then a business component can include more than one join that references the same destination table. For example, the Action business component includes two joins that reference the S_CONTACT table:
  - The Owner join gets the person who created the activity.
  - The Primary Contact join gets the contact that is associated with the activity.
- Make sure the Alias property of the join is unique even though the destination table is the same. Do not use the table name as the Alias name, even though this is common in the predefined Siebel repository. An implicit join uses the table name as the Alias to make sure that Siebel CRM does not use the name of the explicit join. To make sure that no conflict exists, you must always create a unique alias name for the join.

Guidelines for Creating Joins With a Party Table

If you create a join that does or does not involve a party table, then use the following guidelines:

- If your join brings party data to a nonparty business component, then create a new join where the join specification references PAR_ROW_ID.
- If your join brings party data to a party business component, then use the appropriate explicit join.
- If you map fields in a party business component, then use the implicit join for the extension table.
- If a join references a table that is a party table, then you must display the foreign key value as the source field. Unlike a join to a table that is not a party table, the destination column must reference the PAR_ROW_ID column that resides in the joined table.
- If a join references a table that is not a party table, then Siebel CRM can update only the column that the field in the parent business component that contains the foreign key value references. You must define the following objects:
  - The joined table.
  - The join specification. The source field property must reference the parent business component that stores the foreign key value. The destination column property must reference the child table, which is typically ROW_ID.

For more information, see *How the S_Party Table Controls Access*.

About Multi-Value Links

This section describes multi-value links. It includes the following information:

- *How Siebel CRM Creates a Direct Multi-Value Link*
- *How Siebel CRM Creates an Indirect Multi-Value Link*

A multi-value link (MVL) is a child object of a business component. It describes the link that provides field values from the child business component that the multi-value group (MVG) applet references. The MVL does the following:

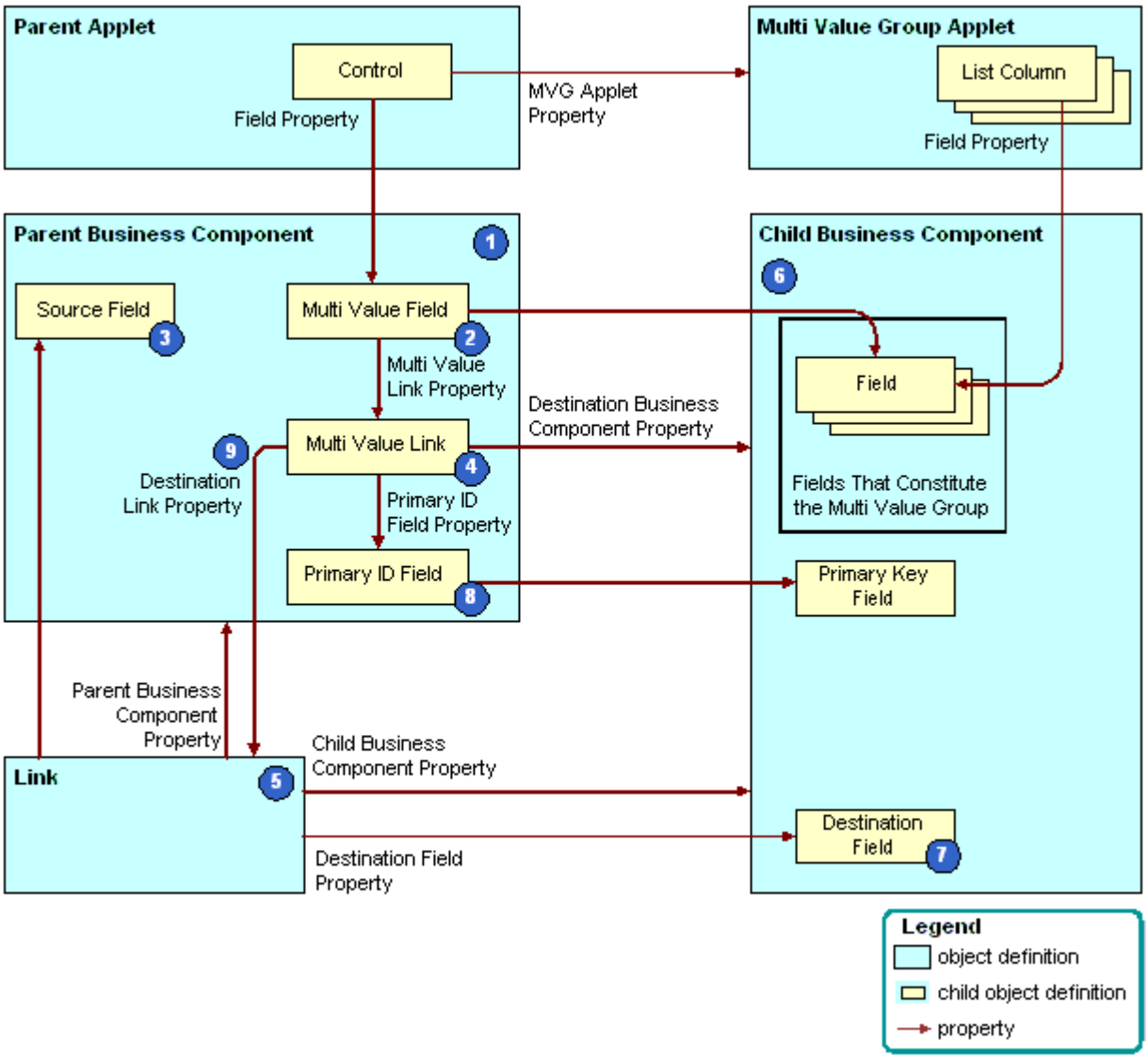
- Defines the parent-child relationship that Siebel CRM uses to display fields from the child business component directly in the parent business component.
- Provides a field that resides in the parent business component with access to the values that the primary record of a MVG contains.

A parent-child relationship exists between the business component that the originating applet references and the business component that the MVG applet references. A link defines this parent-child relationship. The relationship between the two business components is one-to-many in the context of the MVL and MVG. A many-to-many relationship can exist – for example, between opportunities and positions. In the context of the MVG, Siebel CRM presents only one parent-child relationship.

For more information, see *Multi-Value Group, Multi-Value Link, and Multi-Value Group Applet*.

How Siebel CRM Creates a Direct Multi-Value Link

The following figure describes how Siebel CRM creates a direct multi-value link.



Explanation of Callouts

As shown in this figure, Siebel CRM uses the following objects to create a direct multi-value link:

1. **Parent business component.** The parent in the parent-child relationship that Siebel CRM defines in the link. Siebel CRM displays fields from this business component in the applet that the user opens in the multi-value group applet. In *Viewing an Example of a Multi-Value Group Applet*, the Account business component is the parent that Siebel CRM uses to open the Account Address Mvg Applet.
2. **Multi-value fields.** For more information, see *About the Multi-Value Field*.
3. **Source field.** Defines the primary key in the parent business component that uniquely identifies records in the business component. It typically represents the ROW_ID column from the base table of the business component, and it fulfills the role of the primary key field. If the Source Field property is empty, then the source field references the ROW_ID column.
4. **Multi-value link.** Defines the relationship that exists between the link and fields in the parent business component.

5. **Link.** Defines a parent-child relationship between the parent business component and the child business component. You can use the link in multiple ways, such as with a master-detail view or with another multi-value link. In *Viewing an Example of a Multi-Value Group Applet*, the name of the link is Account/Business Address.
6. **Child business component.** Supplies the child records in the parent-child relationship. It contains the records that constitute a multi-value group. In *Viewing an Example of a Multi-Value Group Applet*, this is the Business Address business component.
7. **Destination field.** Contains row ID values that reference back to records in the parent business component and uniquely identify the parent for each child business component record. The link identifies the foreign key field in the Destination Field property. In *Viewing an Example of a Multi-Value Group Applet*, the foreign key field is Account Id. For more information, see *About the Destination Field*.
   No foreign key field is defined in a link that references an intersection table.
8. **Primary ID Field.** Identifies the foreign key field in the parent business component. For more information, see *About the Primary ID Field*.
9. **Destination Link.** Identifies the link that defines the parent-child relationship between the parent business component and the child business component.

About the Destination Field

The *Destination Field* is a foreign key that references back to the parent business component. It identifies the field in the child business component that contains the data that constitutes the multi-value group (MVG). This field identifies the master record for each detail record.

A foreign key field represents a foreign key column from the base table of the child business component. Account ID and Opportunity ID are typical foreign key fields.

The Destination Field property must contain the name of a field in the base table of the business component that doesn't reference a join, and this field must be updated. The exception to this requirement occurs if a link references an intersection table, which is indicated if the Inter Table, Inter Parent Column, and Inter Child Column properties aren't empty. In this situation, the Source Field and Destination Field properties aren't defined.

If you add a record to the child business component in a link, then Siebel CRM initializes a link destination field. You can create a source field for a many-to-many link. The destination always defaults to `Id`, even if you create another value.

About the Primary ID Field

The *Primary ID Field* is a field in the parent business component that includes these items:

- Contains the row ID value of the primary record for each record of the multi-value group (MVG) in the child business component.
- Identifies the field in the child business component that designates the record that is the primary.
- Identifies the foreign key field in the parent business component.
- Is identified in the Primary Id Field property of the multi-value link.

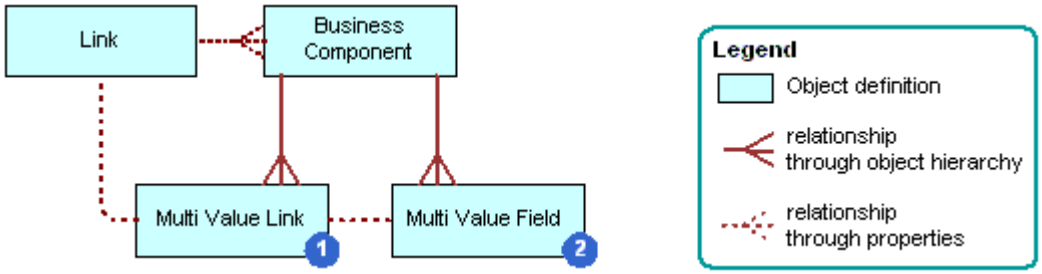
For more information, see *Defining the Primary ID Field of a Multi-Value Link* and *Determining Whether You Can Reuse a Predefined Business Component Field*.

About the Multi-Value Field

The *Multi-Value Field* is a business component field that contains the name of the multi-value link (MVL) in the Multi Value Link property and a check mark in the Multi Value property. All other business component fields are single-value fields. A multi-value field contains data from a record in the child business component because of the MVL.

To search the parent applet for all parent records that contain a child record that holds a field value, then you must use a multi-value field.

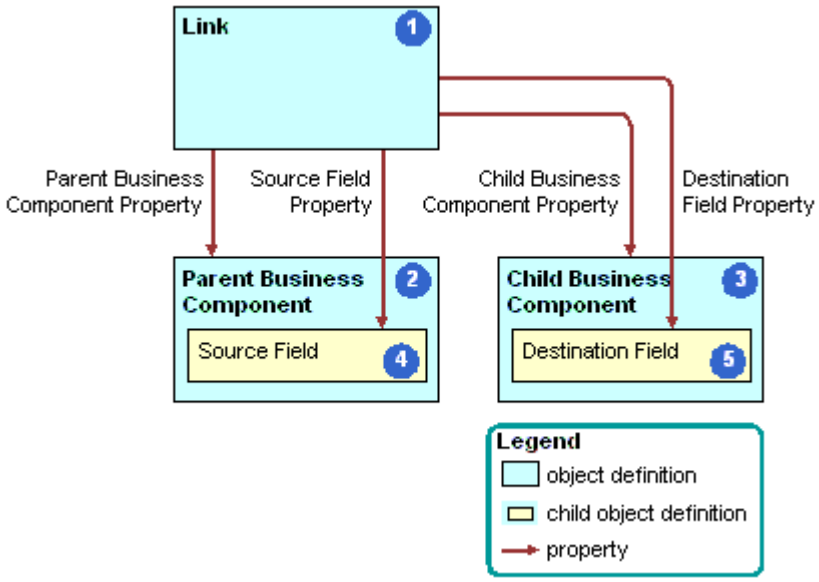
The following table describes some properties, as illustrated in the following figure, of the multi-value field.



Number (as shown in preceding figure)	Property	Description
1	Multi Value Link	Identifies the MVL that provides values through the link from the child business component. A multi-value field in the parent business component contains data that Siebel CRM gets from the current record in the child business component through the MVL and link. The Column property of a multi-value field is empty because Siebel CRM gets values from the current record in the child business component rather than from the base table of the parent business component.
2	Field	Identifies the field in the parent business component that provides values for the field in the child business component. Siebel CRM provides these values through the MVL object and the link object.

As shown in the following figure, a link (1) contains the following properties:

- (2) Parent Business Component – (4) Source Field.
- (3) Child Business Component – (5) Destination Field.



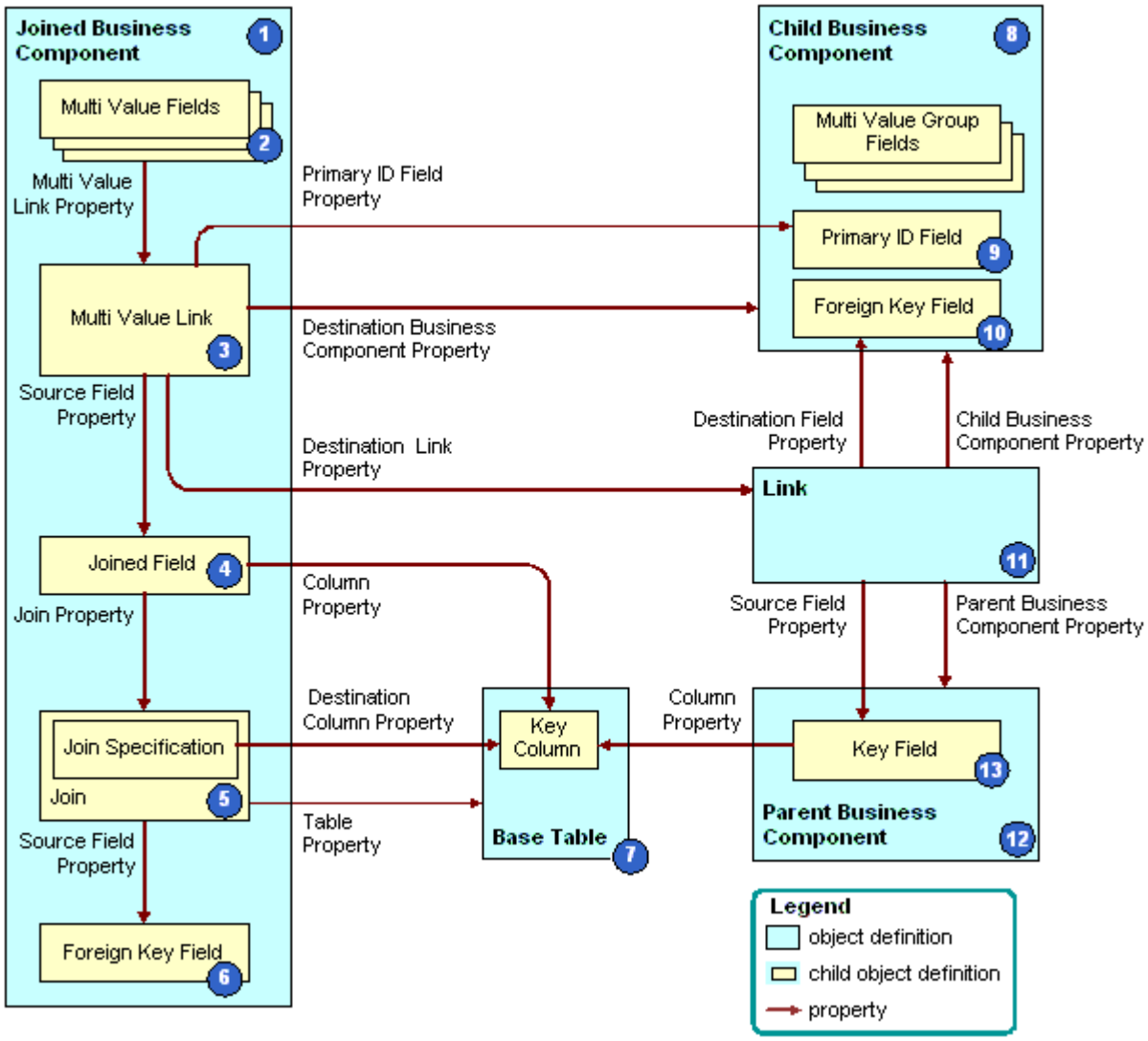
In *Viewing an Example of a Multi-Value Group Applet*, the Street Address multi-value field in the Account parent business component contains data from the primary record of the multi-value field in the child business component.

For more information, see *How Siebel CRM Sorts a Multi-Value Field* and *Activating a Multi-Value Field*.

How Siebel CRM Creates an Indirect Multi-Value Link

An *indirect multi-value link* is a type of multi-value link (MVL). It includes a join that relates the business component that contains the MVL to the parent business component. The source field of an indirect MVL references a column that Siebel CRM joins from another table and not from a column in the base table.

This figure describes how Siebel CRM creates an indirect MVL.



Explanation of Callouts in the Create an Indirect Multi-Value Link Figure

As shown in the figure, Siebel CRM uses these objects to create an indirect MVL:

1. **Joined business component.** Fulfills the child role in a parent-child relationship with the parent business component in the link. Siebel CRM creates the indirect MVL as a child of the join business component.
2. **Multi-value fields.** For more information, see *About the Multi-Value Field*.
3. **Multi-Value link.** The MVL uses these properties to create the relationship between the link and fields in the parent business component:
   - **Primary Id Field property.** Identifies the field from the business component that contains the MVL. For more information, see *About the Primary ID Field*.
   - **Destination Business Component property.** Identifies the child business component.
   - **Destination Link property.** Identifies the link.
4. **Joined field.** The Source Field property in:
   - A MVL is empty.
   - An indirect MVL defines a joined field in the same business component as the MVL. The joined field represents the ROW_ID column from the base table of the parent business component.

Siebel CRM gets the ROW_ID column through a join.

> **CAUTION:** Do not use a column other than ROW_ID. If you use a column other than ROW_ID, then you might experience unpredictable application behavior.

5. **Join and join specification.** Allows Siebel CRM to bring data into the joined field. For more information, see *About Joins*.
6. **Foreign key field in the joined business component.** Represents a foreign key column in the base table. The foreign key field references rows in the joined table. In this situation, this table is the base table of the parent business component. Siebel CRM uses the foreign key field to create the join.
7. **Base table.** The join, join specification, and foreign key field in the join business component access the base table of the parent business component. This makes a join relationship possible that provides a parent business component record and, indirectly, a set of child business component records for each join business component record.
8. **Child business component.** Supplies the child records in the parent-child relationship.
9. **Primary ID Field.** Identifies the foreign key field in the parent business component. For more information, see *About the Primary ID Field*.
10. **Foreign key field in the child business component.** Contains row ID values that reference back to records that reside in the parent business component. These row ID values uniquely identify the parent for each record in the child business component.
11. **Link.** Specifies the parent-child relationship between the parent business component and the child business component.
12. **Parent business component.** The parent in the parent-child relationship that's defined in the link.
13. **Key field.** The primary key for the parent business component.

Example of How Siebel CRM Creates an Indirect Multi-Value Link

This table describes some objects that Siebel CRM uses to create an indirect multi-value link (MVL) that involves the Business Address in the Contact business component. The Contact business component and the Account business component each contain the Business Address MVL.

Object	Name of Object Definition
Joined Business Component	Contact
Multi Value Link	Business Address
Joined Field	Joined Account Id
Join	S_ORG_EXT
Join Specification	Account Id
Foreign Key Field	Account Id
Base Table	S_ORG_EXT
Child Business Component	Business Address

Object	Name of Object Definition
Link	Account Address
Parent Business Component	Account

How Siebel CRM Uses the Source Field Property

If the Source Field property in a MVL is empty, then Siebel CRM uses the ID field in the current business component. This field corresponds to the ROW_ID in the base table. In the indirect MVL for the Contact business component, the Source Field property specifies the Joined Account ID field that resides in the S_ORG_EXT table. The Joined Account ID field provides the Account ID of the Account that corresponds to the current Contact.

The parent business component of a MVL is typically the same as the business component that contains the MVL. You can use the Source Field property of the link to create a MVL whose parent business component is related to the current business component indirectly through a join or another MVL.

How a Multi-Value Link References a Link

A link defines a one-to-many relationship between two business components. Typically, the business component that contains the MVL is the same as the parent business component of the underlying link that the MVL references.

For example, the following table lists some properties that Siebel CRM defines for the Business Address MVL in the Account business component.

Property	Value
Destination Business Component	Business Address
Destination Link	Account/Business Address
Primary Id Field	Primary Address Id
Check No Match	TRUE
Popup Update Only	TRUE

The Destination Link property indicates that this MVL references the Account/Business Address link. The following table lists some properties that Siebel CRM defines for the Account/Business Address link.

Property	Value
Name	Account/Business Address
Parent Business Component	Account

Property	Value
Child Business Component	Business Address
Destination Field	Account Id
Cascade Delete	Delete

The parent business component of the Account/Business Address link is the Account business component. The MVL resides in the Account business component. To update the multi-value group applet, Siebel CRM uses data from the children business address records for the account record that's currently chosen in the Account business component.

Usage of a Predefined Indirect Multi-Value Link

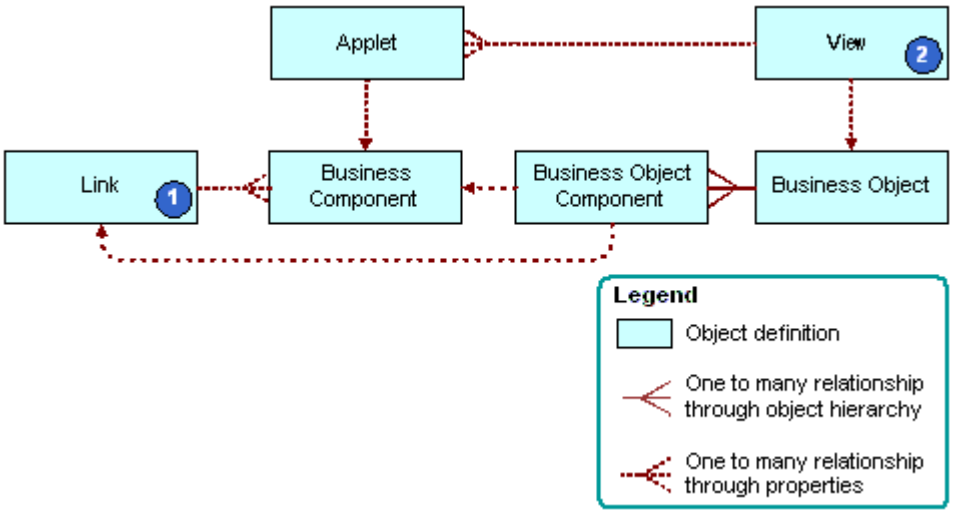
If a predefined link exists that's appropriate for use in a multi-value link (MVL), and if the originating business component is different from the parent business component, then you can use an indirect MVL instead of a conventional MVL. If a predefined join exists that joins the required parent business component to the parent business component of the link, then you can use the predefined link in the MVL.

About Links

A link defines a one-to-many relationship between two business components. For more information, see *Link*, *How Siebel CRM Handles a Hierarchy of Search Specifications*, *Hierarchy of Object Types and Relationships*, and *About Views*.

How a Business Object Uses a Link

The following figure describes how a business object uses a link.



Explanation of Callouts

As shown in this figure, a link that Siebel CRM uses with a business object includes the following objects:

1. **Link.** In a master-detail view, to create the parent-child relationship, Siebel CRM includes a link to a business object. This relationship applies to any use of the two business components that Siebel CRM uses in the context of the business object.
2. **View.** Each view references the business object that it uses in the Business Object property of the view. This configuration forces the view to operate as a master-detail view, as defined in the link, without more configuration of the view.

Visibility Rule Property of a Link

The Visibility Rule property of a link uses these values to determine if Siebel CRM displays the link:

- **Always.** Allows visibility rules in the child records when the current master-detail view references this link. This situation is true even if the Visibility Applet and Visibility Applet Type properties of the view are not defined.
- **Never.** Disables visibility rules in the child records if the current view references this link.

Creating a New Business Component using the Web Tools Wizard

To create a new business component using the web tools wizard

1. Log into the Web Tools client.
2. Open an editable Development Workspace.
3. Click the **New Object Wizard** button. It has a magic wand icon.
4. Choose the *Buscomp* icon and click Start.
5. In the first view there are six fields that you may configure.
    a. **Name**(Required): This is the name for your new Business Component. It must be a unique Name for a Business Component in the Repository.
    b. **Project**(Required): This is the Project in which to put the new Business Component.
    c. **Table**: The Table that this Business Component will use for its data.
    d. **Class**: This is the C++ class that the Business Component will use. The default is *CSSBusComp*.
    e. **Sort Specification**: This is an optional string to sort the data for the Business Component.
    f. **Search Specification**: An optional string to force the Business Component to restrict records based on search criteria.
6. Once you have configured the required fields and, optionally, the other fields. Click the Next button in the wizard.
7. In the next view you can choose the Columns in the Table to include in your Business Component. You can select one or many and move them over to the right side of the Applet. The Field names that appear in the Name column on the right are suggested by the Column itself. You can change them if desired.
8. Once you have moved all the Columns over or moved ones that you did not intend to move back to the left, click the Next Button.

9. Now the configuration of the Business Component is complete. Review your choices and amend them if necessary, by using the Previous button.
10. If you are satisfied with your configuration, click the Finish button.

5 About Business Objects

About Business Objects

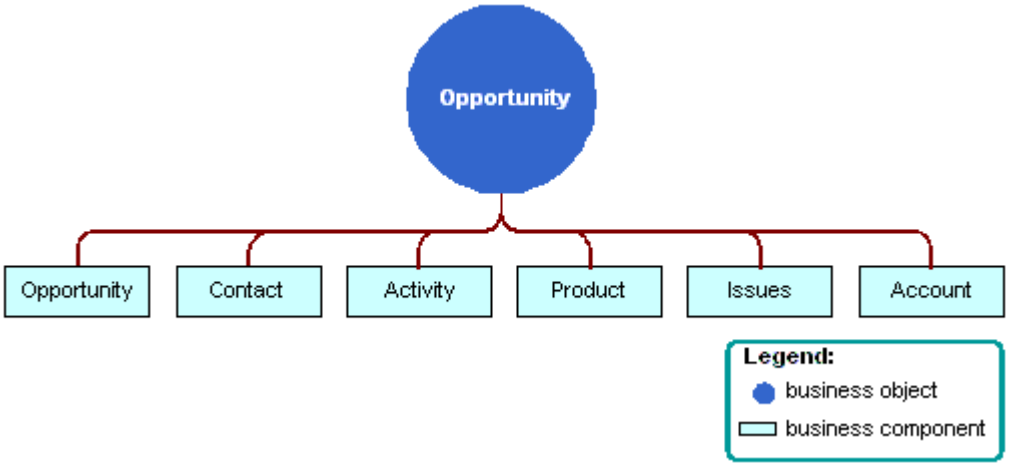
This chapter describes business objects and how to configure them. It includes the following topics:

- *Business Objects and Business Components, Views, and Screens*
- *How Siebel CRM Creates a Business Object*
- *Guidelines for Creating a Business Object*

Business Objects and Business Components, Views, and Screens

A business object represents a major area of the enterprise. An opportunity, account, or contact are examples of a business object. For more information, see *Business Object*. For an introduction to the relationships that this topic describes, see *Hierarchy of Object Types and Relationships*.

The following figure describes an example of how a business object groups business components into a logical unit. For example, the Opportunity business object groups together the Opportunity, Contact, Activity, Product, and other business components.



Each business object includes one business component that works as the parent business component. In the figure the parent business component is Opportunity. A link creates a relationship between the parent business component and other child business components, such as Contact and Product. This link allows the business object to display products that Siebel CRM relates to an opportunity or contacts that it relates to an opportunity.

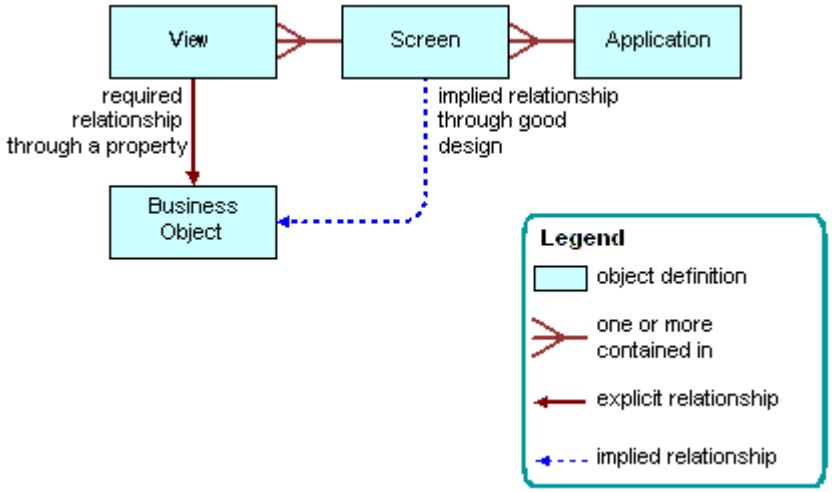
Relationship Between a View or Screen and a Business Object

A business object provides the foundation for a view and a screen. Typically, each view that a screen references uses the same data for the view when Siebel CRM gets the data from the same business component. For example, the Opportunities Screen references the following views:

- All Opportunity List View
- Opportunity Detail - Contacts View
- Opportunity Detail - Products View

Siebel CRM gets the data for each of these views from the Opportunity business component. The Siebel schema groups views that get most of their data from an opportunity into the Opportunity screen. Views in a screen typically get their data through the same business object. A screen is indirectly related to the business object.

The following figure describes the relationships and objects that Siebel CRM uses with a business object, screen, and view.



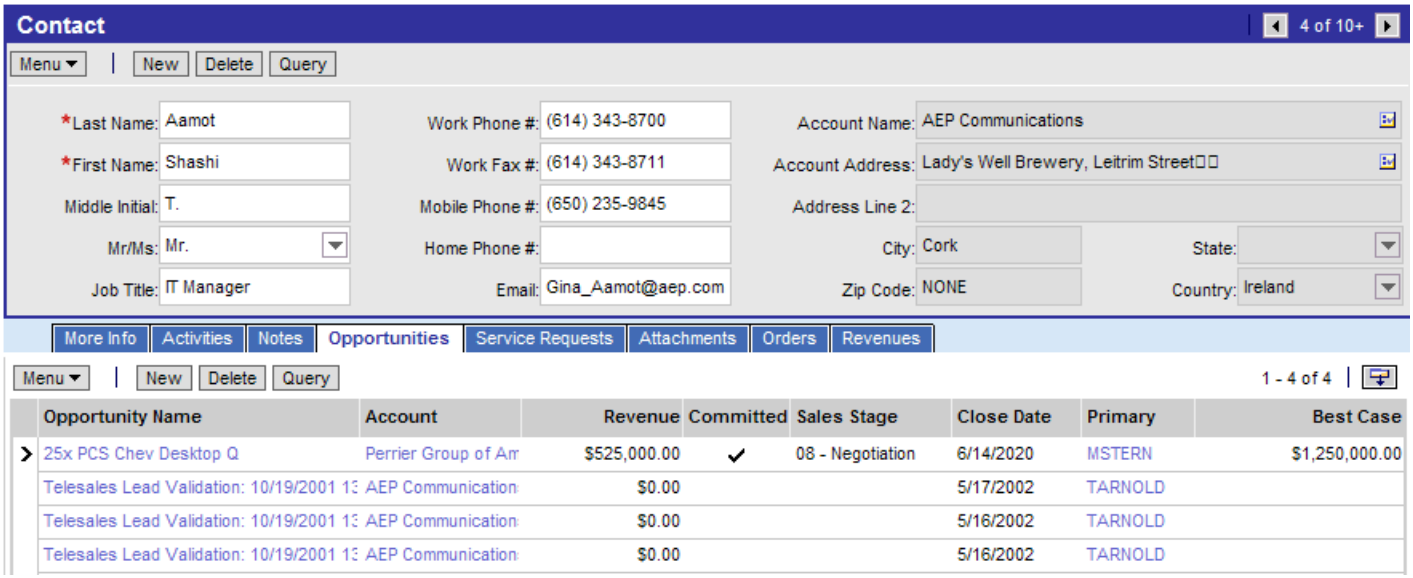
A one-to-one relationship typically exists between a screen and a business object. A view references a business object through a formal property of the view. A screen does not reference a business object through a formal property. An informal relationship exists between a business object and a screen. Siebel CRM applies design principles to create this informal relationship. Siebel Tools does not formally enforce this relationship. All the views that a screen contains are typically informally related to the same business object.

Not all business components that a business object references participate in a parent-child relationship. A business object can reference a business component that is not part of the business model.

Multiple business objects can reference a business component or a link. For example, two business components can each possess a one-to-many relationship in one business object. In the context of one business object, an unambiguous set of relationships exist between the business components that a business object references.

Example Parent and Child Relationships In a View That References a Business Object

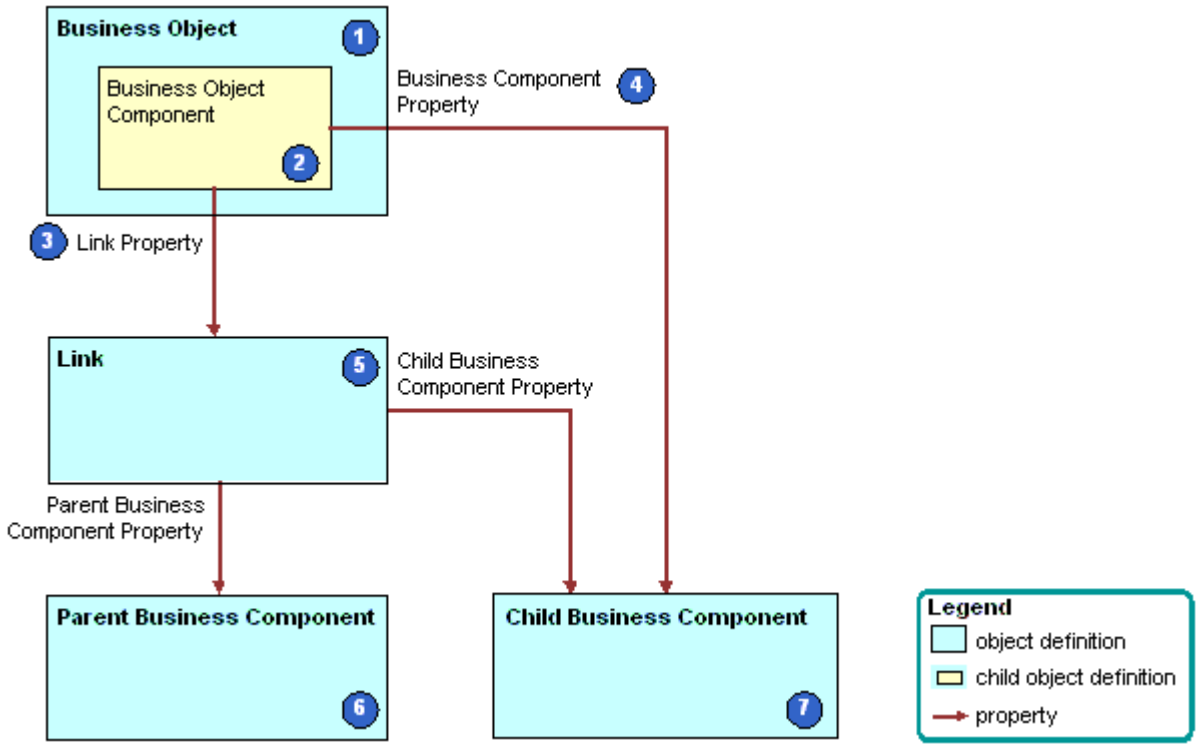
Each view references a business object. A master-detail view can define only a one-to-many relationship that the business object that the view references supports. To examine an example of this relationship, in the Siebel client, you can navigate to the Contacts List, drill down on the Last Name field of a contact, and then click the Opportunities tab. The parent Contact form displays prior to the Opportunities list. This contact to opportunities relationship is a one-to-many relationship that Siebel CRM defines in the Contact business object. To examine this relationship in Siebel Tools, locate the Contact Detail - Opportunities View in the Views list. This view references the Contact business object.



To implement a view that displays a many-to-one relationship between contacts and an opportunity, where many contact child records are related to one parent opportunity, a view references the Opportunity business object. To view this relationship in the Siebel client, navigate to the Opportunities List, drill down on the Opportunity Name field, and then click the Contacts tab.

How Siebel CRM Creates a Business Object

The following figure describes how Siebel CRM creates a business object.



Explanation of Callouts

As shown in this figure, Siebel CRM uses the following objects and properties to create a business object:

1. **Business object.** The parent for multiple business object components. Each business object component specifies a parent-child relationship. A view references the business object in the Business Object property of the view.
2. **Business object component.** A child object of the business object. Typically, each business object component defines one parent-child relationship in the parent business object. The Link property and the Business Component property of the business object component create this relationship.
3. **Link property.** Identifies the link.
4. **Business Component property.** Identifies the child business component. You can use a business object component to reference a business component in the business object without using a link. To do this, you must make sure the value in the Link property of the business object component is empty. This configuration allows you to include a business component in the business object for use in a view that references the business object, even though the business component does not possess a one-to-many relationship with another business component in the context of that business object.
5. **Link.** Each business object component references one link. This link specifies the parent-child relationship that the business object includes. For more information, see *About Links*.
6. **Parent business component.** The *one* in the one-to-many relationship that the link defines. The Parent Business Component property of the link specifies the parent business component.
7. **Child business component.** The *many* in the one-to-many relationship that the link defines. The following properties define the child business component:
    - The Child Business Component property of the link
    - The BusComp property of the business object component

Guidelines for Creating a Business Object

You only rarely need to create a new business object. The following situations might require you to create a business object:

- You require a new screen that groups several new business components together.
- You require a group of predefined business components that a predefined business object does not already support.

If you create a business object, then use the following guidelines:

- You can include a business component only one time in each business object.
- You can link a business component to only one other business component in the business object. For more information, see *Siebel CRM Can Link an Applet Only to One Other Applet in a View*.
- If you create a new business component to support an administration or system activity, then you do not need to create a new business object. Make sure the new business component is part of the predefined business object that Siebel CRM uses to support administration views, then assign the view to the Marketing Administration or System Administration screen.
- Delete any custom business object that Siebel CRM does not use and that does not reference any other object definition, such as a view.
- Other objects might reference an unused business object. Do not delete, deactivate, or rename any predefined business object that Siebel CRM does not use.

Guidelines for Defining the Link Property of a Business Object Component

If any of the following situations exist, then you can define the Link property of a business object component:

- If Siebel CRM can link the business component to more than one business component in the business object. For example, in the Opportunity business object, Siebel CRM can link the Action business component to the Opportunity, Account, or Contact business component.
- If the relationship between the parent business component and the child business component is a many–to–many relationship and where either business component can be the parent. For example, in the Opportunity business object, a relationship exists between the Opportunity business component and the Contact business component. Either business component can be the parent, so you can define the configuration so that Siebel CRM uses the Opportunity/Contact link. This configuration makes sure the Opportunity business object is the parent.

If you do not define the Link property, then Siebel Tools uses the Parent Business Component/Child Business Component link as a default. Siebel Tools sets the following properties for this link:

- The Parent Business Component property is the name of the source business object.
- The Child Business Component property is the value of the destination business component property.

If Siebel Tools cannot find a suitable link, then it displays the business component without a link to any other business component in the parent business object. In this situation, Siebel CRM displays all records that satisfy the search specification of the business component that are independent of the parent business component. This situation could create a problem because the user might not realize that the values in the child business component are not directly

related to the parent business component. In reality, these values represent all data for the child business component. If you must display records that possess a parent-child relationship, then you must enter a value for all links. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.

Siebel CRM Can Link an Applet Only to One Other Applet in a View

Siebel CRM can link a business component to only one other business component in the business object, and it can link an applet to only one other applet in a view. Except for the Home dialog box view, each view includes a parent applet that gets data from the parent business component in the business object. This parent applet can include related applets that get data from other business components. These applets are always child applets of the parent applet. A business component in the business object is the parent business component for the business object or it includes data that is related to the parent business component. For example:

- To display contacts that Siebel CRM relates to an opportunity, it must define a business object component that references the Contact business component. It must define this business object component on the Opportunity business object.
- To display the contacts that Siebel CRM relates to an account, it must define a business object component that references the Contact business component. It must define this business object component on the Account business object.

For more information, see *Guidelines for Naming an Object*.

6 About Applets, Controls and List Columns

About Applets, Controls and List Columns

This chapter describes applets, controls and list columns. It includes the following topics:

- *About the Form Applet and List Applet*
- *About Applet Controls and List Columns*
- *Options to Create an Applet*
- *Guidelines for Creating an Applet*
- *Guidelines for Creating a Control or List Column*

About the Form Applet and List Applet

This topic describes form applets and list applets. It includes the following information:

- *How Siebel CRM Creates a Form Applet*
- *How Siebel CRM Creates a List Applet*

An applet allows the user to view, enter, and modify data that it gets from a single business component. For more information, see *Applet*.

This topic describes the form applet and list applet, which are the most common types of applets. Many other types of applets exist. Some of the properties and concepts that the form applet and list applet use are found in other types of applets. For more information, see the following topics:

- *Configuring Special-Purpose Applets*
- *Configuring Multi-Value Group, Association, and Shuttle Applets*

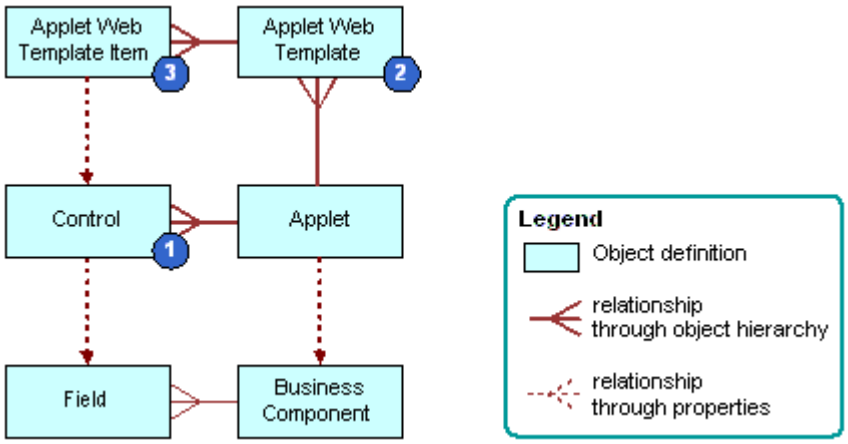
The applet user property allows you to define functionality beyond what is available as part of the applet class. For more information, see *Siebel Developer's Reference* .

How Siebel CRM Creates a Form Applet

A *form applet* is a type of applet that uses a form to display data from a business component. It includes the following qualities:

- Displays many fields for a single record.
- Provides a complete view of a record and are useful for data entry because the user can access all the necessary fields at once.
- Associated with a single business component.

The following figure describes how Siebel CRM creates a form applet. For more information, see *Hierarchy of Object Types and Relationships*.



Explanation of Callouts

As shown in this figure, Siebel CRM uses the following objects to create a form applet:

1. **Control.** Defines controls on the applet, such as a text box, check box, button, or link. For more information, see *About Applet Controls and List Columns*.
2. **Applet web template.** Associates an applet to a web template. A web template determines the layout and format of the applet when Siebel CRM displays the applet in the Siebel client. You can create an applet web template for each mode to display an applet in a different mode. For more information, see *About Siebel Web Templates* and *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.
3. **Applet web template item.** A child object of an applet web template. It references a control and identifies a placeholder tag or location in a web template. The placeholder determines where Siebel CRM locates the control in the web page. If you use the Applet Web Template Editor to relocate a control on to a web template, or if you use an applet wizard to create an applet, then Siebel Tools creates an applet web template item. For more information, see *Properties of the Applet Web Template Item*.

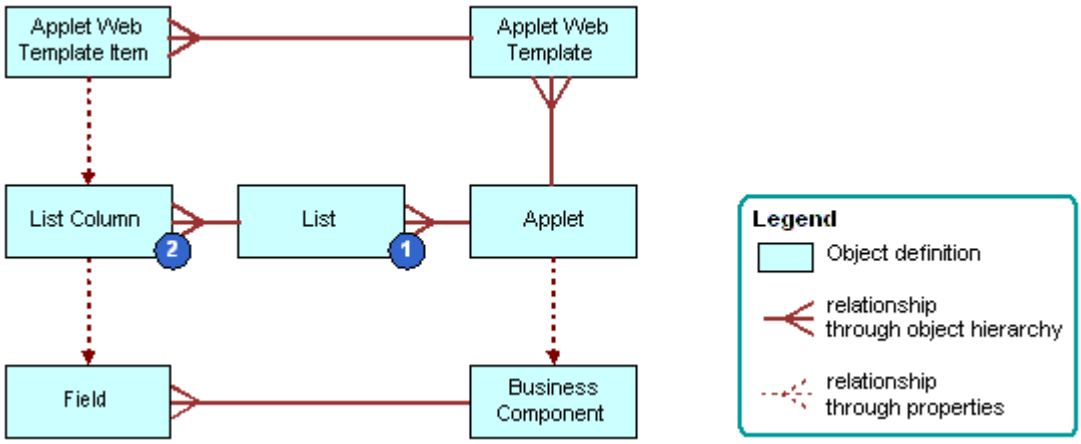
For more information, see *How a Business Component Field Provides Data to the Control or List Column of an Applet* and *Siebel Object Types Reference* .

How Siebel CRM Creates a List Applet

A *list applet* is a type of applet that displays multiple records at one time. It includes the following qualities:

- Uses multiple columns to display data in table format. Each row of the table represents a record from the business component that the applet references.
- Allows the user to scroll through multiple records of data and view several fields for each record.
- Associated with a single business component.
- A list column creates a relationship between the business component field and the applet web template item.

The following figure describes how Siebel CRM creates a list applet. Siebel CRM creates a list applet in a way that is similar to how it creates a form applet. For more information, see *How Siebel CRM Creates a Form Applet*. For background information, see *Hierarchy of Object Types and Relationships*.



Explanation of Callouts

As shown in this figure, Siebel CRM uses the following objects to create a list applet:

1. **List.** Includes properties that affect the list. It works as a parent object for all the list columns in the applet. A list applet includes one list object definition, named List.
2. **List column.** Identifies one column in the list. It references one field in the business component.

For more information, see *About Applet Controls and List Columns*.

About Applet Controls and List Columns

This topic describes applet controls and list columns. It includes the following information:

- *Types of Applet Controls and List Columns*

A *control* is an object that defines a user interface element, such as a text box, check box, or a button. Siebel CRM displays this element in the Siebel client. In a form applet, a control references a field in the business component that the applet references. A control creates a relationship between the business component field and the applet web template item.

A *list column* is an object that identifies one column in the list. It references one field in the business component. The user enters data in a list applet in a cell that resides at the intersection of a row and list column. A cell in a different list column can work differently, depending on the properties of the list column of the cell. The following examples describe cell behavior that references the properties of a list column:

- A cell can work similar to how a text control in a form applet works. This type of cell allows the user to view and edit text, numeric data, a date, or a currency. If the list column is not read-only, then the user can click the cell to edit the text.
- A cell can work in a way that is similar to how a check box control works in a form applet. A check mark in the check box indicates that the value for the check box is TRUE. An empty check box indicates that the value for the check box is FALSE. If TRUE, then a check box in a list column contains a check mark symbol, and a check box in a control in a form applet contains an X symbol.
- A cell that contains underlined, colored text is a *drilldown field.* For more information, see *Options to Drill Down to Another View*.

A form applet uses a control to display Siebel CRM data in the applet. A list applet uses a list column to display Siebel CRM data in an applet.

For more information, see *How a Business Component Field Provides Data to the Control or List Column of an Applet*.

You use the New Applet Wizard to create controls and list columns for a new applet. For a predefined applet, you use the Applet Web Template Editor to add, remove, or modify a control or list column. For more information, see *Adding a Control or List Column to an Applet Layout* and *Adding a Control or List Column to an Applet Layout*.

Types of Applet Controls and List Columns

Many types of applet controls and list columns exist that you can define in the HTML Type property of the control or list column. This topic describes some of the types that Siebel CRM commonly uses. For more information, see *Types of Applet Controls and List Columns* and *Siebel Object Types Reference* .

Siebel CRM does not support the .NET control type.

MiniButton Control and MiniButton List Column

You can use a MiniButton with a control or list column where Siebel CRM defines the Method Invoked property. If the user clicks the button, then Siebel CRM calls the method. This method can be predefined or you can create a custom method that you code in Siebel Visual Basic or Siebel eScript. Siebel CRM commonly uses the following types of MiniButtons:

- **MiniButton.** Displays a button.
- **MiniButtonEdit.** Displays a button with an Edit caption.
- **MiniButtonEditNew.** Displays a button with a New caption.
- **MiniButtonEditQuery.** Displays a button with a Query caption.

You must set the Runtime property of the button to TRUE. If the Runtime property is FALSE, then Siebel CRM does not run the method.

You can define the appearance and functionality of a minibutton by editing the CCHtmlType web template. For more information, see *Configuring an HTML Control Type*.

For more information, see *Using Declarative Configuration to Enable a Button*.

Text Control and Text List Column

A *text control* or list column displays text in a rectangular box. To view an example of a text list column in a Siebel application, such as Siebel Call Center, navigate to the Opportunities list. Note the Opportunity Name text list column in the opportunity list. For more information, see *Defining the Properties of a Control or List Column If HTML Type Is Text* and *Types of Applet Controls and List Columns*.

A text control or list column does the following:

- Allows the user to enter and edit text. If a text control or list column is read-only, then the user cannot enter text. A read-only text control or list column includes a shaded background and displays text that the user cannot edit.
- Displays as a list, multi-value group, calculator, or calendar icon, depending on the business component field that the control or list column references.
- Displays data of a data type, such as alphanumeric, numeric, date, or currency.

- The HTML Height property of the control determines the number of rows of text that Siebel CRM displays in the text box.
- Displays a select icon on the near edge of the text control if the MVG Applet property includes a value that is not empty or if the Pop-up Edit property is TRUE. This functionality allows the user to call up a multi-value group applet or a calendar or calculator widget.

  If the field must be a pop-up calendar or calculator control, then the Runtime property must equal TRUE.
- Displays a select icon on the near edge of the text control if the Pick Applet property references a pick applet. If the user clicks the select icon, then Siebel CRM displays the list. For more information, see *About Static Lists*.

If data includes trailing spaces, then Siebel CRM truncates the data if Siebel CRM displays it in a Siebel application or in Siebel Tools. This truncation includes full width spaces in Japanese.

Options to Create an Applet

This topic describes some options that are available if you create an applet. It includes the following information:

- *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*
- *Options to Filter Data That Siebel CRM Displays in an Applet*
- *Options to Determine How Siebel CRM Displays Controls and List Columns in a Siebel Application*

Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data

The *applet mode* is a type of behavior for an applet web template that determines if the user can or cannot create, edit, query, or delete Siebel CRM records in an applet. For more information, see *Controlling How the User Creates, Edits, Queries, and Deletes CRM Data*.

The following table describes the types of applet modes.

Applet Mode	Description
Edit	Allows the user to edit a record, create a new record, and query while working in a form applet. Siebel CRM uses the Edit and Edit List applet modes most frequently. If a New or Query applet template does not exist, and if the user creates or queries data, then Siebel CRM uses Edit mode. For more information, see *Types of Siebel Web Templates*.
Edit List	Allows the user to edit a record, create a new record, and query while working in a list applet. You can use the Edit List mode to allow the user to edit, create, and query in a Siebel application. Edit List mode displays a list applet as persistently editable. An editable list applet allows the user to modify the records in a list applet without switching to an edit page.
Base	Displays fields in read-only mode. Use this mode to display the applet in read-only mode until the user does something, such as clicking the Edit button. Siebel CRM displays a view in Base mode, by default.

Applet Mode	Description
	The user cannot edit or update a field in an applet that is in Base mode. The user can use the applet menu or right-click and use the pop-up menu to create a new record. The user can then edit or delete the new record, unless these operations are disabled at the applet or business component level. You can use the Read-Only field in the Responsibility Administration View to make a view read-only. For more information, see *Siebel Security Guide* .
New	Allows the user to create a new record where the requirements for the new mode are different from the edit or edit list mode. Only use the New and Query modes if the Edit mode does not meet your required functionality.
Query	Allows the user to query if the requirements for the Query mode are different from the requirements for the Edit or Edit List mode. **A**llows the user to perform a query-by-example (QBE).

Qualities of the Applet Mode

The applet mode includes the following qualities:

- Is a property of an applet web template, which is a child of an applet.
- Applies only to the active applet. For example, if the parent highest-level applet in a view is in Query mode, then the child lower applet is not in Query mode.
- Siebel CRM associates each mode with a web template.
- An applet can use one or more applet modes.
- You can use the Mode list in the Controls/Columns window of the Applet Web Template Editor to modify the applet mode. The applet web templates that Siebel CRM defines for the applet determine the modes that are available in the Mode list. For more information, see *Adding a Control or List Column to an Applet Layout*.
- Controls appearance of the minibutton. For more information, see *MiniButton Control and MiniButton List Column*.
- A multi-value group applet typically uses the Popup List template. If you define base, edit, or edit list mode for a multi-value group applet, then Siebel CRM references the mode from the parent applet. If you define only the base mode for a multi-value group applet, then Siebel CRM uses the base mode for the multi-value group applet regardless of the mode of the parent applet. For more information, see *Creating a Multi-Value Group Applet*.

For more information, see the following topics:

- *Process of Using the Applet Web Template Editor*
- *Process of Creating a Screen Home Page View*
- *Pick Applet Usage in Query Mode*
- *About Siebel Web Templates*

Options to Filter Data That Siebel CRM Displays in an Applet

A *search specification* is an expression you can define in the Search Specification property that filters the set of CRM data that Siebel CRM displays in an applet. This topic describes a search specification for an applet, but this information typically applies to the search specification for a business component, link, or list.

For more information, see the following topics:

- *Filtering Data That Siebel CRM Displays in an Applet*
- *Modifying Custom Search Specifications*

See also *Siebel Object Types Reference* and *Siebel Developer's Reference* .

The search specification contains the names of one or more fields in the business component and various operators. These items constitute a logical condition that identifies the records that Siebel CRM displays in the applet:

- If the result of the search specification is TRUE for a Siebel CRM record, then Siebel CRM displays the record in the applet.
- If the result of the search specification is FALSE for a Siebel CRM record, then Siebel CRM does not display the record in the applet.

The following search specification describes how you can filter CRM data so that Siebel CRM only displays records that contain a revenue that is greater than 5000:

```
[Revenue] > 5000
```

The following search specifications include more examples of how you can filter CRM data:

```
[Type]= "COST LIST"
[Competitor] IS NOT NULL and [Competitor] <> "N"
[Type] = LookupValue ("TODO_TYPE", "In Store Visit")
```

Parts of a Search Specification

The following table describes some of the parts of a search specification.

Element	Description
Comparison Operator	Compares the value in a field to a constant, or the value in one field to the value in another field. Siebel CRM allows the following operators: • = (equal to) • <> (not equal to) • > (greater than) • < (less than) • >= (greater than or equal to) • <= (less than or equal to) The following is an example search specification that uses the greater than comparison operator: `[Revenue] > 5000`
String Constant	You use double quotation marks to enclose the string constant. A string value is case sensitive. Uppercase and lowercase letters in a string constant must match exactly the string that the Siebel CRM record contains. The following is an example search specification that uses the **COST LIST** string: `[Type] <> "COST LIST"`
Logical Operator	The logical operators AND, OR, and NOT negate or combine elements in a search specification. Siebel CRM ignores case in these operators. For example, **and** does the same operation as **AND**. The following is an example search specification that uses the **AND** logical operator:

Element	Description
	`[Competitor] IS NOT NULL and [Competitor] <> "N"`
Field Name	You use square brackets to enclose a field name in a search specification. The following is an example search specification that references the Conflict Id field: `[Conflict Id] = 0`
LIKE Operator	The LIKE operator creates a text string search specification where the specification compares the value of a field to a constant, or compares the value of a field to the value of another field. A match on only the first several characters in the string is required. The LIKE operator uses the following wildcard characters: • *** (asterisk).** Indicates any number of characters. • **? (question mark).** Indicates a single character. The following is an example search specification that uses the LIKE operator: `[Last Name] LIKE "Sm*"` In this example, the Last Name values of Smith, Smythe, Smallman, and so on, causes the search specification to evaluate to TRUE.
Length	The search specification must not exceed 255 characters.

How Siebel CRM Handles a Hierarchy of Search Specifications

If multiple search specifications exist on an applet, business component, link, or list, then Siebel CRM uses the object hierarchy to determine how to run these search specifications. For example, if a search specification is defined on the applet and on the business component, then Siebel CRM does the following:

1. Appends the search specification on the applet to the search specification on the business component. Siebel CRM does not override the search specification on the business component. You cannot use a search specification on an applet to override a search specification that is defined on the underlying business component.
2. In the Siebel client, Siebel CRM converts the search specification on the applet to a WHERE clause.

How Siebel CRM Runs a Search Specification That Is Defined on a Child Applet

If a search specification is defined on a child applet, then Siebel CRM does the following:

- If a child applet references the same business component as the parent applet, then Siebel CRM does not run the search specification that is defined on the child applet.
- If a child applet does not reference the same business component as the parent applet, then Siebel CRM does the following:
  - To maintain the context for the search specification with the parent applet, it amends the search specification that is defined on the child applet with a WHERE clause.
  - Runs the search specification that is defined on the child applet.

How Siebel CRM Runs a Search Specification That Is Defined on a Link or List

If a search specification is defined on a link, then Siebel CRM does the following:

- The Search Specification property of a link applies to the child business component. If a search specification exists in the applet, then Siebel CRM uses an AND query operator to add the search specification that resides on the applet to the search specification that resides on the link.

A sort specification on a link only applies to an association list.

If a search specification is:

- Defined on a list, then Siebel CRM overrides any search specification that is defined on the business component.
- Not defined on the list, then Siebel CRM uses the search specification that is defined on the business component.

How Siebel CRM Handles a Search Specification if Multiple Applets Are Involved

If two applets reference the same business component, and if these two applets are included in the same view, then Siebel CRM creates one query against the Siebel database to update these applets. A database SELECT statement only supports one WHERE clause, so the following conditions apply:

- Only one of the applets can contain a search specification.
- If multiple applets each contain a search specification, then each search specification on each applet must be identical.

For example, Siebel CRM displays the Account List Applet and the Account Entry Applet in the Account List View. In the Account Entry Applet, it displays the record that the user chooses in the Account List Applet. If the user chooses a different row in the list or scrolls through the list, then Siebel CRM updates the Account Entry Applet to make sure the same record is chosen in the Account List Applet. Siebel CRM enters data into these applets from the same query, so the applets display the same record set.

You must not define the same search specification on the business component and on the applet. If you do this, then duplicate joins might result.

If a view must include two applets that must not display master detail relationships, then make sure each applet references a different business component. If these applets reference the same business component, then Siebel CRM might synchronize them in the Siebel client because of links that it defines in the business objects.

How the Applet Visibility Type Property Affects a Search Specification

If the Applet Visibility Type property of the view web template item includes a value that is not null, then Siebel CRM might ignore a search specification that is defined for the applets in this view. It is recommended that you use this property for applets in a view that reference a different business component. If you use this property, then you must test it thoroughly.

Defining the Search Specification Property or the Sort Specification Property

You define the Search Specification property or the Sort Specification property of an object the same way define the expression for a predefined query except that you do not identify the business component and you do not include the following reserved words:

```
Search
Sort
```

Example Expressions That the Search Specification Property Can Contain

The following are example expressions that the Search Specification property can contain:

- "[Close Date] > ""04/15/95"""
- "[Opportunity] LIKE ""C*"""
- "[Revenue] > 500000 AND [State] = ""CA"""
- "[Revenue] > 500000 OR [Revenue] < 10000"
- "([Revenue] > 500000 AND [State] = ""CA"") OR ([Revenue] > 200000 AND [State] = ""FL"")"
- "NOT ([State] = ""CA"")"

The entire search specification must reside on one line. If you use more than one line, then Siebel CRM displays a message that is similar to the following at run time:

```
"Invalid search specification..."
```

Note the following:

- Each field that an expression references must exist in the parent object, such as a business component or report, and it must use the format that the object type requires.
- If the user drills down on a record, and if the Search Specification property that is defined for the target applet is different than the Search Specification property that is defined for the originating applet, then Siebel CRM displays the first record of the destination view instead of the drilled-down record.
- A search that involves the Search Specification property is case-sensitive. You can use the ~ (tilde) modifier to make the search case-insensitive. For example, you can use the following code:

```
[Last Name] ~LIKE 'g*' or [Last Name] ~= 'GRANER'
```

Example That Includes a Sort in the Sort Specification Property

The following examples include a sort in the Sort Specification property:

- "[Close Date]"
- "[Opportunity] (DESCENDING)"
- "[Revenue]"
- "[Revenue] (DESCENDING)"
- "[Revenue] (DESC), [State]"

Sorting With a Predefined Query

> **Note:** Predefined Queries can be Workspace enabled in your development environment. If they have been Workspace enabled in your development environment and you are working in that environment, then you can only modify them in an editable Workspace. You do not need an editable Workspace to create and edit Predefined Queries in your Production environment.

To modify the expression that a saved query uses, you can use the PreDefined Query view to add a sort expression. You can specify one or more fields that refine an ascending or descending search. You can use the following format for a predefined query:

```
'Business Component Name'.Sort = "[Field] [[(DESC[ENDING])], [Field]
[(DESC[ENDING]],...]"
```

where:

- Business Component Name is the name of the business component that Siebel CRM sorts.
- Sort indicates a sort expression.
- Field is the name of the field that Siebel CRM sorts.

Sorting in the Client

To do a sort in the client, the user can sort items in a list column, use a predefined query, or you can configure the Search Specification property. To specify an ascending or descending sort after Siebel CRM gets data, the user can click a list column header in a list applet, and then click a sort button. For more information, see *Siebel Fundamentals* .

Sorting Versus Searching

Siebel CRM creates:

- The ORDER BY clause in the SQL code from the sort specification.
- The WHERE clause in the SQL code from the search specification.

Siebel CRM does not support the GROUP BY clause for a business component.

Options to Determine How Siebel CRM Displays Controls and List Columns in a Siebel Application

You can determine how Siebel CRM displays a control or list column in a Siebel application. For example, you can display or hide a control, or reposition it in the applet layout. The following values in the Application drop-down list in the Applet Web Template Editor determines how Web Tools maps a control or list column that you add, move, or delete:

- **All Applications.** The layout editor uses the All Applications option, by default. This configuration leaves the controls that you add or delete unmodified.
- **An application.** Web Tools applies any add, move, or delete of a control or list column only to the chosen application.

  If you choose another application in the Application drop-down list during an editing session, then Web Tools modifies the appearance of the Applet Layout to reflect the set of controls and list columns that Siebel CRM defines for this application.

  Siebel CRM does not display controls and list columns that it negates for this Siebel application.

  > **Note:** The feature described in this topic is available in Siebel CRM 18.10 Update and later releases.

How the Expression Property Determines If Siebel CRM Displays a Control or List Column

The Expression property of the applet web template item for the control or list column defines a condition that is specific to a Siebel application. The Expression property works like a search specification or query. If the Expression property:

- Is empty, which is the default, then Siebel Tools displays the control or list column in all Siebel CRM applications.

- Includes the name of a single Siebel application, such as Siebel eSales, then Siebel CRM displays the control or list column only in this Siebel application.
- Includes a negation expression before the name of a single Siebel application, such as NOT Siebel Web Sales, then Siebel CRM does not display the control or list column in this Siebel application.

You must not define the Expression property directly. If you choose one Siebel application in the Application list of the Configuration Context toolbar, and then modify the applet in the Applet Web Template Editor, then Siebel Tools sets the Expression property.

For more information, see *Properties of the Applet Web Template Item*.

How Siebel Tools Modifies the Expression Property of the Applet Web Template Item

If you choose one Siebel application in the Application list of the Configuration Context toolbar, then Siebel Tools makes the following modifications on the applet web template item:

- If you add a control or list column to an applet, then Siebel Tools sets the Expression property of the applet web template item to the name of the Siebel application. Siebel CRM displays the control only in the chosen Siebel application.
- If you delete a mapped control or list column from the applet layout, then Siebel Tools does the following:
  - Creates a new applet web template item.
  - Sets the Expression property of this new item to NOT *application*. For example, NOT Siebel Financial Services.
  - Sets the value in the Item Identifier property of this new item to the value that the Item Identifier property of the deleted control contains. For more information, see *Properties of the Applet Web Template Item*.

  At run-time, Siebel CRM displays the control in every application except *application*. For example, it displays the control in every application except Siebel Financial Services.
- If you move a mapped control or list column in the applet layout, then Siebel Tools creates a duplicate applet web template item named *Name*2. This new item includes an Expression property of *application* and a different Item Identifier property. Siebel Tools creates a NOT *application* object. Siebel CRM displays the control in a different location in this application. For more information about the item identifier, see *Properties of the Applet Web Template Item*.

If you delete the NOT *application* object, then the behavior reverts to All Applications.

How an Application Mapping Affects Wizards

Unlike a target browser-specific mapping that you create in the Target Browser list of the Configuration Context toolbar, a wizard does not affect how Siebel Tools maps an object for an application. If you use a wizard to create an object, then Siebel Tools creates the object for all applications.

Guidelines for Creating an Applet

This topic describes guidelines for creating an applet. It includes the following information:

- *Guidelines for Naming an Applet*
- *Guidelines for Creating a Check Box*

If you create an applet, then use the following guidelines:

- Keep the user interface consistent and intuitive.
- To reduce complexity, keep applet design simple.
- Avoid inactive objects.
- If possible, modify a predefined applet instead of creating a new applet. This configuration often requires less work and helps to minimize the objects that reside in the Siebel repository that you must maintain. For more information, see *Guidelines for Reusing an Applet*.
- To avoid unnecessary duplication, reuse an applet in multiple views and screens.
- If you do require a new applet, then set the Upgrade Ancestor property on each custom applet that you define from another applet. For more information, see *Guidelines for Setting the Upgrade Ancestor Property*.
- Define the applet mode:
  - If the applet is in read-only mode, then you only define it in Base mode.
  - If the applet is editable, then you must define it in Edit and Edit List modes.
- Do not display a field that no applet uses.
- You cannot add a user interface control in the column of a list applet. Buttons, picklists, and fields are examples of user interface controls.

Guidelines for Naming an Applet

If you name an applet, then use the following guidelines:

- Give duplicate applets that do not contain drilldowns the same name as the original applet but add the phrase *Without Navigation* to the name immediately before the word Applet. For example, ABC Selective Account List Without Navigation Applet.
- Add the phrase Administration Applet to the end of the name of each applet that the user uses to do administrative work. For example, Master Forecast Administration Applet.
- Name each new applet with a prefix that identifies your company. For example, if your company name is ABC Incorporated, then name the new applet ABC Opportunity List Applet.
- Include the type of applet in the name just before the word Applet.
- Capitalize the first letter of each word. For example, Account List Applet rather than account list applet.
- Do not use the following characters in an applet name:
  - backslash (\)
  - quotation mark (")
  - slash (/)
  - less-than symbol (<)
  - colon (:)
  - greater-than symbol (>)
  - asterisk (*)
  - vertical bar (|)
  - question mark (?)
  - tab characters

> **Note:** When naming objects in the Repository, only use elements of the ASCII character set. When pasting names or strings from programs such as Microsoft Word, you will get Word's equivalent of some characters such as " becomes “, and - becomes – in Word. When you paste those into the Repository it becomes an invalid name.

- Make sure the applet name is meaningful. Avoid adding a number suffix to an applet name, such as ABC Opportunity List Applet 2. For example, if the applet differs because it does not allow drill down, then indicate this situation in your applet name. For example, ABC Opportunity List Applet - Without Drill Down.

The following table describes the naming formats for an applet, where *business component* is the name of the business component that the applet references in the Business Component property. For more information, see *Guidelines for Naming an Object*.

Type of Applet	Name Format	Example
Association applet	*description* Assoc Applet	Opportunity Assoc Applet
multi-value group applet	*business component* Applet	Fulfillment Position Mvg Applet
Pick applet	*description* Pick Applet	Order Status Pick Applet
List applet	*business component name* List Applet	Account List Applet
Form applet	If the applet does not contain buttons, then use *business component name* Form Applet. If the applet contains buttons, then use *business component name* Entry Applet.	The following examples use this format: • Account Form Applet • Account Entry Applet
Chart applet	*description* Chart Applet - *description* Analysis	Bug Chart Applet - Severity Analysis
Tree applet	*description* Tree Applet	List of Values Tree Applet

Guidelines for Creating an Applet Title

If you create an applet title, then use the following guidelines:

- Always define the Title property of an applet. Do not leave it empty.
- Do not use the same title for more than one applet that Siebel CRM displays in the same view. If a view contains multiple applets that display data from the same business component, then distinguish the titles according to the type of applet. For example, use distinct titles in a view that displays accounts, such as *Account List Applet* or *Account Form Applet*.

The following table describes formats for an applet title. For more information, see *Guidelines for Naming an Object*.

Type of Applet	Title Format	Example
Association applet	Add *business component name*	Add Opportunities
multi-value group applet	*business component name*	Contacts
Pick applet	Pick *business component name*	Pick Product
List applet	*business component name* List	Account List
Form applet	Use one of the following formats: • *business component name* Form • *business component name* Entry	The following examples use this format: • Account Form • Account Entry
Chart applet	Use one of the following formats: • *type of action* Analysis • *description* by *description*	The following examples use this format: • Open Defect Analysis • Lead Quality By Campaign
Tree applet	*business component name*	Opportunities

Guidelines for Creating a Control or List Column

This topic describes guidelines for creating a control or list column. It includes the following information:

- *Guidelines for Creating a Check Box*
- *Guidelines for Creating a Text Control or List Column*

If you create a control or list column, then use the following guidelines:

- To simplify data entry, use the appropriate type of pop-up, if possible. For example, associate a calendar control with a date field, or a multi-line edit box with a multi-line text field.
- Align fields to the near margin in a form or list.
- Align labels to the far margin in a form.
- To accommodate multiple translations of a text string, include extra spaces in a display name or caption. For more information, see *Using Siebel Tools* .
- Associate a control for a form applet that uses the Applet Form 4 Column (Edit/New) web template to a field that is two columns wide, or to a field that is one column wide. To associate a control to a field that is two columns wide, you must set the HTML Width property to 412. If you do not create an HTML Width property, then Siebel CRM displays the control as a one column wide field even if it is associated to a two column wide field in a form applet.

Guidelines for Creating a Check Box

If you create a check box control or check box list column, then use the following guidelines:

- Make sure a check box is the appropriate way to display the data. If the data does not map to a yes or no reply, or if the meaning of the unchecked value is not obvious, then it is recommended that you use a list of values instead. For example, instead of using a check box labeled Standard, use a combo box labeled Shipping Method with a list of values that contain Standard and Next Day. For more information, see *Creating a List of Values*.
- Avoid using a negative. For example, instead of Not Required, use Optional.
- In a form applet, align the first characters in the labels and position them to before the check box.

Guidelines for Creating a Text Control or List Column

If you create a text control or text list column, then use the following guidelines:

- To display a calendar or calculator for a control, set the Runtime property of the control to TRUE.
- To display a pop-up editor, set the Popup Edit property of the control to TRUE.
- To make a list column unavailable, set the Available property to FALSE.
- To display a text list column in a list by default, set the Show in List property to TRUE.
- To hide a text list column by default, set the Show in List property to FALSE. The user can use the Columns Displayed dialog box to display the text control.
- A list column header is a specialized control, so it does not render HTML. For example, you cannot add an asterisk to indicate that a field is required, as you can in a form applet. For more information, see *Caution About Using Specialized Classes*.
- You cannot apply a background color to a list column.

7 About Views, Screens, and Applications

About Views, Screens, and Applications

This chapter describes views, screens, and applications. It includes the following topics:

- *About the Siebel Client Navigation Model*
- *About Views*
- *About Screens*
- *Options to Create a View or Screen*
- *About Applications*

About the Siebel Client Navigation Model

This topic describes the levels of the Siebel client navigation model.

The user navigates between tabs, links, and lists that Siebel CRM displays in one of the following levels of the Siebel client:

- **First level.** Screen tabs that allow the user to navigate between screens.
- **Second level.** Links to groups of views or single views. Siebel CRM can display these links in the following ways:
  - In the link bar directly after the screen tabs.
  - In a list that Siebel CRM displays in the header of an applet. Visibility rules typically determine how this list filters data. The My Contacts, My Team's Contacts, and All Contacts views are examples of these types of views.
- **Third level.** Tabs that allow the user to navigate to a group of detail views or to a single detail view.
- **Fourth level.** One of the following, depending on the web template that Siebel CRM uses:
  - Links in the link bar directly after the tabs.
  - Tabs on a grandchild applet.
  - Links in a list.

Other user interface elements provide the user with more navigation options, such as the Site Map, drilldowns, and the thread bar.

Related Books

Siebel Fundamentals

About Views

This topic describes views. It includes the following information:

- *About List-Form Views*
- *About Master-Detail Views*

The user can access a view in one of the following ways:

- Screenbar, which displays the default view for that screen
- The second-level visibility list
- A third-level tab
- The fourth level list for the category view
- The thread bar
- The history list
- History forward and back buttons
- Drilldown from another view

The navigational devices in the physical user interface determines access to certain views.

For more information, see *Overview of the Logical User Interface Object Layer*.

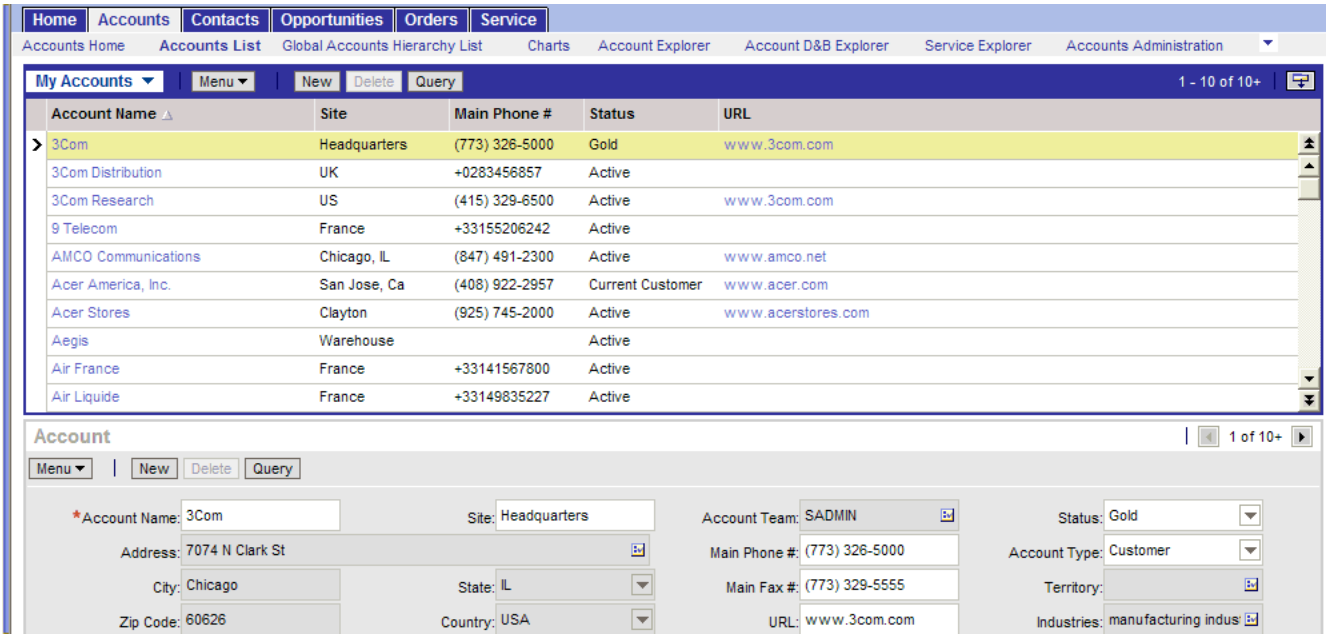
About List-Form Views

A *list-form view* (as shown in the following image) is a type of view that includes a list applet and a form applet that displays data from the same business component. Siebel CRM displays the list applet prior to the form applet. It displays a list of records. The form applet displays detailed information about the record currently chosen in the list applet.

To view an example of a list-form view, you can open Siebel Call Center, navigate to the Accounts Screen, and then the Accounts list. Note the following:

- Siebel CRM displays the Account List Applet and the Account Entry Applet.
- The list applet displays a list of account records and the form applet displays details about the account that is chosen in the list, but in a format that the user can view without scrolling.

These applets reference the Account business component.



About Master-Detail Views

A *master-detail view* is a type of view that typically includes a form applet and a list applet that displays data from two different business components. A link defines a parent-child relationship between the two business components. Siebel CRM displays the form applet prior to the list applet and displays one record from the parent business component. The list applet displays all of the records from the child business component that Siebel CRM associates with the record that is chosen in the form applet.

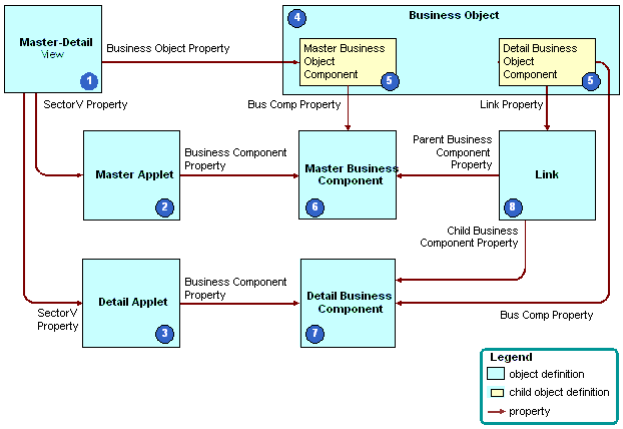
A view can include two list applets in a master-detail view. The records in the detail list applet are child records of the record that is currently chosen in the parent list applet.

To view an example of a master-detail view in Siebel Call Center, you can navigate to the Accounts screen, and then the Accounts list. Click the link in the Account Name field. Note the following:

- Siebel CRM displays the Account Contact List Applet. It references the Contact business component.
- Siebel CRM displays the Account Entry Applet. It references the Account business component.
- The view references the Account business object.
- In the context of the Account business object, the parent-child relationship between Account and Contact references the Account/Contact link.

How Siebel CRM Creates a Master-Detail View

The following figure describes how Siebel CRM creates a master-detail view.



Explanation of Callouts

As shown in this figure, Siebel CRM uses the following objects to create a master-detail view:

1. **Master-detail view.** The object definition of the view.
2. **Master applet.** The form applet that displays the parent record.
3. **Detail applet.** The list applet that displays the child records.
4. **Business object.** The business object that the Business Object property of the view references. The business object creates the context that determines the active link between the business components that the applets reference.
5. **Business object components.** Child objects of the business object. Each business object component associates a business component to the business object.
6. **Master business component.** The business component that the parent applet references.
7. **Detail business component.** The business component that the detail applet references.
8. **Link.** The link that specifies the parent-child relationship that exists between the parent business component and the child business component. The Link property of the detail business object component identifies the link.

For more information, see the following topics:

- *About Business Components*
- *About Links*
- *Business Objects and Business Components, Views, and Screens*

About Screens

This topic describes screens. It includes the following information:

- *About Screen Views*
- *Guidelines for Creating a View*

A *screen* is a collection of related views:

- A screen displays a logical grouping of views that pertain to one business operation.
- All the views in a screen typically reference a single business object.
- To simplify navigation, you can group views in a screen into categories.

The user can access a screen through a screen tab or the Site Map. The links to each screen are defined as part of the page tab object definition, which is a child of the screen. The screen defines the default view that Siebel CRM displays if the user clicks a screen tab.

A screen includes a child screen view. The screen view controls the views that Siebel CRM displays in the Siebel client if the user chooses a screen tab.

The Site Map is limited to nonvisibility views. Siebel CRM does not display a visibility level view, such as My Accounts or My Team's Accounts, on the Site Map.

For more information, see *About the Siebel Client Navigation Model* and *Process of Creating a Screen*.

About Screen Views

A *screen view* is an object that displays groups of views or a single view in the Siebel client. It allows you to group related views together and to control the location where Siebel CRM displays links in the Siebel client. The Type and Parent Category properties determine where Siebel CRM displays the screen view in the Siebel client. The screen view plays a major role in determining where Siebel CRM displays a view in the Siebel client, but a view web template ultimately controls appearance. For example, most web templates display links under tabs. Other web templates display these links in a list. For more information, see *Creating a Screen View*.

The following table describes types of screen views.

Type of Screen View	Description
Aggregate Category	Groups all remaining screen view types. Siebel CRM displays it as a link in the link bar after screen tabs.
Aggregate View	Siebel CRM displays an Aggregate View as follows: • If no value is defined for the Parent Category property, then Siebel CRM displays the screen view as a link in the link bar after the screen tabs. • If the Parent Category property contains a valid Aggregate Category, then Siebel CRM displays the screen view as a link in the view list in applet headers.
Detail Category	Groups detail views. Siebel CRM displays it as a tab.
Detail View	Siebel CRM displays a Detail View as follows: • If the Parent Category property contains a valid Aggregate Category, then Siebel CRM displays the screen view as a tab. • If the Parent Category property contains a valid Detail Category, then Siebel CRM displays the view as a link in a link bar after the tabs, or in another location depending on the web template. For example, Siebel CRM can display the screen view in a view list or in another row of tabs. Siebel CRM defines a nonvisibility view as a detail view with the Parent Category property set depending on the business object that the view references.

How Siebel CRM Groups Aggregate Categories

Visibility views are grouped under aggregate categories according to the business object. For example, some of the views in the Accounts Screen belong to the Account and Global Account Hierarchy business objects:

- Siebel CRM groups the visibility views that it associates with the Account business object under the Accounts List aggregate category.
- Siebel CRM groups the visibility views it associates with the Global Account Hierarchy business object under the Global Accounts Hierarchy List aggregate category.

How Siebel CRM Uses Screen Views in Each Navigation Level

The following table describes the type of screen view that is defined for each navigation level and the properties for each screen view. For more information, see *About the Siebel Client Navigation Model* and *Process of Creating a Screen*.

Navigation Level	Siebel Client Placement	Object That Is Defined
1	Screen tabs	Page Tabs, which are child objects of an application. For more information, see *Creating a Screen Menu Item*.
2	Links in the link bar directly after screen tabs. These links refer to groups of views.	Screen view with the following properties: • Type is Aggregate Category • Parent Category is empty
	Links in the link bar directly after screen tabs These links refer to a single view.	Screen view with the following properties: • Type is Aggregate View • Parent Category is empty
	Links in the view list in an applet header	Screen view with the following properties: • Type is Aggregate View • Parent Category is Aggregate Category
3	View tabs These tabs refer to groups of detail views.	Screen view with the following properties: • Type is Detail Category • Parent Category is an Aggregate Category
	View tabs These tabs refer to a single detail view.	Screen view with the following properties: • Type is Detail View • Parent Category is an Aggregate Category
4	Links in the link bar after view tabs, or in an alternate location depending on the web template.	Screen view with the following properties: • Type is Detail View • Parent Category is Detail Category

Example of a Screen View Hierarchy

The following table describes several screen views from the Account screen and some of the properties that determine the location of the screen view in the Siebel client.

Level	Siebel Client Location	Type	Category Name	Category Default View	View	Parent Category
2	Link in link bar after screen tabs	Aggregate View	Not applicable	Not applicable	Account Screen HomePage View	Not applicable
	Link in link bar after screen tabs	Aggregate Category	Account List	Account List View	Not applicable	Not applicable
	Link in view list in applet header	Aggregate View	Not applicable	Not applicable	Account List View	Account List
3	View tabs	Detail View	Not applicable	Not applicable	Account Detail - Contacts View	Account List
	View tabs	Detail Category	ESP	ESP Account Plan Overview View	Not applicable	Account List
4	Link in link bar after view tabs	Detail View	Not applicable	Not applicable	ESP Account Plan Overview View	ESP

Guidelines for Creating a View

If you create a view, then use the following guidelines:

- Use the guidelines for configuring access control. For more information, see *Siebel Security Guide* .
- Do not associate a view with more than one screen. If you do this, then problems with the Thread Manager might occur. When Siebel CRM saves a thread in the session file, it stores the name of the view without the name of the associated screen. If the user chooses a thread that navigates to a duplicate view, then Siebel CRM always navigates the user to one screen, even if it created the thread in the other screen. If you define the duplicate view as the default view on both screen tabs, then the user experiences an anomaly in the Siebel client. Siebel CRM chooses one screen tab as the active tab. It never displays the duplicate screen tab as an active tab. For more information, see *Configuring the Thread Bar*.
- Do not modify a view that Siebel CRM displays in the Administration - Server Configuration screen or in the Administration - Server Management screen. Siebel CRM reads information in these views from the Siebel Gateway registry. The Server Manager displays these views in the Siebel client. Siebel CRM does not support modifying a server view.
- Due to the specialized nature of the code that the calendar references, supported changes to the Siebel Calendar are limited to those specifically described in this guide and in *Configuring Siebel Open UI* .

Guidelines for Naming a View

If you name a view, then use the following guidelines:

- Use a prefix that identifies your company. For example, name a new view for ABC Incorporated as ABC Opportunity Detail - Tasks View.
- Make the view name meaningful. Avoid adding a number suffix to a predefined name, such as Opportunity List View 2.
- If the view differs because it is read-only, then indicate that it is read-only. For example, ABC Opportunity List View - Read Only.
- Capitalize the first letter of each word. For example, Opportunity List View rather than opportunity list view.
- Do not use the following characters in Siebel object names:
  - backslash (\)
  - quotation mark (")
  - slash (/)
  - less-than symbol (<)
  - colon (:)
  - greater-than symbol (>)
  - asterisk (*)
  - vertical bar (|)
  - question mark (?)
  - tab characters

> **Note:** When naming objects in the Repository, only use elements of the ASCII character set. When pasting names or strings from programs such as Microsoft Word, you will get Word's equivalent of some characters such as " becomes “, and - becomes – in Word. When you paste those into the Repository it becomes an invalid name.

Guidelines for Naming a View According to the Type of View

The following table describes guidelines for naming a view according to the type of view.

Type of View	Name Format	Example
List-form view	*business component* List View	Account List View
Master-detail view	*detail business component* Detail - *master business component* View	Opportunity Detail - Contacts View
Explorer view	*business component* Explorer View	Account Explorer View
Chart view	*master business component* Chart View - *detail business component* Analysis	Account Chart View - State Analysis

Guidelines for Naming a View According to the Type of Aggregate View

The following table describes guidelines for naming a view according to the type of aggregate view. The text in italics indicates the text that Siebel CRM modifies according to the underlying entity. For more information, see *Guidelines for Naming an Object*.

Type of Aggregate View	Example
Personal	My Personal *Contacts*
Sales Rep	My *Contacts*
Manager	My Team's *Contacts*
Organization	All *Contacts*
Sub Organizations	All *Accounts* Across My Organizations
All	All *Contacts* Across Organizations
Group	User *Catalog List View*
Catalog	*Products* Across Catalogs
Admin Mode	*Contacts* Administration

Options to Create a View or Screen

This topic describes options to create a view or screen. It includes the following information:

- *Options to Drill Down to Another View*
- *Options to Toggle Between Applets in a View*

For more information, see the following topics:

- *Configuring Views, Screens, and Applications*
- *Guidelines for Reusing a Predefined View*

Options to Drill Down to Another View

A *drilldown* is a type of field that allows the user to navigate from a field to another view that displays more information about the chosen record. Siebel CRM displays a drilldown primarily in a list applet. The drilldown object is a child object of an applet. A drilldown can be static or dynamic.

Consider the following drilldown behavior:

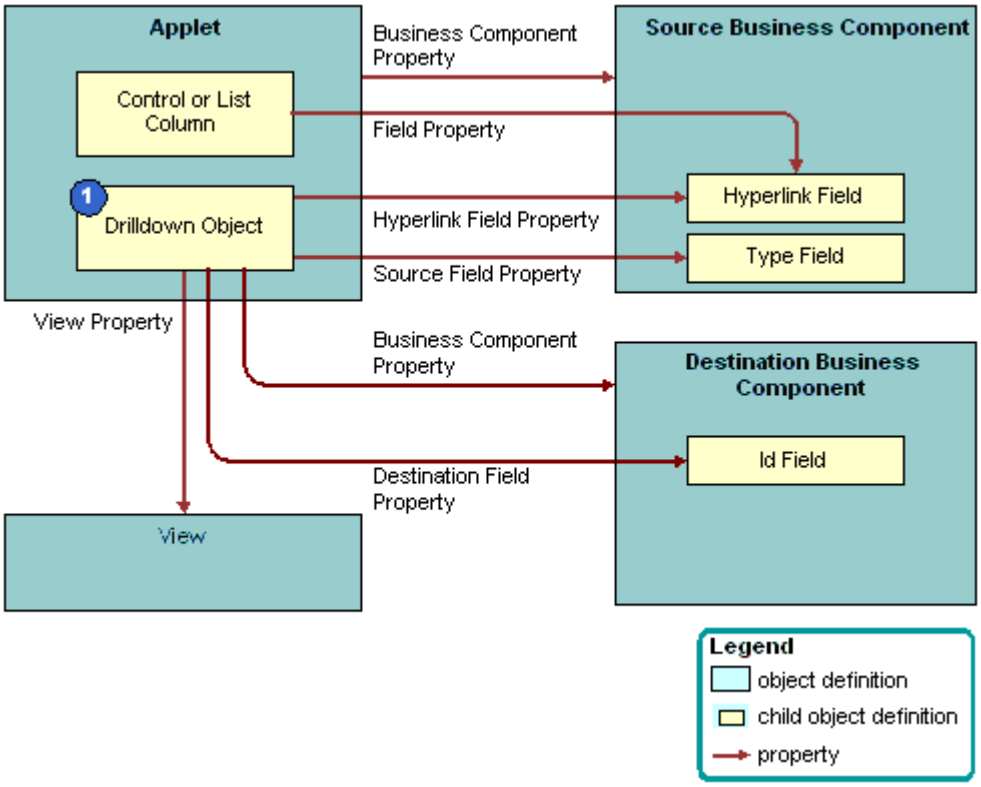
- If the parent applet of a view includes a search specification, and if the user drills down on a field, then Siebel CRM applies this search specification to the destination view. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.
- If the target view of a drilldown includes a visibility type that is different from the original view, and if the user drills down on a field, then Siebel CRM navigates the user to the first record of the destination view and not to the drilldown record.

Siebel CRM does not support drilldown on a multi-value group applet, pick applet, or association applet.

How Siebel CRM Creates a Static Drilldown

A *static drilldown* is a type of drilldown that navigates the user to the same view.

The following figure describes how Siebel CRM creates a static drilldown.



Explanation of Callouts

As shown in this figure, Siebel CRM uses the following objects to create a static drilldown:

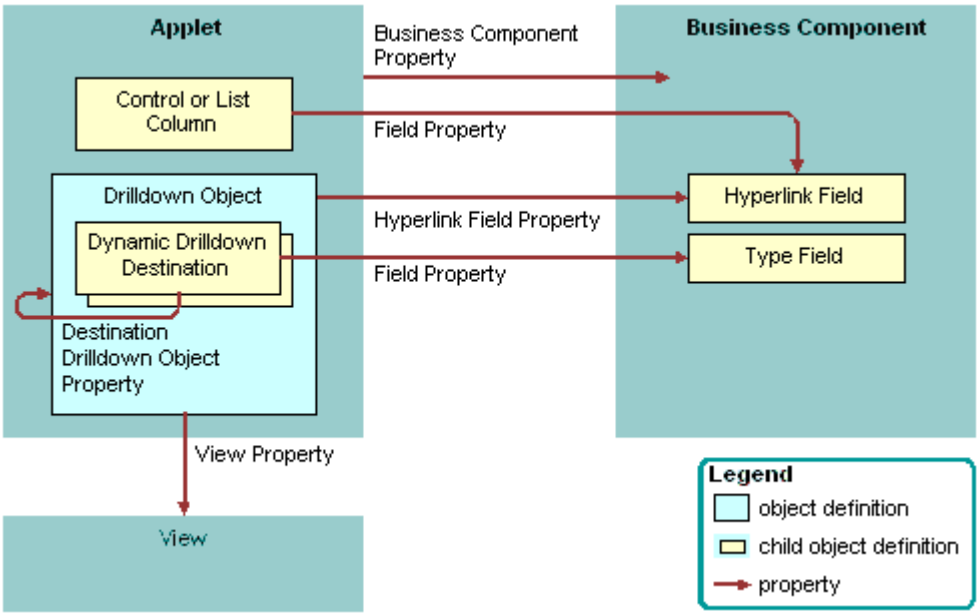
1. **Drilldown Object.** Identifies a link field and a view. These properties define the list column or control that includes a link and the destination view that Siebel CRM displays if the user drills down on the link.

Objects Siebel CRM Uses to Create a Dynamic Drilldown

A *dynamic drilldown* is a type of drilldown that navigates the user to a different view. This navigation depends on a condition, such as the value of a field. A dynamic drilldown allows the user to navigate to multiple views from the same link field, depending on the value of a field in the current record of the applet. This functionality is useful if some processing is required for various types of contacts, opportunities, accounts, and so on. For example, the

business component might include a field where Siebel CRM can evaluate the condition, such as the Lead Quality of an opportunity or the primary Industry of an account. The drilldown then navigates the user to a different view depending on the value in the field.

The following figure describes the relationships between objects in a dynamic drilldown. To create a dynamic drilldown, you define one or more dynamic drilldown destination children of the drilldown object for the field and the corresponding list column or control.



The functionality of the drilldown object in a dynamic drilldown is the same as it is with a static drilldown with the following exceptions:

- Siebel CRM defines a drilldown object for each candidate view.
- Each dynamic drilldown destination specifies a condition.
- The drilldown object that contains the *lowest* sequence number includes child dynamic drilldown destinations that define the following conditions that Siebel CRM uses for each of the drilldown objects:
    - If the conditions in the dynamic drilldown destination are true, then Siebel CRM flows to one of the drilldown objects.
    - If the conditions in the dynamic drilldown destination are false, then Siebel CRM uses the parent drilldown as the default drilldown.
    - If the conditions in the dynamic drilldown destination are true but the user is not assigned the responsibility that is required to access the destination view, then Siebel CRM uses the parent drilldown as the default drilldown.

For example, assume the Industry field in the Account business component is designated as the type field in a list of dynamic drilldown destinations:

- If the Industry is Manufacturing, then the drilldown navigates to a drilldown object that includes a view that is tailored for a manufacturing account.
- If the Industry is Transportation, then the drilldown navigates to a drilldown object that includes a view that is tailored for a transportation account.

CAUTION: You must avoid defining a link that routes from one dynamic drilldown object to another dynamic drilldown object. If you create child dynamic drilldown destinations of a drilldown object, then make sure they do not route to a drilldown object that includes child dynamic drilldown destinations. This configuration might cause ambiguity or looping problems.

How Siebel CRM Handles a Dynamic Drilldown If Multiple or No Conditions Are Met

If the condition in one dynamic drilldown destination is met, then the link navigates to the defined drilldown object. If more than one condition is met, then Siebel CRM uses the lowest value that the Sequence property contains to identify the first condition that it uses as the destination drilldown object. If no condition is met, or if no dynamic drilldown destinations exist that are children of the drilldown object, then the drilldown object supplies the name of the destination view.

If you define multiple drilldown objects for an applet, then you can reference any field in the business component only one time for all available drilldown objects. For a dynamic drilldown, you can set the Hyperlink Field property of the drilldown object that contains the dynamic drilldown destinations.

Options to Toggle Between Applets in a View

An *applet toggle* is a feature that allows the user to navigate back and forth between different applets in the same view. This feature allows you to display different types of data or to display the same data in a different way. The following types of applet toggles are available:

- **Static applet toggle**. Allows the user to choose the name of the applet from the Show list to toggle between applets.
- **Dynamic applet toggle**. Toggles between applets that reference the value of a field in a parent applet.

An applet toggle includes the following configurations:

- Siebel CRM applies the search specification on the form applet in the view. To create a search specification on a list applet during a toggle, you must add the search specification for the form applet. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.
- A static toggle applet is not required to reference the same business component.
- You can define only one static applet toggle in a single view.
- A dynamic toggle Applet must reference the same Business Component, which can be a predefined Business Component or a Virtual Business Component. The point of a dynamic toggle Applet is to, based on the value of a Field in the parent Applet, display the Business Component data in a different way. That is why both Applets must reference the same Business Component.
- You can define more than one dynamic applet toggle in a single view.
- You can define one static applet toggle and one dynamic applet toggle in a single view.
- You can define one static applet toggle and multiple dynamic applet toggles in a single view.

  CAUTION: If you define more than one static applet toggle in a single view to access multiple views, then unpredictable behavior might result. Instead, it is recommended that you use detail views with the Parent Category property set to Detail Category. For more information, see *About Screen Views*.

- You cannot define multiple static applet toggles in a single view.
- You cannot create a static applet toggle and a dynamic applet toggle in the same applet.

For more information, see *Creating an Applet Toggle* and *Improving Performance When Using Applet Toggles*.

About Applications

An *application* is an object that includes a collection of screens. Siebel Call Center and Siebel Partner Relationship Manager are examples of applications. You can create a new application, but it is recommended that you modify a predefined application to meet your business requirements.

The application object defines the screens that the user can access through a menu or tab. The following child objects of the application object can associate a screen with the Siebel application:

- **Page tab.** Adds a screen to the tab bar. For more information, see *Page Tab*.
- **Screen menu item.** Adds a screen to the Site Map.

An application object definition includes the following items:

- **Find Objects.** Configures the Find dialog box. For more information, see *About Screens*.
- **Server script and browser script.** Can be defined as an event procedure on startup, prior to closing, and so on. You define these scripts through an Application Script child object. You use the Script Editor to create and maintain a script. For more information, see *Siebel VB Language Reference* , *Siebel eScript Language Reference* , and *Siebel Object Interfaces Reference* .
- **Custom menu option for a Siebel method.** Defined with an applet method menu item and created in the Applet Method Menu Item Wizard. For more information, see *Applet Method Menu Item Object Type*.

For more information, see *Creating and Deploying an Application* and *How Siebel CRM References Web Pages*.

Guidelines for Creating an Application

If you create an application, then use the following guidelines:

- The name of the Siebel application is case-sensitive and space-sensitive.
- To identify the name of the Siebel application, use the appropriate parameter in the configuration file of the Siebel application.
- To minimize locking of the application object, Siebel CRM contains the object in a separate project.

8 About Siebel Web Templates and Siebel Tags

About Siebel Web Templates and Siebel Tags

This chapter describes Siebel web templates and Siebel Web Engine (SWE) tags. It includes the following topics:

- *About Siebel Web Templates*
- *About View Web Templates*
- *About Applet Web Templates*
- *About Siebel Tags*
- *Guidelines for Configuring Siebel Web Templates and Siebel Tags*

For more information, see *How the Siebel Web Engine Creates a Siebel Application* and *Configuring Siebel Web Templates and Web Pages*.

If you must modify a list applet in Siebel Open UI, then you must use a specific JavaScript file to do the rendering. You cannot modify only the Siebel web template. For more information about customizing Siebel Open UI, see *Configuring Siebel Open UI* .

About Siebel Web Templates

This topic describes Siebel web templates. It includes the following information:

- *Overview of Siebel Web Templates*
- *How Siebel CRM References Web Pages*
- *How Siebel CRM Uses HTML Frames in the Container Page*

Overview of Siebel Web Templates

A Siebel web template includes a preset format that Siebel CRM reuses each time it requires a particular layout. This configuration allows Siebel CRM to use only a single template rather than multiple files every time it requires a particular layout. For more information, see *Siebel Web Template*.

A web browser uses HTML to define the layout and format of a page. Siebel web templates provide this HTML layout to the Siebel Web Engine when it displays Siebel objects in the Siebel client. The templates contain markup tags, such as HTML and XML, that Siebel CRM intersperses with Siebel tags. Siebel CRM prefixes these tags with swe in Siebel Tools and od in Web Tools.

A Siebel web template includes empty placeholders that contain no data. To enter data and user interface elements into a template,Siebel CRM associates views, applets, controls, and other objects with each template. These objects are defined in the Siebel repository. Siebel CRM maps each object to an empty placeholder in the template. For example, assume a view maps to three applets. You associate a view web template with the view, and then map each applet to a placeholder in that template.

Types of Siebel Web Templates

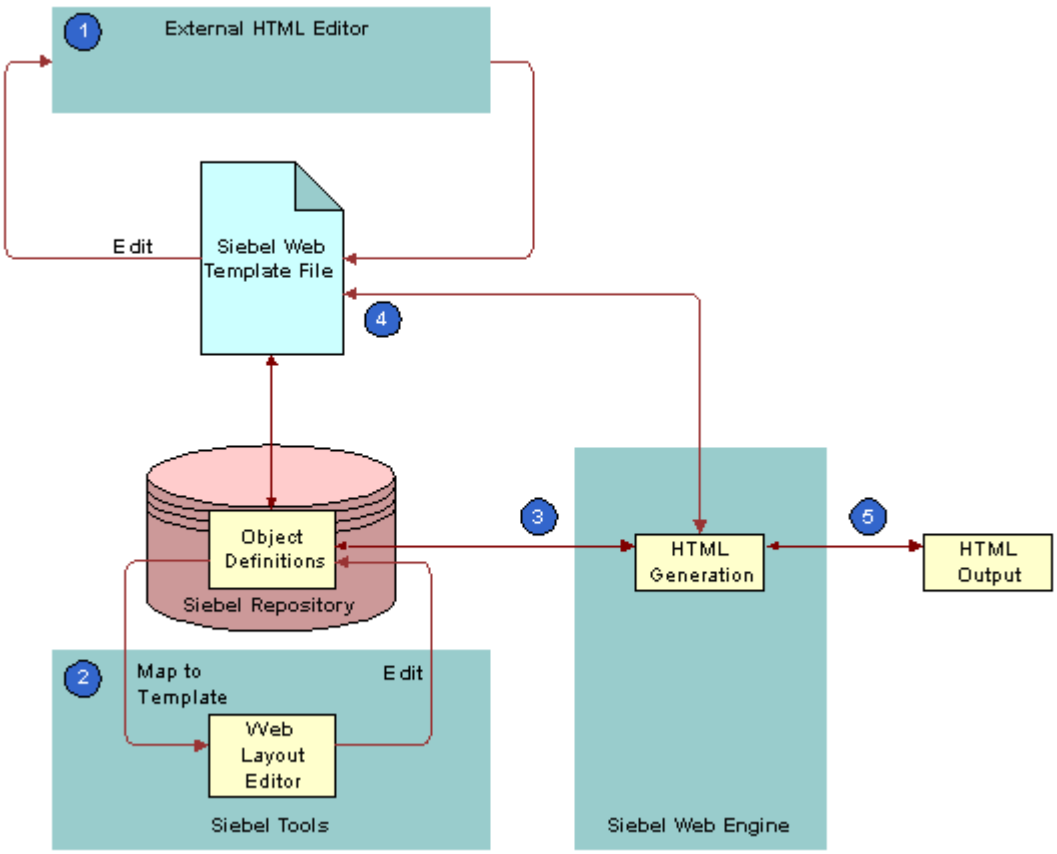
Siebel CRM uses the following types of Siebel web templates:

- **Page container template.** Provides a structure for Siebel CRM. Each Siebel application uses one page container that it uses as a container for view web templates. A view that does not use the container page typically varies significantly from other views. For example, the login page does not use the page container.
- **Web page template.** Defines the layout for the entire display. Includes information about where Siebel CRM displays the screen bar, view bar, and view.
- **View web template.** Defines the layout for a view. Specifies where to position applets and other page-level controls on the view. Specifies the format of the view.
- **Applet template.** Defines the layout for fields and controls in an applet. Specifies how to format elements in an applet. An applet can include more than one mode. Siebel CRM associates each mode with a template. For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.
- **Format template.** A web template definition containing Siebel Web Format (SWF) content. Allows you to create custom HTML types, such as specialized controls, list items, and page items, by editing Siebel web templates. For more information, see *Configuring an HTML Control Type* and *Caution About Using Specialized Classes*.

Siebel CRM can contain other pages that do not contain any Siebel tags. For example, you might include an About This Application help page. This page is not a template.

Development Environment You Use to Create HTML Output

The following figure describes the development environment you can use to create HTML output. The Siebel Web Engine uses templates, repository definitions, and HTML to create HTML output.



Explanation of Callouts

As shown in this figure, you use the development environment to do the following:

1. Edit the HTML in an external HTML editor.
2. Use the Web Layout Editor in Siebel Tools to edit the layout of an applet and to map object definitions to the Siebel web template.

   If a Siebel client requests a view, then the Siebel Web Engine does the following:
3. Gets the object definition for the view and the object definition for each applet in that view from the Siebel runtime repository. Gets the data defined in the object definition from the data manager layer of the Application Object Manager (AOM).
4. Matches this data with the template that the view references and each applet in the view.
5. Uses the placeholders in the template to display this view. Defines where Siebel CRM places each user interface element in the object definition and how to format this element.

   If the user accesses the HTML file in a web browser, then Siebel CRM displays it as a web page. It uses the layout that the original template defines and it gets the data and controls.

Siebel Web Template Reuse

You can share a Siebel web template among many objects in the Siebel repository. A template includes placeholders that do not contain data, so Siebel CRM can map any number of repository objects to a placeholder. This configuration allows you to modify only one template to apply style or structural modifications to numerous user interface elements. A typical Siebel application contains approximately 5 to 50 templates. These templates form the basis for several hundred views and applets. For example, Siebel CRM can share a template that defines the layout and format of a predefined list applet among all the list applet definitions that reside in the Siebel repository.

If a placeholder is not mapped, then the Siebel Web Engine ignores it. It also ignores the HTML that exists between the Siebel tags that define the placeholder. For example, if the template contains the layout for a list applet that is 10 columns wide but only two of the columns are mapped, then the Siebel Web Engine ignores the other eight unmapped columns. This configuration improves efficiency and performance.

Flexibility of Use

Siebel CRM comes with many predefined applet web templates and view web templates that you can modify. To support your business processes, it might not be necessary for you to modify any of the applet web templates and view web templates. In some situations, especially with customer and partner applications, you can modify predefined templates to reflect the look and feel that your company requires. You can create entirely new templates.

A Siebel web template can include another Siebel web template. Siebel CRM uses this configuration to improve efficiency. For example, to separate how Siebel CRM handles an applet title from how it handles the applet body, you can create a template file that includes the title in the applet template. This configuration allows you to define an applet layout one time, and then combine it with multiple different title layouts.

A Siebel web template must use valid HTML. It is recommended that you do not add JavaScript beyond what the Siebel Web Engine already creates. If it is necessary to add JavaScript, it is recommended that you create a custom Presentation Model (PM) or Physical Renderer (PR). For more information, see *Configuring Siebel Open UI* .

You can use Siebel Tools or an external editor to modify a template. For more information, see *Using Siebel Tools* .

Support for Multiple Browser Types

The layout and style of HTML web pages is dynamic. This configuration allows Siebel CRM to simultaneously support multiple browser types and versions. A Siebel web template supports conditional branching. Siebel CRM evaluates conditions according to the results that a business service returns.

How Siebel CRM References Web Pages

This topic describes some of the properties of the application object that identify the template that Siebel CRM uses for a particular situation. For more information, see the following topics:

- *About Applications*
- *Creating and Deploying an Application*
- *Properties of an Application*

About the Container Page

The container page is the outermost template. It references view web templates and view web templates that reference applet templates. The container page contains markup language and Siebel Web Engine tag elements that define the web equivalent of the application window. You can examine this logic in the CCPageContainer_Common_ss web template. The Siebel Web Engine processes the container page template, view web templates, and applet web templates.

Container Page Elements

The container page includes the following elements:

- **Markups for the top of the container page.** Example markups include the corporate banner and Siebel tags that Siebel CRM uses for predefined queries. For example, in the Web Page Layout Editor you can view how the Queries menu label is the FavoritesLabel web page item. For more information, see *Guidelines for Modifying a Predefined Query*.
- **Screen tab bar.** Created beneath container page markups as a table. The Siebel Web Engine logic that is associated with the following tags loads the screen tab bar:
  -
  -
  -
- **View bar.** The Siebel Web Engine logic that is associated with the following tags loads the view bar:
  -
  -
  -

After Siebel CRM loads the container page and displays screen and view names, the screen and view names work like links in the following ways:

- If the user clicks a screen tab, then Siebel CRM uses the template for the default view for this screen to create and display the view.

- If the user clicks a view name in the view bar, then Siebel CRM loads the view web template that is defined in the object definition of the view.

The Siebel Web Engine does the following:

- Processes the set of tags in the view web template to include applets in the page.
- Uses the view object definition, view web templates, and applet web templates to identify the applets that Siebel CRM displays in a sector.
- Gets controls from the Siebel repository to resolve tag references to controls in each applet. Loads controls into the web page as defined in the applet web template child object of the applet. For more information, see *How Siebel CRM Uses HTML Frames in the Container Page*.

How Siebel CRM Uses HTML Frames in the Container Page

The container page can contain HTML frames that allow Siebel CRM to independently update and scroll different parts of a page. Example elements include toolbars, menus, the main content area, and so on. For more information, see the following.

You can group applets into separate frames in a view web template. It is recommended that you do not use this configuration except where independent refresh or independent scrolling is required.

Siebel CRM uses the tag and the tag to do the following:

- Create attributes for HTML frames.
- Allow the Siebel Web Engine to control how it targets and refreshes URLs.

OD Tag That Defines the Set of Frames in a Document

The tag defines the set of frames that the document contains. Similar to the HTML frameset tag, the Siebel Web Engine displays it as an HTML frameset tag. The body of this tag can only contain tags.

The tag uses the following format:

```
<div od-type="frameset" htmlAttr="xxx"> ...
<!--od section frameset close--> </div>
```

The tag includes the htmlAttr attribute. This attribute defines the attributes for the HTML frameset tag. For example, the following code supports a layout where the frames that belong to the frameset use 89 pixels, 25 pixels, and the remainder of the window:

```
htmlAttr="rows='89,25,*'"
```

OD Tag That Marks the Beginning and End of Content in a Frame

The tag marks the beginning and end of the contents that Siebel CRM places in a frame. The Siebel Web Engine displays this tag as an HTML frame tag, with the src attribute of the tag set to a Siebel Web Engine URL that gets the contents of the frame. You must place this tag in the body of the tag.

The tag uses the following format:

```
<div od-type="frame" type="xxx" name="yyy"> ....
<!--od section frame close--> </div>
```

The tag includes the following attribute:

```
type
```

The type attribute indicates the nature of the contents of the frame. The Siebel Web Engine uses this information to decide when to refresh the frame. It supports the following values for the type attribute:

- Siebel CRM uses the following values in a container page template:
  - **Toolbar.** Specifies that the frame contains the toolbar.
  - **Screenbar.** Specifies that the frame contains the primary tab bar.
  - **Viewbar.** Specifies links to views and categories of views.
  - **View.** Specifies that the frame contains the current view, that is, the content area.
  - **Page.** Specifies that the frame contains a web page. Siebel CRM does not refresh these frames after initial loading.
- **Applet.** In a view web template, specifies that the frame contains an applet.
- **Content.** Defines the content area and contains a view frame that displays the main view. To display an alternate view, it can contain one or more AltView frames. The search center is an example of an alternate view.
- **AltView.** Designates subframes to display one or more alternate views in the content frame in addition to the one in the view frame.
- **Name.** Used only if the type of the frame is page. In this situation, you can use this attribute to define a name for the frame. For other frame types, the Siebel Web Engine creates consistent names for the frame.

Nested Framesets

The Siebel Web Engine supports nested framesets. In this situation the tag contains a tag, and the type attribute of the outer tag is the following:

```
page
```

Example Frameset Code From a Container Page

The following code is from the Page Container web template:

```
<div od-type="frameset" htmlAttr="rows='60,21,25,*' border='0' frameborder='No'">
 <div od-type="frame" div od-type="page" htmlAttr="marginheight='0' marginwidth='0'noresize scrolling='No'">
 <div od-include="CCFrameBanner"/>
 <!--od section frame close-->
 </div>
 <div od-type="frame" div od-type="screenbar" htmlAttr="marginheight='0' marginwidth='0' noresize
 scrolling='No'">
 <div od-include="CCFrameScreenbar"/>
 <!--od section frame close-->
 </div>
 <div od-type="frame" div od-type="viewbar" htmlAttr="marginheight='0' marginwidth='0' noresize
 scrolling='No'">
 <div od-include="CCFrameViewbar"/>
 <!--od section frame close-->
 </div>
 <div od-type="frame" div od-type="view" htmlAttr="marginheight='0' marginwidth='0' noresize
 scrolling='Auto'">
 <div od-type="current-view"/>
 <!--od section frame close-->
 </div>
<!--od section frameset close-->
</div>
```

About View Web Templates

Siebel CRM uses a view web template to associate a view web template with a view. A view web template uses the tag to define placeholders for applets. You can use the Web Layout Editor to map an applet to each placeholder.

Example Code of a View Web Template

The following is code is from an example view web template:

```
<!-- Template Start: CCViewBasic -->
<!------------ Page Title ------------->
<title>
<div od-property="Title"/>
</title>
<!----------- Salutation applet and Search Applet, table 3.1 ------>
<table border="0" cellspacing="0" cellpadding="1" width="100%">
 <tr>
 <td width="66%"><div od-type="applet" id="101"/> </td>
 <td width="33%"><div od-type="applet" id="201"/> </td>
</tr>
</table>
<!-------- End Salutation applet and Search Applet, table 3.1 ------>
<!------------ Regular Applet(s) ---->
<div od-prefix od-id=[1:5] od-iterator="currentId">
 <div od-type="applet" id="od-type:currentId"/>
<!--od section iterator close-->
</div>
<!------------ Special Applet(s) ---->
<div od-prefix od-id=[11:3] od-iterator="currentId">
 <div od-type="applet" od-id="od-attr-currentId"/>
<!==od section iterator close-->
</div>
<!-- Template End: CCViewBasic -->
```

Applet ID Tags

Each tag is a placeholder that determines the location for an applet in the view web template. To display different views, you can map applets that currently exist in the view to placeholders in this same view web template. View web templates that come predefined with Siebel CRM include the following tags:

- **Tags with IDs of 101 and 201.** Displays the salutation and search applets that Siebel CRM displays at the start of the views.
- **Tags with IDs 1 through 10.** Displays the main applets in the view.
- **Tags with IDs that begin with 11.** Displays special applets that Siebel CRM displays at the lowest level of some views.

HTML Frames in a View Web Template

To display applets in a view, you can use HTML frames in view web templates and create a frame definition document. The Siebel Web Engine refreshes these frames only if one or more of the applets that the frame contains includes new data.

The following situations require HTML frames in the content area of a view web template:

- If a tree applet occupies a frame in the first section and the corresponding list applet occupies the frame in the section in an explorer view.
- If the user does a search. Siebel CRM requires a search frame and a results frame in the second portion of the content area.

Example Code for Using HTML Frames in a View Web Template

The following is an example of code that uses HTML frames in a view web template:

```
<!-- CCView_33_66_Frame start -->
<div od-type="frameset" htmlAttr="cols='33%,66%'' border='1' frameborder='Yes'">
<!-- Column 1 Applets -->
<div od-type="frame" type="Applet" htmlAttr="marginheight='0' margin width='0' scrolling='Auto'">
<div od-prefix od-id=[101:10] od-iterator="currentId">
 <div od-type="applet" od-id="od-attr-currentId" hintText="Applet" od-context="<parent>">
 <!--start applet-->
 <div od-property="FormattedHtml"/>
 <!--end applet-->
 <!--od section applet close-->
 </div>
<!--od section iterator close-->
</div>
<!--od section frame close-->
</div>
<!-- Column 2 Applets -->
<div od-type="frame" type="Applet" htmlAttr="marginheight='0' marginwidth='0' scrolling='Auto'">
<div od-prefix od-id=[201:10] od-iterator="currentId">
 <div od-type="applet" od-id="od-attr-currentId" hintText="Applet" od-context="parent">
 <!--start applet-->
 <div od-property="FormattedHtml"/>
 <!--end applet-->
 <!--od section applet close-->
 </div>
<!--od section iterator close-->
</div>
<!--od section frame close-->
</div>
<!--od section frameset close-->
</div>
<!-- CCView_33_66_Frame end --> </HTML>
```

About Applet Web Templates

This topic describes applet web templates. It includes the following information:

- *Overview of Applet Web Templates*

- *About Grid Form Applet Templates*
- *About Nongrid Form Applet Templates*
- *About List Applet Templates*
- *About Tree Applet Templates*
- *About Catalog List Applets and Rich List Templates*

Overview of Applet Web Templates

An *applet web template* is a type of Siebel web template that allows you to define multiple templates for a single applet, where each template file is associated with one or more modes. The applet web template is a child of the applet. For more information, see *About the Form Applet and List Applet* and *Adding a Web Template to an Applet*.

Properties of the Applet Web Template

The following table describes properties of the applet web template.

Property	Description
Type	Indicates the edit mode that the applet template supports, such as Edit or New. For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.
Web template	Provides the name of the web template used for that mode.

Properties of the Applet Web Template Item

The applet web template item defines the mappings that exist between controls and list columns to placeholders in the web template file. The applet web template item is a child of the parent applet web template. The following table describes properties of the applet web template item.

Property	Description
Name	Name of the applet web template item, which is typically the same as the Control property.
Control	Name of the control as Siebel CRM displays it in the Siebel client.
Item Identifier	A unique, numeric identifier for each control that Siebel Tools creates in the layout editor. Siebel CRM uses the value in the markup language tag that specifies the corresponding control in a template. It binds the control to a position on the page. Each applet web template item that comes predefined with Siebel CRM includes this unique number. If you use the Web Template Editor to add an object to an applet web template, then Siebel CRM automatically adds an applet web template item for this object and it sets the value for the item identifier to a unique value. If you modify the value of the Item Identifier property, then you must make sure the value you enter is unique among all objects that the Applet Web Template Items list displays. Siebel Tools does not automatically verify that the value you enter is unique. If every item identifier value in the Applet Web Template Items list is not unique, then the Siebel CRM run-time environment might not work correctly.

Property	Description
Type	Indicates the type of control that the applet web template item defines. You can choose one of the following values: • Control • List Item • Web Control

About Grid Form Applet Templates

A grid template simplifies the work of creating the layout of a form applet. Web Tools displays a grid layout template in a graphic interface that includes objects that you relocate from a palette to a work space. The grid layout applet web template, Siebel tag, and other features in the Applet Web Template Editor allow you to modify the layout of a form without directly modifying the underlying applet web template. For more information, see *Using Grid Layout for an Applet*.

You can do the following with an applet web template that uses a grid. You cannot do this work in an applet web template that does not use a grid:

- Use Web Tools to modify the layout of the form without having to directly modify the web template.
- Place labels and controls independently in the applet layout. Labels and controls are a single object in the Siebel repository that use one set of shared properties, but you can manipulate them as separate items in the Applet Web Template Editor.
- A template that uses a grid does not automatically compress empty space in a column.

A grid layout applet web templates uses the following Siebel tags:

```
<div od-type="form-applet-layout">
<!--od section form-applet-layout close--> </div>
```

These tags do not use placeholder tags. Instead, they provide a single container that contain all controls in the main body of a form applet. These tags allow you to use the Applet Web Template Editor to configure the layout of a form applet. You must use the Applet Web Template Editor to modify the layout of an applet that uses a grid applet web template.

Note: The feature described in this topic is available in Siebel CRM 18.10 Update and later releases.

About the Body, Header, and Footer

A grid layout applet web template includes a body, header, and footer:

- The tag defines the body. It contains no placeholder tags.
- The header and footer use placeholder tags for buttons, such as New and Save. You cannot use the grid layout features of the Applet Web Template Editor to edit the layout of the header or footer.

About Nongrid Form Applet Templates

Predefined applet web templates that do not use a grid use placeholder tags to define the layout of the applet. You can use the Applet Web Template Editor in Web Tools to map controls to any available placeholder. You cannot use the Applet Web Template Editor to modify the layout of the placeholders themselves. To modify the layout of the placeholders that Siebel CRM displays in these templates, you must modify the applet web template file.

Siebel CRM can display a form applet in any of the following modes:

- Base
- Edit
- New
- Query

Example Code of a Nongrid Form Applet Template

The following example code of a nongrid form applet template can run in Edit, New, and Query mode. An applet that runs in Base mode is similar except it does not contain the tag. For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.

```
<div od-type="control" id="1100">
 <div class=CmdTxt>
 <div od-property="FormattedHtml" hintText="Outside Applet Help Text"/>
 </div>
<!--od section control close-->
</div>
<table class="AppletStyle1" width="100%" align="center">
 <div od-type="form">
 <tr>
 <td colspan="2">
 <div od-include ="CCTitle"/>
 </td>
 </tr>
 <tr>
 <td>
 <div od-type="error">
 <div od-property="FormattedHtml"/>
 <!--od section error close-->
 </div>
 </td>
 </tr>
<div od-prefix od-id=[1301:10] od-iterator="currentId">
<div od-type="control" od-id="od-attr-currentId" hintMapType="FormItem">
 <tr valign="top">
 <td class="scLabelRight"> 
 <div od-property="RequiredIndicator" hintText="Required"/>
 <div od-property="DisplayName" hintText="Label"/>
 </td>
 <td class="scField">
 <div od-property="FormattedHtml" hintText="Field"/> 
 </td>
 </tr>
 <!--od section control close-->
 </div>
<!--od section iterator close-->
</div>
<!--od section form close-->
</div>
```

```
</table>
```

Tags Included in a Nongrid Form Applet Template

This topic describes the some of the tags that a nongrid form applet template includes.

OD Tag That Accepts User Input

The tag encloses a section of a page that accepts user input. It is similar to an HTML form tag. This tag includes the following attributes:

- **htmlAttr.** Must include valid attributes of the HTML form tag other than method, name, or action. Siebel CRM uses these attributes in the same way it uses the HTML form tag that it creates.
- **name.** Creates an HTML form with the defined name. If the name attribute is not defined, then Siebel CRM uses an internally created name.

OD Tag That Specifies Placeholders for Controls

The tag specifies placeholders for controls. This tag includes the following attributes:

- **od-id.** References the control for the placeholder.
- **od-property.** References the value of the control to display. This attribute includes the following values that are relevant for a form applet:
- **FormattedHTML.** Configures Siebel CRM to display the data value of the control.
- **DisplayName.** Corresponds to the Caption property.
- **RequiredIndicator.** Configures Siebel CRM to display HTML if the underlying business component field is required.

OD Tag That Handles Errors

For more information, see *Configuring How Siebel CRM Displays an Error That Occurs on the Siebel Server*.

About List Applet Templates

This topic describes the list applet template. For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.

Example Code of a List Applet Template

The following is example code of a list applet template:

```
<table width="100%" cellspacing="0" cellpadding="0" border="0" align="center">
<div od-type="form">
 ...
<div od-type="list">
<!-- List Header Section Start>
 <div od-type="control" id="147">
 <td class="Header" align="center">
 <div od-property="DisplayName"/>
 </td>
 <!--od section control close-->
 </div>
 <div od-type="select-row">
 <td width="42" align="center" class="Header"> 
```

```
 </td>
 <!--od section select-row close-->
 </div>
<div od-prefix od-id=[501:20] od-iterator="currentId">
 <div od-type="control" id="od-attr-currentId">
 <td align="od-context-TextAlignment" class="Header">
 <div od-property="ListHeader"/>
 </td>
 <!--od section control close-->
 </div>
<!--od section iterator->
</div>
 <div od-type="control" id="142">
 <td class="Header" align="center">
 <div od-property="DisplayName"/>
 </td>
 <!--od section control close-->
 </div>
<!-- List Header Section End>
<!------------ Loop for all 7 records, List Body ------------>
<div od-iterator="row" od-id="[0:7]">
<tr class="od-context-RowStyle">
 <div od-type="control" id="147">
 <td width="42" align="center" class="Row">
 <div od-property="FormattedHtml" hintMapType="Control"/>
 </td>
 <!--od section control close-->
 </div>
 <div od-type="select-row">
 <td width="42" align="center" class="Row">
 <div od-property="FormattedHtml"/>
 </td>
<!--od section select-row close-->
</div>
 <!-- ---------- List Field Values (501-520) ------------>
 <div od-prefix od-id=[501:40] od-iterator="currentId">
 <div od-type="control" od-id="od-attr-currentId">
 <td align="od-context-TextAlignment" class="Row">
 <div od-property="FormattedHtml">
 hintText="Field"/>
 </td>
 <!--od section control close-->
 </div>
 <!--od section iterator close-->
</div>
 <!-- ---------- Per-record Control Buttons ------------>
 <div od-type="control" id="142">
 <td align="center" class="Row">
 <div od-property="FormattedHtml" hintMapType="Control"/>
 </td>
 <!--od section control close-->
 </div>
 </tr>
 <!--od section iterator close-->
 </div>
 <!-- ---------- End Loop, List Body ------------>
 <!--od section list close-->
 </div>
...
 <!--od section form close-->
 </div>
 </table>
```

Tags That a List Applet Template Includes

This topic describes some of the tags included in a list applet template.

OD Tag That Encloses an Editable Section

The tag encloses an editable section. You use it for an editable list applet.

OD Tag That Encloses the List Header and Body

The tag encloses the section of the template that contains the list header and body. Siebel CRM replaces the section between the start and end of the tags with the specialized list control that supports some capabilities, such as resizing columns. For more information, see *Caution About Using Specialized Classes*.

OD Tag That Defines a Placeholder for a List Column

The tag defines a placeholder for a list column. It includes the Property attribute that specifies the property of the control to display. This attribute includes the following values that are relevant for a list applet:

- **FormattedHTML.** Configures Siebel CRM to display the data value of the control.
- **DisplayName.** Corresponds to the Caption property.

You can use some properties of a list column to control the attributes of an HTML element that the tag contains. For example, you can use the following code to set the align attribute of a TD tag to equal the Text Alignment property of the enclosing list column:

```
<td align="od-context-TextAlignment">
```

OD Tag That Repeats for Each List Row

The tag encloses the section of the template that Siebel CRM repeats for each list row.

About Tree Applet Templates

This topic describes the tree applet template. For more information, see *Configuring a Tree Applet*.

How Siebel CRM Builds and Displays a Tree Applet

To display a tree, Siebel CRM iterates through each item of the tree in a top-down, depth-first fashion, and displays one item at a time. Siebel CRM defines this behavior in the tag.

Siebel CRM uses the tag to indent each tree item. It uses this tag to do the following:

- Place the text in the correct indent level relative to the root.
- Display the expand icon, collapse icon, the text of the item, and the links.

Siebel CRM uses a series of GIF images to do the indentation or to insert white spaces if in text-only mode. Siebel CRM uses images to display the expand icon and collapse icon, or to display text if in text-only mode. The tag defines these objects. Siebel CRM displays, as part of the view, the list applet that is associated with the currently chosen tree node. For more information, see *Configuring Icons in a Tree Applet*.

Example Code of a Tree Applet Template

In this example, if the user clicks or expands a tree, then the tag provides a placeholder for a list applet that Siebel CRM displays. The applet that Siebel CRM displays depends on the node that is currently chosen.

The following is example code of a tree applet template:

```
<!--View with tree applet in the first section and list applet in the second-->
<table border="0" cellspacing="0" cellpadding="1" width="100%">
 <tr>
 <!-- Begin Tree Applet -->
 <td>
 <div od-type="applet" id="1" hintText="Tree Applet"/>
 </td>
 <!-- Begin List Applet -->
 <td>
 <div od-type="applet-tree-list"/>
 </td>
 </tr>
</table>
```

Example Code of a Tree Applet Template That Displays the Tree in a Single Column

In this example, Siebel CRM ignores the following items:

- For tree items, it ignores the tag that includes a DisplayName type.
- For tree nodes, it ignores the tag that includes a FieldValue type.

The following example code of a tree applet template displays the tree in a single column:

```
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0>
<TBODY>
<div od-iterator="nodeIterator">
<TR VALIGN=top>
<TD NOWRAP>
<div od-iterator="indentIterator">
<div od-type="indent-img"/>
<!--od section iterator close-->
</div>
<div od-type="node" type="DisplayName">
<div od-property="FormattedHtml"/>
<!--od section node close-->
</div>
<div od-type="node" type="FieldValue">
<div od-property="FormattedHtml"/>
<!--od section node close-->
</div>
</TD>
</TR>
<!--od section iterator close-->
</div>
</TBODY>
</TABLE>
```

Tags That Siebel CRM Includes in a Tree Applet Template

This topic describes some of the tags that Siebel CRM includes in a tree applet template.

OD Tag That Displays Tree Nodes and Field Values

Siebel CRM uses the <div od-node> tag to display tree nodes and field values. It iterates through each visible item that resides in the tree control in a top-down, depth-first fashion.

The attributes are optional. If the Count attribute is not defined, then the tag iterates through all nodes in the tree. The <div od-node> tag includes the following attributes:

- **Count.** Specifies the number of times the tag iterates through content. If you require a specific tree format, then you use can use this attribute.
- **StartValue.** The value that Siebel CRM uses to start the iteration. To start the iteration, the tag assigns this value to an internal iterator, and then increments it by one for each iteration.

OD Tag That Indents Tree Items

Siebel CRM uses the <div od-indent> to indent tree items. It iterates through each level of a tree item. The <div od-indent> tag does not include any attributes.

OD Tag That Provides a Placeholder for a GIF Image

Siebel CRM uses the as a placeholder for a GIF image that corresponds to the indent level for the tree item that the user chooses in the Siebel client. At each level, the Siebel Web Engine uses the tag to identify the GIF file. The GIF image can include an empty space or a vertical bar. The tag does not include any attributes.

OD Tag That Provides a Placeholder for a Tree Item

Siebel CRM uses the as a placeholder for tree item. A tree item can be a repository tree node or a field value. Note the following:

- If the tree item is a tree node, then Siebel CRM displays the display name.
- If the tree item is not a tree node, then Siebel CRM creates a field value.

The expand icon, collapse icon, and links are parts of a tree item. Depending on the configuration file settings, Siebel CRM displays the expand icon or collapse icon as a GIF image or text. It only displays the expand icon or collapse for a tree item that contain child items. The following links are associated with each item:

- A link for the expand icon and collapse icon that allows the user to expand or collapse the item
- A link for the item image that allows the user to choose the item or to navigate to next or previous workset

Choosing an item allows the user to access the list applet that is associated with the tree node. The <div od-type="node> tag must use the <div od-property="<propertyName>"> tag as a child tag.

The tag includes the type attribute. This attribute can include the following values:

- **DisplayName.** Displays the Display Name of the tree node.
- **FieldValue.** Displays a field value.

About Catalog List Applets and Rich List Templates

A catalog list applet or a rich list template supports a layout that is similar to a catalog. Siebel CRM can display a catalog in a view that references applets that maintain a parent and child relationship. You can display records from the parent applet and the child applet so that Siebel CRM interweaves them with each other.

Example of an Applet That Interweaves Records

An example of an applet that interweaves records is shown in the following image.



As shown in this image, Siebel CRM interweaves records in this example as follows:

- The parent applet provides data for the bullet items under Portable Music.
- The child applet provides data for values that Siebel CRM displays after Portable Music.

To create this layout, the parent and child applets are list applets. The parent applet is a root level applet. You can use more than one root level applet to display more than one set of parent-child relationships in a view.

The Position Property Defines Relationships Between Applets

The Position property of the view web template item defines the relationship that exists between the applets. It works similarly to the Position property of the tree node. The root level applets contain position values, such as 1, 2, and so on. You assign the immediate child applets of the applet with position 1 with position values 1.1, 1.2, and so on. You can define third level applets with position 1.1.1, 1.1.2, and so on. These third level applets are child applets of the applet with position 1.1. For more information, see *Defining the Position Property of a Tree Node*.

Tags That Define the Layout for Nonroot Applets

This topic describes tags that define the layout for nonroot applets layout. Siebel CRM only supports applets in the base mode in this layout. Note the following:

- If the Applet property in the view web template item references a root applet, then Siebel CRM maps this applet to a tag in the view web template.

- If the Applet property in the view web template item references a nonroot applet, then Siebel CRM does not assign an Id value to this applet. It does not define the layout of these nonroot applets in the view web template. It defines them in the applet template of the root level applets.

SWE Tag That Iterates Through Each Child Applet

The <div od-child> tag iterates through each of the child applets defined for the applet, as determined by the Item Identifier property of the view web template item of the view that the applet references. You can use this tag only in the base template of an applet. If the applet does not include any child applets, then Siebel CRM skips this tag.

The <div od-child> tag uses the following format:

```
<div od-child> ... <!--od section close--> </div>
```

For more information about the item identifier, see *Properties of the Applet Web Template Item*.

SWE Tag That Places the Child Applet in the Parent Applet

The tag places the child applet in the parent applet. Siebel CRM uses the base template of the child applet to display the child applet at the location where you place this tag.

The tag uses the following format:

```
<div od-type="child-applet"/>
```

Example of a Parent-Child Applet Relationship

This topic describes the following example parent-child applet relationship:

- The parent applet is Category Items List Applet
- The child applet is Sub Category Items List Applet

The following table describes the properties of the view web template item in the view that references these applets. For more information about the item identifier, see *Properties of the Applet Web Template Item*.

Item Identifier	Applet	Applet Mode	Position
100	Category Items List Applet	Base	1
101	Sub Category Items List Applet	Base	1.1

Code That Defines the Table for the Base Template

The following code defines the table for the base template of the Category Items List Applet:

```
<table>
<div od-iterator="row">
<tr>
<td>
<div od-type="control" id="5001"/> <!-- field value like "Small Business" -->
</td>
<td>
<div od-child>
<div od-type="child-applet"> <!-- Show the child applet -->
<!--od section child-applet close-->
</div>
</td>
```

```
</tr>
<!--od section iterator close-->
</div>
</table>
```

Code That Resides in the Base Template

The following code resides in the base template for the Sub Category Items List Applet:

```
<table><tr>
<div od-iterator="row">
<td>
<div od-type="control" id="5001"/> <!-- field value like "Desktop" -->
</td>
<!--od section iterator close-->
</div>
</tr></table>
```

Set the HTML Number of Rows property of the Sub Category Items List Applet to the number of values that Siebel CRM must display under each category value. To allow the user to drilldown from the category and subcategory values, you must configure the appropriate drilldown objects.

About Siebel Tags

This topic describes Siebel tags. It includes the following information:

- *Overview of How Siebel CRM Uses Siebel Tags*
- *About Singleton and Multipart Tags, and the This Tag*
- *About Iterator Tags*
- *About Search and Find Tags*
- *About Siebel Conditional Tags*

For more information, see *Siebel Tag*.

Overview of How Siebel CRM Uses Siebel Tags

Siebel CRM maps a Siebel object to an Id in a web template. The Siebel web templates do not include references to controls in the Siebel repository. Instead, they include placeholder tags that define layout and style. The following is an example of a Siebel tag that places a web page item in a web page:

```
<div od-type="control" id="1" property="FormattedHtml"/>
```

Other Siebel tags place other items in a web page, such as view bars, applets, or controls.

To process this tag and create the final HTML, the Siebel Web Engine does the following:

1. For the current web page, it examines the Siebel runtime repository for web page item whose Item Identifier property is equal to 1. This is the mapping between the template file object and the repository object. For more information about the item identifier, see *Properties of the Applet Web Template Item*.
2. To replace the placeholder in the template file, it displays the Formatted HTML representation of this repository object.

The following image shows how the mapping between controls and IDs work to display an image as a link to add a new contact. In this image:

- The SWT File for the Applet is as follows:

```
...
<div od-type="control" id id="1" property="FormattedHtml">
 <img src="images/new_contact.gif" width="14" height="14">
<!--od tag control close-->
</div>
...
```

- Control 1 in the Siebel Repository has the following properties:

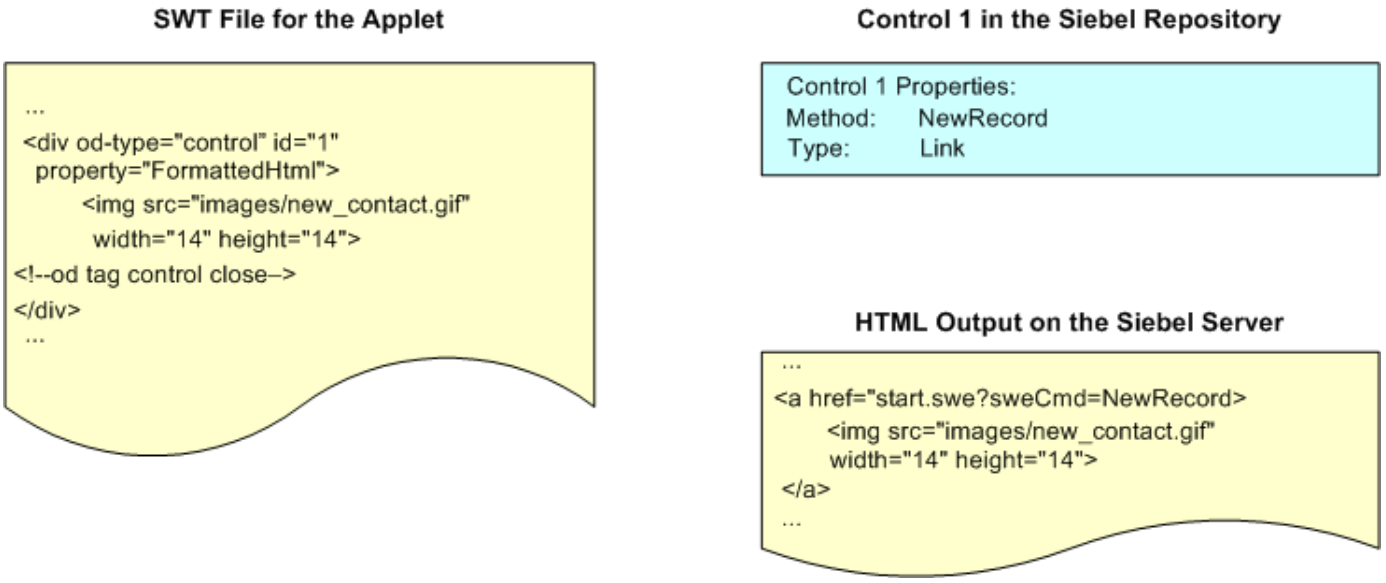
  Method: New Record

  Type: Link

- The HTML output on the Siebel Server is as follows:

```
...
<a href-"start.swe?sweCmd=NewRecord>
 <img src="images/new_contact.gif" width="14" height="14">
</a>
...
```

The HREF is likely different in your implementation. If you create the correct controls and template mappings, then the Siebel Web Engine creates a URL in the HREF that runs the NewRecord method in the correct context.



Siebel Tag Usage in an HTML Tag

You cannot nest a Siebel tag in an HTML tag. For example, the following code is not valid. It creates an error:

```
<img src="div od-type="control" id="1""/>
```

You cannot nest some Siebel tags. For example, the following is not valid. It creates an error:

```
<div od-type="control" id="1">
 <div od-type="control" id="2" property="formattedHTML"/>
 <div od-property="formattedHTML"/>
```

```
 <!--od section control close-->
 </div>
<!--od section control close-->
</div>
```

About Singleton and Multipart Tags, and the This Tag

This topic describes singleton and multipart tags, and the this tag.

About Singleton Tags and Multipart Tags

Singleton and multipart tags are part of the basic vocabulary of SGML (Standard Generalized Markup Language). This topic only describes them as they pertain to Siebel CRM, which uses singleton and multipart tags in the typical way.

A *singleton* element is a tag that includes a slash that indicates the end of the tag. It occurs in the same tag as the tag name. A singleton tag does not include child elements. The following is an example of a singleton tag:

```
<div od-type="pageitem" od-name="value"/>
```

The following is an example of a *multipart* tag. It does not include a slash at the end of the tag:

```
<div od-type="control" id="1" property="formattedHTML">
 ...HTML here...
<!--od section control close-->
</div>
```

About the This Tag

A *this* tag is a type of Siebel tag that you can use if you must use a multipart tag but reference the control that the Siebel Web Engine creates at a location other than at the beginning or end of the tag. The following code includes an example of a this tag:

```
<div od-type="control" id="1">
 ...HTML here...
 <div od-property="formattedHTML"/>
<!--od section control close-->
</div>
```

The `<div od-property>` tag is an alias for the nearest enclosing Siebel context. You can enclose the `<div od-type="xxx">` element to create this context. For example, Siebel CRM commonly includes the `<div od-property>` tag in a multipart `<div od-type="control">` element. In this situation, the `<div od-property>` tag is an alias for the control. You use it to display properties of the control. In some situations, the context is less direct. For example, if Siebel CRM includes a `<div od-property>` element in an applet template file, and if the `<div od-property>` tag is not in any `<div od-type="control">` tag, then it is an alias for the applet and you can use it to display properties of the applet.

About Iterator Tags

An *iterator* tag is a type of Siebel tag that defines the number of times the tag must iterate the contents of the iterator tag. For example, if you use the same HTML and Siebel tags with controls or page items that contain different values for the Id parameter, then you can use the following <div od-prefix> tag to reduce the size of the template files:

```
<div od-prefix=<id attr> od-iterator="<iteratorName>" od-id=[startValue:count]>
```

You can use the following iterator tags:

- <div od-prefix>
-
-
-
-
-

The <div od-prefix> tag includes the following attributes:

- **count.** Specifies the number of times the <div od-prefix> tag must iterate the contents of the <div od-prefix> tag.
- **startValue.** The value assigned to the iterator at the start of the iteration. The tag assigns this value to the iterator to start the iteration. The tag increments the value by one for each iteration.
- **iteratorName.** The name of the iterator. You can use this name to get the value of the iterator during the iteration. You use the following format: <div od-iterator="<tag>Iterator">.

Determining the Current Value of the Iterator

The name defined in the iteratorName attribute determines the current value of the iterator. The section that the <div od-iterator="<tag>Iterator"> tag encloses includes this name.

To determine the current value of the iterator

- For example, if you set the value of the iteratorName attribute to CurrentID, then you can use the following format to get the value of the iterator:

```
<od-attr-currentid>
```

You can use the <od-attr-currentid>+x tag to reference a value that is an increment over the current value. The following code is an example of this usage:

```
<div od-prefix od-id=[2301:50] od-iterator="currentId">
 <div od-type="control" id="od-attr-currentId">
 .
 .
 <!--od section control close-->
 </div>
 <div od-type="control" id="od-attr-currentId+100"/>
<!--od section iterator close-->
</div>
```

About Search and Find Tags

Siebel CRM merges search, find, and query in a unified search model. This configuration provides users with multiple ways to locate records. You can configure a template to display search and query features from an application menu and query features from an applet menu. You can use Siebel tags to display the Search and Find applet, and the Results applet.

Search and Find Applet Tags

You can use the following search and find tags to configure how Siebel CRM displays basic search, advanced search, or find in an applet:

-
-
-
-

Example Code for Search and Find Applet Tags

The following code is an example of using search and find applet tags:

```
<div od-type="srchCategoryList">
<div od-type="srchCategory">
<td><div od-type="srchCategoryText"/></td>
<td><div od-type="srchCategoryControl"/></td>
<!--od section srchCategory close-->
</div>
<!--od section srchCategoryList close-->
</div>
```

Search Result Applet Tags

You can use the following search results tags to display search and find results in a list applet:

-
-
- <div od-property>

Siebel CRM displays search results tags in the Search_ListBodySearchResults web template.

Example Code for Search Result Applet Tags

The following code includes an example that uses search result applet tags:

```
<div od-type="srchResultFieldList">
 <div od-type="srchResultField><td align="od-context-TextAlignment"
 class="Row"><div od-property="FormattedHtml"/> </td>
 <!--od section srchResultField close-->
 </div>
<!--od section srchResultFieldList close-->
</div>
```

Summary of Search, Find, and Search Result Tags

The following table describes search, find, and search result tags.

Tag Name	Description
	Search tag that is an iterator that encloses all the search categories that Siebel CRM must display. It creates context and encloses the following tags: • •

Tag Name	Description
	• It uses the following format: `<div od-type="srchCategoryList"> ...` `<!--od section searchCategoryList close--> </div>`
`<div od-type="srchCategory">`	Search tag that represents a search category object. It encloses the following tags: • • It uses the following format: `<div od-type="srchCategory"> ...` `<!--od section srchCategory close--> </div>`
`<div od-type="srchCategoryControl">`	Search tag that displays the control of the search category. It is a check box in advanced search. It must be called in the context of a srchCategory tag. It uses the following format: `<div od-type="srchCategoryControl"/>`
`<div od-type="srchCategoryText">`	Search tag that displays the display name of the search category. It must be called in the context of the srchCategory tag. It uses the following format: `<div od-type="srchCategoryText"/>`
`<div od-type="srchResultFieldList">`	Search result tag that is an iterator. It encloses all the search result fields that the search engine **object** contains. Siebel CRM creates result fields dynamically in the business component, and then displays them in the applet. This tag creates a context and encloses the following tags: • • <div od-property> It uses the following format: `<div od-type="srchResultFieldList"> ...` `<!--od section srchResultFieldList close--> </div>`
`<div od-type="srchResultField">`	Search result tag that represents a result field object. Siebel CRM must call it in the context of the srchResultFieldList tag. It encloses the <div od-property="<xxx>"/> tag. It uses the following format: `<div od-type="srchResultField"> ...` `<!--od section srchResultField close--> </div>`
`<div od-property="<>">`	A search result tag. Depending on the value of the property attribute, it does one of the following:

Tag Name	Description
	• **Property is TextAlignment.** Gets the text alignment property for the result field from the following object: Search Definition - Custom result Field • **Property is FormattedHtml.** Gets the value for the current result field from the results that Siebel CRM gets when it runs the search on the search adapter. It uses the following format: `<div od-property/>`

About Siebel Conditional Tags

This topic describes the following Siebel Web Engine conditional tags. For more information, see *Siebel Developer's Reference* .

- *If Conditional Tag*
- *Switch, Case, and Default Conditional Tags*
- *Variable Conditional Tag*

If Conditional Tag

The `<div od-if="<>">` tag provides a simple conditional branching capability. It uses the following format:

```
<div od-if="<>"> ... <!--od <tagName> close--> </div>
```

The `<div od-if="<>">` tag includes the Condition attribute. Siebel CRM does the following:

- If the condition is TRUE, then Siebel CRM processes the body of the `<div od-if="<>">` tag.
- If the condition is FALSE, then Siebel CRM skips the body of the `<div od-if="<>">` tag.

The `<div od-if="<>">` tag does not provide an else capability. To implement an else condition, you can use some combination of the <div od-switch>, <div od-case="<>">, and <div od-default> tags.

Switch, Case, and Default Conditional Tags

If used together, then the following tags provide a conditional branching capability that is similar to the switch, case, and default statements in JavaScript:

- <div od-switch>
- <div od-case="<>">
- <div od-default>

The <div od-switch> tag is a container tag for the <div od-case="<>"> and <div od-default> tags.

Format for the Switch, Case, and Default Conditional Tags

The <div od-switch>, <div od-case="<>">, and <div od-default> tags use the following format:

```
<div od-switch>
<div od-case="xxx">
...
<!--od section xxx close-->
```

```
</div>
<div od-case="yyy">
...
<!--od section yyy close-->
</div>
<div od-default>
...
<!--od section default close-->
</div>
<!--od section switch close-->
</div>
```

Attributes for the Switch, Case, and Default Conditional Tags

The <div od-case="<>"> tag includes the Condition attribute. The <div od-switch> and <div od-default> tags include no attributes. To process these tags, Siebel CRM does the following:

- Ignores any tags that exist in the body of the <div od-switch> tag that are not the <div od-case="<>"> tag or <div od-default> tag.
- Examines the <div od-case="<>"> tags, starting with the first <div od-case="<>"> tag, and then does one of the following:
  - If any of the <div od-case="<>"> tags satisfy the condition, then Siebel CRM skips any other <div od-case="<>"> tags and <div od-default> tags, and then processes the body of the <div od-case="<>"> tag that satisfies the condition.
  - If none of the <div od-case="<>"> tags satisfy their conditions, then Siebel CRM processes the body of the <div od-default> tag. You must make sure that the body of a <div od-switch> tag contains only a <div od-default> tag.

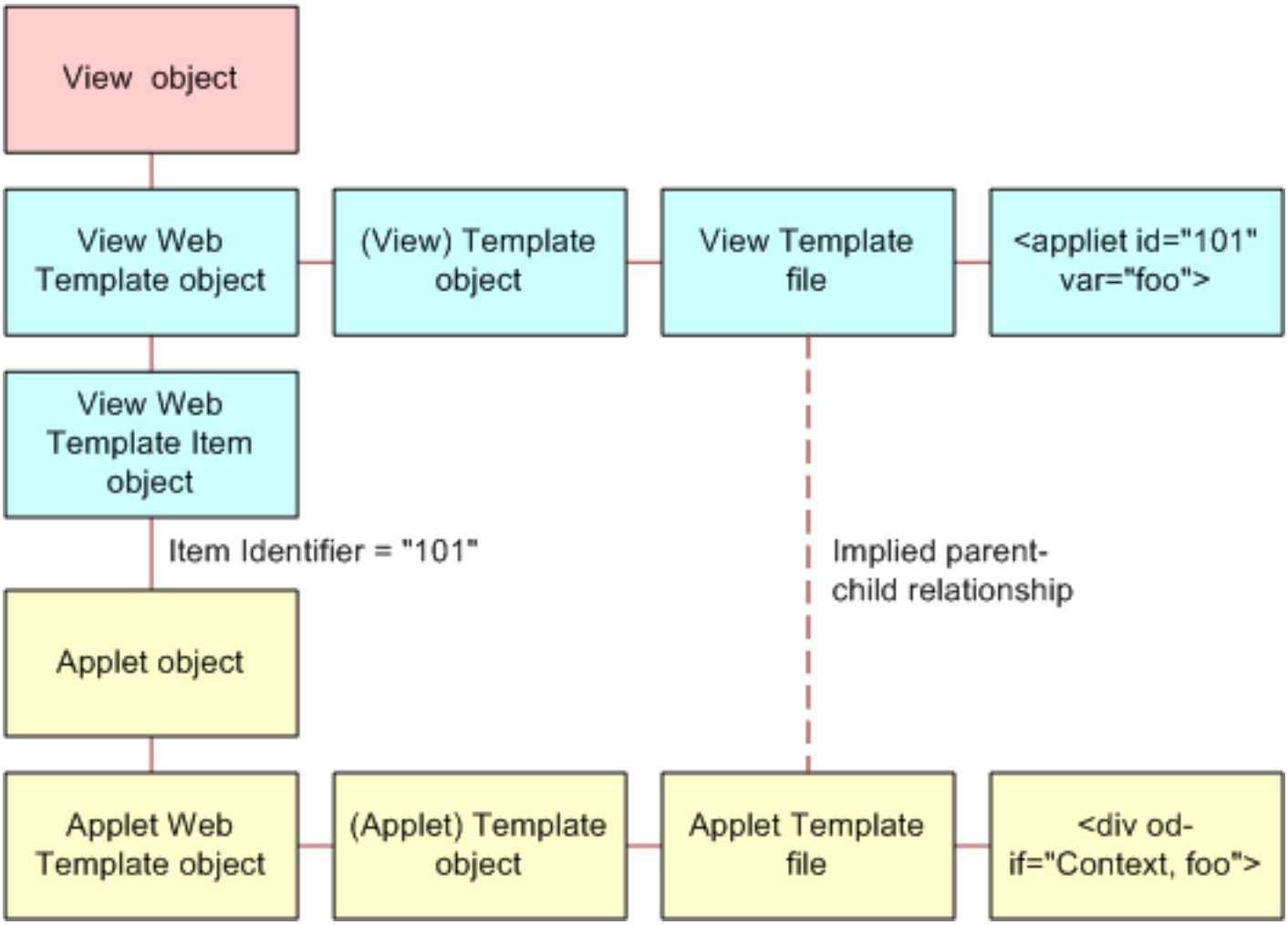
Variable Conditional Tag

An applet template includes the <div od-if="Context, <object>"> tag. It conditionally express the body of the <div od-if="Context, <object>"> tag as determined by a variable that is set in a parent view web template. For the purposes of the <div od-if="Context, <object>"> tag, if an applet is associated with a view, then the web template in the applet acts as a child of the view web template.

The applet placeholder in the view web template must define a variable that the <div od-if="Context, <object>"> tag in the child applet template can evaluate.

The expression in the <div od-if="Context, <object>"> tag returns a value of true or false depending on if the variable it evaluates is a property of the tag in the corresponding view web template. You can use this configuration to conditionally display parts of an applet depending on the position of the part in a view.

The following image shows the object relationships that Siebel CRM uses with the variable conditional tag.



Example Code That Uses the Variable Conditional Tag

As show in this image:

- A view uses a template that contains the following tags

```
<div od-type="applet" hintMapType="Applet" id="1"
 property="FormattedHtml" hintText="Applet" var="Parent"/>
<div od-type="applet" hintMapType="Applet" id="2"
 property="FormattedHtml" hintText="Applet" var="Child"/>
```

- The view object references an applet through a view web template item. The template for this applet includes the following tags:

```
<div od-if="Context, Parent">
 <td valign="middle" nowrap>
 <div od-type="menu" type="Button" bitmap="MenuBttn" width="38" height="15" bgcolor="gray"
 fgcolor="blue"/>
 </td>
<!--od section Parent close-->
</div>
<div od-if="Context, Child">
 <td valign="middle" nowrap>
```

```
 <div od-type="menu" type="Button" bitmap="MenuBttn" width="38" height="15" bgcolor="gray"
 fgcolor="red"/>
 </td>
<!--od section Child close-->
</div>
```

- If you move the applet into the placeholder in the view web template, and if the applet Id for this placeholder is:
  - **1.** The first condition returns TRUE and the second condition returns FALSE. This occurs because the var property of the placeholder that contains an Id of 1 is set to Parent. As a result, Siebel CRM displays the button menu with a "blue" foreground.
  - **2.** Siebel CRM displays the button menu with a "red" foreground.

Guidelines for Configuring Siebel Web Templates and Siebel Tags

This topic describes guidelines for configuring Siebel web templates and Siebel tags.

Guidelines for Using Modes with Web Templates

In many situations, it is not necessary to use Base mode forms that are read-only. You can use persistently editable forms because work often includes data editing and input. This type of form improves usability because the user can enter data without having to click an edit button, and then wait for Siebel CRM to display the form in edit mode.

If an applet is in Edit mode in a view, as defined by the applet mode property of the view web template item, then Siebel CRM never displays this applet in Base mode. If the user updates the field values in this applet, and then commits the modification, then Siebel CRM continues to display the applet in this mode after Siebel CRM writes the modifications to the Siebel database.

To display an applet in Query or New mode, you can call a method, such as NewQuery or NewRecord, on an applet that Siebel CRM displays in Edit mode. After Siebel CRM runs the query or writes the new record, Siebel CRM displays the applet in Edit mode.

For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.

Guidelines for Using HTML Frames in a View Web Template

If you use HTML frames in a view web template, then use the following guidelines:

- You can use frames in a view web template only if Siebel CRM uses frames in the container page, and only if a separate frame exists in the container page for the view.
- If you place an applet in a frame, then you must make sure that Siebel CRM maps at least one tag in the frame to an applet in the Siebel repository. Otherwise, empty frames will occur.
- If a block contains a tag, then set the type attribute of the tag to Applet.
- Do not group applets into separate frames in a view web template unless you require independent refresh or independent scrolling.

Guidelines for Creating HTML Frames in a Container Page

Siebel CRM uses the container page template to create the frame definition document for the Siebel application. If you define HTML frames in a container page, then use the following guidelines:

- Use the <div od-include> tag to define the contents of a frame. You can place the contents directly into the body of the tag.
- Make sure the contents of the tag constitutes a complete HTML document. The contents must contain the required HTML document structure tags, such as html, head, body, and so on. This requirement applies to view web templates.
- If the type is view, then make sure the contents of the tag contains only the tag.

Guidelines for Using Cascading Style Sheets

A cascading style defines qualities of user interface elements, such as color schemes and fonts. The following examples describe how you can use a cascading style sheet to modify the look and feel of the Siebel client:

- Display text in the font of your choice.
- Define the size of text in points, pixels, and other units.
- Configure color for images or background color.

You can configure Siebel web templates to use format tags, but if you store style information in cascading style sheets rather than in Siebel web templates, then you can realize the following benefits:

- Increase the modularity of a Siebel application.
- Increase consistency of a Siebel application.
- Simplify modification and reuse of Siebel web templates.

Siebel CRM displays style information that it stores in a cascading style sheet slightly differently in different browsers, so you must test your configuration in all browsers that your users use.

Siebel CRM locates cascading style sheet files in the following directories:

- The Siebel Application Interface installation directory:

```
SIEBEL_ROOT\applicationcontainer_external\siebelwebroot\FILES
```

- The Siebel client installation directory:

```
SIEBEL_CLIENT_ROOT\PUBLIC\FILES
```

- The Siebel Tools installation directory:

```
SIEBEL_TOOLS_ROOT\PUBLIC\FILES
```

If you apply a patch, then Siebel CRM might overwrite the CSS files. If this happens, then you must manually reenter the modifications you made to the cascading style sheets.

For more information, see *Cascading Style Sheet* and *Siebel Developer's Reference* .

Guidelines for Modifying a Predefined Query

If you modify a predefined query, then use the following guidelines:

> **Note:** Predefined Queries can be Workspace enabled in your development environment. If they have been Workspace enabled in your development environment and you are working in that environment, then you can only modify them in an editable Workspace. You do not need an editable Workspace to create and edit Predefined Queries in your Production environment.

- The tag defines the predefined query bar. It includes no parameters and can be located anywhere in Siebel CRM. The user chooses the query that Siebel CRM runs. You must explicitly define the following:
  - The tag
  - The Favorites label that Siebel CRM displays before the tag.
- To allow translation in a localized or multilingual deployment, it is recommend that you define the favorites label as a control and not as HTML text.
- If Siebel CRM does not use HTML frames, then it is not necessary to place the tag in the view frame.
- If Siebel CRM uses HTML frames, then you must place the tag in the view frame or the view bar frame.

Consider the following requirements:

- You cannot place the predefined query bar in the view frame.
- You can align the predefined query bar.

Query Management Commands

A user can add a named query in the combo box. Siebel CRM does this through the query management commands that are available as invoke method calls through the class of the base applet. Siebel CRM makes these queries available to the user as menu items or toolbar buttons.

The following table describes the query management commands that you can use.

Command	Method	Description
New	SWEMthdNewQueryE	Places the applet in new query mode.
Refine	SWEMthdRefineQueryE	Places the applet in query refinement mode.
Save	SWEMthdSaveQueryE	Uses the current name of the query to save the current query.
Save As	SWEMthdSaveQueryAsE	Displays a dialog box that allows the user to save the current query that uses a name that the user defines.
Delete	SWEMthdDeleteQueryE	Displays a dialog box that allows the user to delete one of the queries.

Command	Method	Description

It is strongly recommended that you do not define an Edit button for the predefined query feature. The Edit button must call the Refine method. A problem might occur if an Edit button exists in a multiview environment and Siebel CRM cannot determine the active view.

9 Configuring a Siebel Application

Configuring a Siebel Application

This chapter describes an overview of configuring a Siebel application. It includes the following topics:

- *About Configuring a Siebel Application*
- *Roadmap for Configuring a Siebel Application*
- *Developing an Implementation Plan*
- *Using Development Tools and Setting Up the Development Environment*

About Configuring a Siebel Application

Configuration is the process of modifying a predefined Siebel application to meet business requirements. This process can range from making a minor modification, such as adding text box controls and their underlying fields, to creating a new user interface or creating new business entities.

Siebel Tools is an integrated development environment that allows you to reconfigure and configure Siebel CRM. It is a software configuration toolset rather than a programming language. This toolset allows you to create and modify object definitions and their properties so that you can develop and configure Siebel CRM. You can also modify aspects of the Siebel Open UI application user interface outside of Siebel Tools, such as by using methods described in *Configuring Siebel Open UI* .

Siebel CRM is built on object definitions that Siebel CRM runs when the user uses the Siebel client. You can use Siebel Tools to modify these object definitions. To create a completely new module, you can create new object definitions or modify predefined definitions. It is not necessary for you to write C++ program code, but you can write Siebel Visual Basic (Siebel VB), Siebel eScript, or browser script to supplement the programmatic logic of Siebel CRM. Note that Siebel Visual Basic and Siebel eScript run on the Siebel Server. Browser script runs on the client.

Note: Oracle recommends that all new client-side browser scripts should be placed in the custom Presentation Model (PM) or Physical Renderers (PR). For more information, see *Configuring Siebel Open UI* .

Usage and Configuration of Nonlicensed Objects

The licensing agreement between Oracle and Oracle customers is such that customers are only entitled to use and configure Siebel objects that belong to modules they have purchased. Example objects include business components and tables. If Siebel CRM does not display a Siebel object in the licensed user interface through views that Siebel CRM displays according to your license key, then you are not entitled to use that object in a custom configuration. To display these tables, you can define new tables, new business components, and new user interface objects.

Roadmap for Configuring a Siebel Application

To configure a Siebel application, you perform the following processes and tasks:

1. *Developing an Implementation Plan*
2. *Using Development Tools and Setting Up the Development Environment*
3. *Process of Determining Whether You Can Reuse a Predefined Object*
4. Optional. *Process of Creating and Binding an Entity Relationship Diagram*
5. *Configuring Tables*
6. *Configuring Views, Screens, and Applications*
7. Configure various aspects of your Siebel application.

   You can configure some aspects of your Siebel application. For more information, see the relevant chapter in this book. For example, see the following chapters:
   - *Configuring Special-Purpose Applets*
   - *Configuring Siebel Web Templates*
   - *Localizing Siebel CRM*
   - *Improving the Performance of Siebel CRM*

This roadmap provides a typical guideline that describes how to configure a Siebel application. The actual tasks you perform and the sequence that you use to perform them varies significantly depending on your implementation requirements. For more information, see *Siebel Deployment Planning Guide* and *Developing and Deploying Siebel Business Applications* .

Developing an Implementation Plan

This topic includes the following information:

- *Developing a Configuration Strategy*
- *Developing a Plan to Control File Versions for the Physical User Interface Layer*

This task is a step in *Roadmap for Configuring a Siebel Application*.

To develop an implementation plan

1. Perform a thorough business analysis that details the needs of your organization and users.
2. Get approval and commitments for time and resource from the relevant organizations.
   - Determine whether a predefined Siebel application can or cannot meet the needs of your users.
   - If a predefined Siebel application cannot meet the needs of your users, determine what business needs require modifications to the Siebel application.
   - Determine how you can assure success with your configured application.
3. Write design documents that include the following items:
   - The requirements that the configured application satisfies.

- An entity relationship diagram (ERD) or text that describes the entity relationships. For more information, see *Using the Entity Relationship Designer*.
- The names and descriptions of the business objects and business components that Siebel CRM requires, and how they relate to one another.
- Screen flow diagrams and a list of fields that Siebel CRM displays on each applet.
- (Conditional) How your implementation will use various Siebel technologies. For more information, see *About Siebel Technologies That Configure Siebel CRM Behavior*.
- A description of your development environment and process. For example:
  - Describe how the work is divided among participating developers.
  - Describe naming formats the development team must use. For more information, see *Guidelines for Naming an Object*.
  - Describe how file versions for the physical user interface layer are controlled. For more information, see *Developing a Plan to Control File Versions for the Physical User Interface Layer*.
  - Describe how your organization will test and deploy Siebel CRM to users.
- The complete stepwise procedures your development and test team must follow to complete Siebel CRM configuration.

For more information, see *Developing a Configuration Strategy*.

4. Make sure the participating organizations and users review and approve the design.

Developing a Configuration Strategy

The major goal of configuring Siebel CRM is to develop an application that meets the look, feel, and requirements of your organization and your users, and that is easy to maintain and upgrade.

To develop a configuration strategy

1. Use the following guidelines in the plans for your configuration project:
   - Make as few modifications as possible.
   - Use predefined Siebel application functionality. Create a new object only if you cannot modify a predefined object to meet your requirements. If you follow this principle, then your Siebel application is much easier to maintain and upgrade to future Siebel product releases. For more information, see *Reusing Predefined Objects*.
   - Standardize configuration development.
   - Achieve acceptable system performance. For more information, see *Siebel Performance Tuning Guide* .
   - Build a consistent and intuitive Siebel client. For example, if you create a new form applet, then make sure it uses the same look and feel as other form applets in your Siebel application.
2. Plan your design starting at the top and working downward:
   a. Design the user interface for the Siebel client.
   b. Design the underlying business logic.
   c. Design the data objects layer that is necessary to support your configuration.
3. Develop a plan to configure Siebel CRM starting at the lower level and working upward:
   a. Modify objects at the data layer.
   b. Modify objects at the business object layer.
   c. Modify objects at the user interface layer.

This configuration helps you to make sure the correct values for all required object properties are available as options. For more information, see *About the Siebel Object Architecture*.

4. Use one of the following configurations to structure the development work:
    - Assign a development role to a single developer or group. This configuration allows different groups to work in parallel.

      For example, one group or an individual person can develop a web page and the logical business object definitions and data object definitions that are required to support the page.
    - Assign a single developer or group to one architectural layer.

      This configuration takes advantage of the expertise of developers. For example:

      - The RDBMS specialist defines extensions in the data objects layer.
      - The system architect defines the business object layer.
      - The user interface developer defines the user interface objects layer.

      This configuration requires each group to complete some work before another group begins their work.

Developing a Plan to Control File Versions for the Physical User Interface Layer

This topic describes how to manage modifications that your development team makes to the physical user interface. You do not configure the physical user interface layer in Siebel Tools, so you cannot use the Siebel Tools check out and check in feature to manage web templates, JavaScript files, or style sheets. These files are part of the physical user interface layer. If multiple developers simultaneously modify these files, then follow the recommendations that this topic describes. For more information, see *Overview of the Physical User Interface Layer*.

The description in this topic is appropriate for most projects. For more information, see *Getting Help From Oracle*.

To develop a plan to control file versions for the physical user interface layer

- Assign a single developer or group to manage web templates, JavaScript files, and style sheets. Make sure all modifications are made by this individual or group, who is solely responsible for releasing amended files to the Siebel Application Interface.
- Use version control software to manage modifications to web templates, JavaScript files, and style sheets. ClearCase is an example of source control software. This configuration makes sure that only a single individual amends these files at any time. It provides an audit trail of modifications.
- If source control software is not available, then use manual controls that allow a structured release. Assign an individual or group that is responsible for all amendments to physical user interface files and their subsequent release.
- Use a separate directory structure for each release that includes subfolders for the various objects that are released. Copy all amended physical user interface files that are included in the release to the appropriate subfolder.

  The date on which a file is amended can identify the web templates or JavaScript library files that you must release to users. It is necessary to use central release folders or to copy modified or new objects to these folders. A web template is an example of a modified or new object.

Using Development Tools and Setting Up the Development Environment

This task is a step in *Roadmap for Configuring a Siebel Application*.

This topic summarizes the development process for Siebel CRM and provides information about setting up the development environment. It includes the following information:

- *Overview of the Development Process*
- *Guidelines for Developing a Siebel Application*
- *Setting Up the Configuration File for Siebel Tools*
- *Displaying Object Types You Use to Configure Siebel CRM*
- *Creating Scripts to Configure Siebel CRM*

Overview of the Development Process

This topic describes a summary of work you perform to develop a Siebel application. Developing a Siebel application is not necessarily a serial process. During some phases, it makes sense for you to configure multiple sections of Siebel CRM concurrently. Some tasks are iterative, such as testing and debugging, so it is likely that you will modify the simplified and linear task that this topic describes so that it meets the development requirements for your team.

Most of the topics in *Using Development Tools and Setting Up the Development Environment* describe tasks that you can perform to set up a development environment. See also *Using Siebel Tools* .

To develop a Siebel application, you do the following:

1. Set up your development environment.
    a. Install Siebel Tools.
    b. In Siebel Tools or Web Tools, create your developer workspace.
2. Develop the Siebel application:
    a. Use Siebel Tools to modify or create the following object definitions:
        - Data objects, such as tables, columns, indexes, and so on.
        - Business components and business objects.
        - User interface objects. For example, applets, views, and screens.
    b. Modify web template files.
    c. Inspect or deliver your Workspace and perform unit testing.

    For more information, see *Using Siebel Tools* .
3. Optional. Use the tools that are available to you in the Siebel development environment to define whatever assignment and workflow rules are required.

    Example tools include Siebel Assignment Manager and Siebel Workflow.
4. Optional. Use Siebel Visual Basic or Siebel eScript to configure the functionality of Siebel CRM.
5. Optional. Localize Siebel CRM if the Siebel client must display content in two or more languages. For more information, see *Localizing Siebel CRM*.

6. Perform system and performance testing of your Siebel application.
7. Iterate through the development steps until your design is fully implemented, and until Siebel CRM runs smoothly and meets your performance objectives.
8. Deploy Siebel CRM to your users and train your users on how to use Siebel CRM.

Guidelines for Developing a Siebel Application

This topic describes some guidelines for developing a Siebel application. For more information about some of these points, see *Localizing Siebel CRM*.

Guidelines for Naming an Object

If you name an object, then use the following guidelines:

- Never include a question mark at the end of a field name or user interface label.
- Use an object name that is meaningful and descriptive. For example, Account Detail Applet With Agreement, instead of Account Detail Applet 2.
- Do not use a license key option as an object name. For example, do not use Product Forecasting. This configuration can cause Siebel CRM to not display a user interface object, such as a view.
- Prefix the name for each custom object that you create with your company name. For example, ABC Product Forecasting View. This configuration distinguishes your custom object from a predefined object.
- Be careful with spelling, spacing, and capitalization when you name an object. Typically, the logical name of an object in the Siebel repository uses complete words, mixed casing, and a space between words. A physical database object might use an abbreviation, uppercase, and an underscore. For example, the Service Request business component references the S_SRV_REQ database table.
- Do not use a reserved SQL word in an object name. For example SELECT, COUNT, FROM, WHERE, and UNION. This configuration can cause unpredictable application behavior.
- Avoid modifying the name of an object. It is time consuming to modify the name of an object when Siebel CRM references it throughout the Siebel repository. If you must modify the name of an object, then use the Find in Repository feature from the Tools menu in Siebel Tools to find all of the references.
- Siebel Tools uses English (ENU). It does not support objects from other character sets, such as ASCII (American Standard Code for Information Interchange) codes or accented characters. If you attempt to use objects from another character set, then Siebel Tools displays a unique constraint error.
- Do not use the following characters in Siebel object names:
  - backslash (\)
  - quotation mark (")
  - slash (/)
  - less-than symbol (<)
  - colon (:)
  - greater-than symbol (>)
  - asterisk (*)
  - vertical bar (|)
  - question mark (?)
  - tab characters

> **Note:** When naming objects in the Repository, only use elements of the ASCII character set. When pasting names or strings from programs such as Microsoft Word, you will get Word's equivalent of some characters such as " becomes “, and - becomes – in Word. When you paste those into the Repository it becomes an invalid name.

If you are not sure how to name an object, then you can use the predefined objects in the Siebel repository as a guide. Examine the predefined objects and conform to the naming formats that they use. For example, to create a new Association applet, use the *business component name* Assoc Applet naming format.

For more information, see the following topics:

- *Naming Format for a Siebel Table*
- *Guidelines for Naming a Business Component*
- *Guidelines for Naming an Applet*
- *Guidelines for Creating a Join*
- *Guidelines for Creating a Business Object*
- *Guidelines for Creating an Applet*
- *Guidelines for Naming a View*
- *How Siebel CRM References Web Pages*
- *Reusing Predefined Objects*

Guidelines for Setting the Upgrade Ancestor Property

Upgrade Ancestor is a property that allows a copied object to include properties of the original object that Siebel CRM uses to define the copy. During an upgrade, Siebel CRM modifies the original object and the copied object. You can use the following object types with the Upgrade Ancestor property:

- Applets
- Business Components
- Reports
- Integration Objects

For example, assume you create a copy of the Account List applet, name it the Premium Account List Applet, and then set the Upgrade Ancestor property. The new applet might differ from the original applet because the new applet includes a search specification that Siebel CRM only displays in accounts that are considered premium accounts. In a subsequent release, Oracle might add a new predefined list column to the Account List applet. During an application upgrade, your Account List applet and the Premium Account List Applet retain the configuration modifications you made. These applets receive the new predefined list column added in the new version.

Use caution if you copy an object. For more information, see *Guidelines for Reusing a Predefined Object*.

Note the following factors if you use the Upgrade Ancestor property:

- If you copy an object, then Siebel Tools does not automatically define the Upgrade Ancestor property. You must define it manually.
- Creating a new object without defining the Upgrade Ancestor property could add to your upgrade effort because Siebel CRM does not upgrade a custom object. Siebel CRM copies it to the new repository, but without modifications.
- Creating a new copy of a business component or applet can result in a redundant configuration.

For more information, see *Siebel Database Upgrade Guide* .

Guidelines for Modifying Configuration Files

You can modify an application configuration file, but Siebel CRM does not support modifications that you might make in some of these application configuration files. In general, do not modify configuration files except as explicitly described in Siebel documentation or unless instructed by Oracle Global Customer Support. Any file that contains one of the following extensions and is part of the Siebel CRM installation is a Siebel configuration file:

- cfg
- css
- gif
- htm
- xsl
- sws
- sw
- ctl
- sql
- ucf
- rox
- rod
- prd

Setting Up the Configuration File for Siebel Tools

This topic describes how to set up the configuration file for Siebel Tools. For more information, see *Using Siebel Tools* .

To set up the configuration file for Siebel Tools

1. Open the `tools.cfg` file in a text editor.
2. Set the EnableToolsConstrain parameter to FALSE.

   For more information, see the following.
3. Make sure the ClientConfigurationMode parameter is not set to All.

   You cannot use the Form Applet Wizard, List Applet Wizard, View Wizard, or set the HTML Sequence if the value of the ClientConfigurationMode parameter is All.
4. Save the tools.cfg file.
5. If Siebel Tools is open, then exit out of it, and then open it.

How the EnableToolsConstrain Parameter Affects Text Strings

The following table describes how the EnableToolsConstrain parameter affects text strings. For more information, see *Using Siebel Tools* .

Task	If EnableToolsConstrain Is TRUE	If EnableToolsConstrain Is FALSE
Creating a text string	To enter a value for a translatable text string, such as an Applet Title, you must choose from a list of string references.	You can use the string override property to override the string reference.
Creating a symbolic string	You cannot create a custom symbolic string.	You can create a custom symbolic string.

You can use one of the following configurations to create a custom text string:

- Use a symbolic string to create a translatable text string.
- Enter a value in a string override field. For an example, see *Validating Data That the User Enters in a Business Component Field*.
- Add an HTML tag that modifies a text string. For more information, see *Modifying the Text Style of a Control or List Column in an Applet*.

Displaying Object Types You Use to Configure Siebel CRM

You can display object types in the Object Explorer that you use to configure Siebel CRM.

To display the object types that you use to configure Siebel CRM

1. Open Siebel Tools.
2. Click the View menu, and then click Options.
3. Click the Object Explorer tab.
4. Scroll down through the Object Explorer Hierarchy window until you locate the Entity Relationship Diagram tree.
5. Make sure the Entity Relationship Diagram tree and all child objects of the Entity Relationship Diagram tree include a check mark.

   If all child objects in the Entity Relationship Diagram tree are displayed, then Siebel Tools displays a check mark with a background for the tree.
6. Repeat Step 4 and Step 5 for the following object types:
   - Task Group and all children of the Task Group object type.
   - View and all children of the View object type.
   - Import Object and all children of the Import Object type.
   - Control User Prop and List Column User Prop, which are children the Applet object type.
   - Business Component User Prop, which is a child of the Business Component object type.
   - Class object type.
   - Other object types, as necessary.
7. Click OK.

Creating Scripts to Configure Siebel CRM

This topic describes how you can use Siebel Visual Basic, Siebel eScript, and browser script to write scripts that configure Siebel CRM. It includes the following information:

- *Scripts That You Write for the Server*
- *Handling Browser Script in Siebel Versions Prior to 20.8*
- *Browser Script Architecture*
- *Scripts That You Write for the Browser*
- *Creating Browser Scripts*

A script is associated with an object and event in the Siebel Event Model.

Scripts That You Write for the Server

Siebel Tools includes the following scripting languages:

- **Siebel Visual Basic.** Similar to Microsoft Visual Basic. It supports scripting only on the Windows operating system.
- **Siebel eScript.** Compatible with JavaScript. It supports scripting in Windows and other operating systems, such as UNIX.

You can use Siebel Visual Basic and Siebel eScript to do the following work:

- Integrate Siebel CRM with a third-party application.
- Configure the base functionality of the screens and business components in Siebel CRM.
- Develop a data validation routine to enforce rules before or after Siebel CRM manipulates records. Siebel CRM does validation routines before the user updates or inserts a record. This configuration makes sure that the user does not enter data into the database that is not logical or is not complete.
- Develop a data manipulation or computational routine to modify or analyze data.
- Develop a data transport routine to import and export small volumes of data between Siebel CRM and a third-party application.

You use the Script Editor, Debugger, and Compiler to develop and test Siebel Visual Basic script, Siebel eScript script, or browser script. Siebel CRM integrates this capability with the Applet Web Template Editor. You can attach a script to a control that Siebel CRM displays in the Siebel client, such as a field.

You can associate a server script with the following object types:

- Web Applet
- Business Component
- Business Service
- Application

For more information about:

- Scripting, see *Siebel eScript Language Reference* and *Siebel VB Language Reference* .
- Redeploying a script written for a prior release of Siebel CRM in the Siebel client, see *Siebel Database Upgrade Guide* .

Simultaneous Use of Siebel Visual Basic Script and Siebel eScript

To respond to various client events, you can use Siebel Visual Basic and Siebel eScript simultaneously in the same environment but not in the same object. It is recommended that you use Siebel eScript only because it works on UNIX and Windows servers. When you initially add a script to an object, Siebel Tools prompts you to choose the scripting type.

Handling Browser Script in Siebel Versions Prior to 20.8

This topic explains that Siebel 20.8 and later stores browser scripts in the S_RR tables so genbscript is not needed, while earlier versions require configuration updates and running genbscript to generate and deploy browser script files.

After Siebel version 20.8, browser scripts are compiled and stored in the S_RR* tables. There is no need to run the genbscript utility to extract them from the Repository and store them in a local file.

Before Siebel version 20.8, the genbscript utility is still necessary. Without following these steps genbscript may fail.

Steps to generate browser scripts in Siebel versions prior to 20.8.

Configuration Files

1. Update Configuration File
   a. Choose the configuration file used to start your client. This could be uagent.cfg, fins.cfg and so on.
   b. Update the DataSourceName parameter under the [DBSecAdpt] section to **ServerDataSrc**.
   Example

```
[DBSecAdpt]
 DBSecAdpt_SecAdptDllName = sscfsadb

DataSourceName = ServerDataSrc
 DBSecAdpt_PropagateChange = TRUE
```

2. In the same file, under the [ServerDataSrc] section, make sure the following two values are set. The ConnectString parameter points to the Data Source Name (DSN) or entry in the tnsnames.ora file that points to your database. In this example, there is a DSN or tnsnames.ora section named “DEV01”. Ensure that this DSN or tnsnames.ora entry exists on the machine and that connectivity to the database is successful.
   Example

```
Docked = TRUE
 ConnectString =
 DEV01
```

> **Note:** You can run genbscript on a client machine or on the server. If you are using the server, there are a few changes to the configuration file that are different than the changes made if you are running genbscript on a client machine. The above example uses a *client* configuration file such as uagent.cfg (found in the client machine’s <Siebel Installation Directory>\bin directory). If you are using a configuration file (They are the same files as found on a client machine such as uagent.cfg, fins.cfg, they are just located on the server.) from the siebsrvr\bin directory, set the ClientRootDir parameter to <install directory>\ses\siebsrvr. Also set the ServerDBODBCDataSource parameter to the DSN name or tnsnames.ora entry that connects to the database.

   Example

```
ClientRootDir = <Siebel Install
 Directory>\ses\siebsrvr
ServerDBODBCDataSource = DEV01
```

You must also populate the [ServerDataSrc] section's parameters for running genbscript on the server.

Example

```
[ServerDataSrc]
 Docked = TRUE
 ConnectString =
 DEV01
 TableOwner = SIEBEL
 DLL = sscdo90.dll
 SqlStyle =
 OracleCBO
```

Running the Genbscript Utility

1. Use the following command line from the \bin directory to run genbscript.
   a. After the utility completes your browser scripts will be in a new browser script folder srf_xxxxxxxxxx_xxx will be created in the dest_dir. Each time your run genbscript, a different folder name is used because the current date/time is used for the folder name.

   ```
   genbscript /c <config_file> <dest_dir> /l <LANG> /u <User id>
    /p <Password> /d <Data source name>
   ```

   Example of running this on a client machine:

   ```
   cd C:\Siebel\Client\bin
   genbscript.exe /c enu\uagent.cfg C:\Siebel\Client\public /l enu /u SADMIN /p SIEBEL /d
    ServerDataSrc
   ```

   Example of running this on a server machine:

   ```
   cd C:\Siebel\ses\siebsrvr\BIN
   genbscript.exe /c enu\uagent.cfg C:\Siebel\ai\applicationcontainer\webapps\siebel /l enu /u
    SADMIN /p
   ********* /d ServerDataSrc
   ```

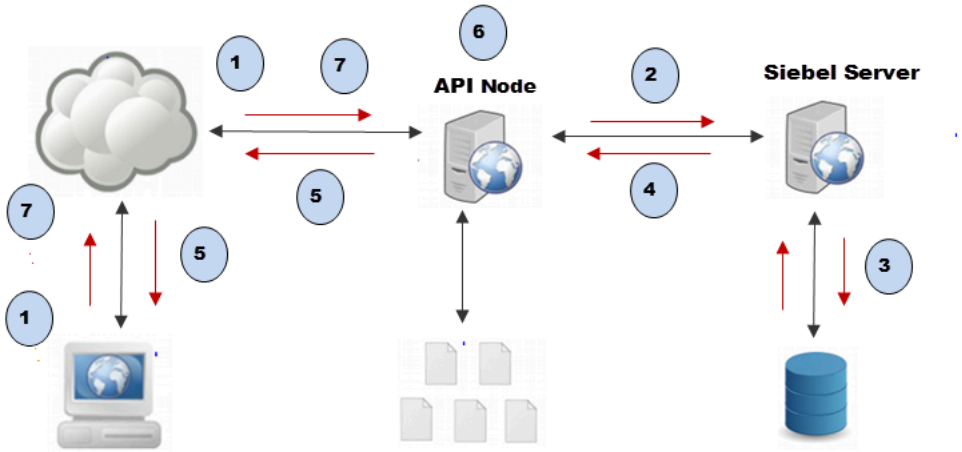
2. After the files have been generated, put the entire srf_xxxxxxxxxx_xxx folder and its contents in the following locations.

Client Type	Browser Script Folder Location
Web Client	**siebel_home\ai\applicationcontainer\webapps\siebel**
Dedicated Client	**client\public**

Browser Script Architecture

The Browser Script architecture supports parallel development of browser scripts for those objects that support it, such as Applets and BusComps. It provides this capability by reading browser scripts directly from the various runtime definitions stored in the runtime tables, such as S_RR_APPLET and S_RR_BUSCOMP, which contain the information needed to get the correct version of the script for the user's current workspace context.

The following figure shows how the Browser Script Architecture works.



As shown in this figure, the Browser Script Architecture works as follows:

1. A client requests a browser script from the web server.
2. The Siebel Application Interface requests the definition from the Siebel Server.
3. The Siebel Server retrieves object definition from the Runtime Repository table, such as S_RR_APPLET, if the object definition is not found in the runtime repository cache.
4. The Siebel Server returns the script to the Siebel Application Interface.
5. The Siebel Application Interface delivers the content to the requesting client.
6. The browser can cache the request with the help of a standard HTML cache-control header.
7. Future requests for the same browser script can be delivered by the browser if the cache is managed.

This architecture provides the following benefits:

- Provides true workspace enablement as the browser script is managed within the workspace framework just like any other object type.
- Simplifies deployment by reading directly from the RR definition for an object such as an applet or BusComp.

Creating Browser Scripts

To create a browser script, select an object that supports it, such as an Applet or BusComp, and select "Edit Browser Scripts" from the list applet menu and add the appropriate script(s).

The browser script can be tested in that workspace by opening an application such as Call Center and inspecting the workspace.

Once the script has been tested and delivered to its parent workspace, it will be compiled into the runtime repository table for that object, such as S_RR_APPLET or S_RR_BUSCOMP, and will be migrated from the DR environment to downstream RR environments via the Siebel Migration Application.

> **Note:** If you are performing a migration to a Runtime Repository environment on an earlier version of Siebel CRM (pre-20.8), you will need to manually run the genbscript utility in the destination environment. For more information on running the genbscript utility, refer to the relevant Bookshelf documentation for the target version.

Scripts That You Write for the Browser

Browser script allows you to use *JavaScript*, which is an interpreted language that runs in many web browsers. A browser script responds to an event on a browser Java object. This browser object works with the corresponding object that runs in the object manager. The set of events that you can script with a Browser object type is different from the set of events that you can script with a server script:

- For Siebel CRM, you can script a wide variety of events that the browser supports. An HTML control does not support the OnClick event. For more information, see *Siebel eScript Language Reference* .
- For a Siebel employee application, you can only script on the OnBlur or OnFocus events.

You use Siebel Tools to write a browser script. You can associate a browser script with the following object types:

- Applet
- Business Component
- Business Service
- Application

Note: A browser script on a business component is appropriate only if Siebel CRM displays the Siebel object that the script references in the Siebel client.

Hierarchy of Object Types That Siebel CRM Uses With a Script

The following figure shows the hierarchy of relationships between object types that Siebel CRM can use with a script. There is a one-to-many relationship between the following objects:

- Applet and Applet Browser Script

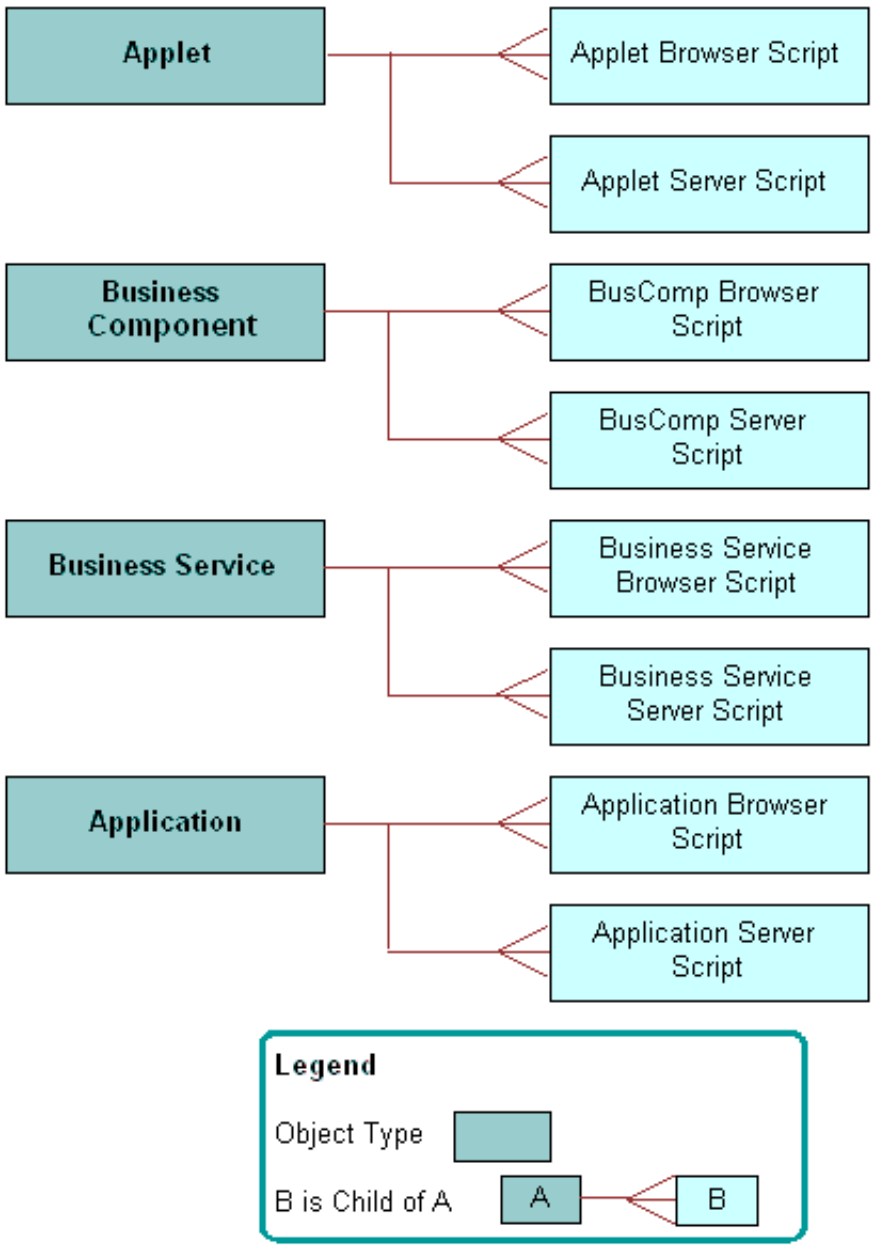
  Applet and Applet Server Script
- Business Component and BusComp Browser Script

  Business Component and BusComp Server Script
- Business Service and Business Service Browser Script

  Business Service and Business Service Server Script
- Application and Application Browser Script

  Application and Application Server Script

To script a browser event, you use a child object type of the parent. You can use these object types with their server counterparts in Siebel Visual Basic, JavaScript, or Java.


Applet
Applet Browser Script
Applet Server Script
Business Component
BusComp Browser Script
BusComp Server Script
Business Service
Business Service Browser Script
Business Service Server Script
Application
Application Browser Script
Application Server Script
Legend
Object Type
B is Child of A
A
B

10 Reusing Predefined Objects

Reusing Predefined Objects

This chapter describes how to reuse a predefined object. It includes the following topics:

- *Reasons to Reuse or Not Reuse a Predefined Object*
- *Guidelines for Reusing a Predefined Object*
- *Process of Determining Whether You Can Reuse a Predefined Object*

It is recommended that you reuse predefined objects to configure Siebel CRM. You must avoid making significant configuration of Siebel CRM, and that you attempt to reuse and configure predefined objects where possible. Situations exist when reusing a predefined object is not appropriate and can cause problems. This topic describes when to reuse and when not to reuse a predefined Siebel object.

Reasons to Reuse or Not Reuse a Predefined Object

This topic describes reasons to reuse or not reuse predefined object.

Reasons to Avoid Extensive Customization of Siebel CRM

Customization is the act of performing significant modifications to the predefined product, such as making the following modifications:

- Creating new modules that do not exist in the predefined Siebel application. This work typically involves configuring the database, and creating many new business components and business objects.
- Modifying a significant number of predefined objects.
- Making significant modifications to predefined behavior, such as visibility.
- Making significant modifications to framework objects, such as JavaScript files.
- Writing a significant amount of custom scripts.

Inappropriate customization of Siebel CRM can cause the following problems:

- Decreased maintainability.
- Increased cost of ownership.
- Potential for decreased performance. A predefined Siebel application is tuned for performance.
- Potential affect on future upgrades.
- Increased testing effort.
- Inconsistent application behavior.

Why Reusing Objects Is Important

Reuse involves building components that you can reuse and configure. Reuse allows you to limit the amount of customization in your deployment. Any modifications that you make to the predefined configuration must maximize reuse and allow for easy customization. Several ways exist to reuse components in Siebel CRM. For example:

- Use predefined configuration objects. For example, business components, business objects, links, applets, views, and so on.
- Use Siebel declarative configurations and tools to translate business requirements into application behavior. For example, Siebel Tools, Siebel Workflow, Task UI, personalization, run-time events, and state model.

Guidelines for Reusing a Predefined Object

This topic describes guidelines for reusing a predefined object. It includes the following information:

- *Reasons to Reuse a Predefined Object*
- *Guidelines for Reusing a Predefined Table*
- *Guidelines for Reusing a Predefined Business Object*
- *Guidelines for Reusing an Applet*
- *Guidelines for Reusing a Predefined View*
- *Guidelines for Reusing a Predefined User Interface Object*
- *Reasons Not to Reuse a Predefined Object*

Reasons to Reuse a Predefined Object

The following reasons describe why it is recommended that you copy, and then modify a predefined object rather than create a new object:

- Oracle configures many predefined objects for optimal performance. A custom object you create might not be configured for performance.
- When troubleshooting, you can use the object in the sample database to revert back to the original object. You can use the comparison feature in Siebel Tools to determine what modifications were made to the object that might cause the problem.
- Repository and application maintenance requires less time and fewer resources.
- Eliminating unnecessary copies of objects reduces the amount of redundancy in the Siebel repository.
- Oracle thoroughly tests a predefined object, so less effort is required to test Siebel CRM or to resolve an application error.
- By reducing the number of repository objects that you must evaluate or upgrade, less effort is required when you upgrade Siebel CRM.

> **CAUTION:** It is recommended that you do not modify administration objects. For example, objects in the Administration - Server Configuration and Administration - Server Management views, and the List Of Values business component. Modifying these objects might cause unpredictable behavior.

Guidelines for Reusing a Predefined Table

If you must create a custom business component because no predefined business component meets your requirements, then you must decide to reuse a predefined table or to create a new table. You must only reuse a predefined table that is a suitable fit. Several reasons exist for this guideline:

- Oracle tunes the indexing for the table for the originally intended use. Using it for an alternative purpose might reduce performance.
- The user key table for the unique indexes might not meet your requirements. For a predefined table, you cannot modify these objects. If a user key column does not contain the required data, then the uniqueness of the record, performance, and Enterprise Integration Manager might be compromised.
- The dock object visibility rules might not meet your requirements. Modifying the rules for the table might compromise Siebel Remote. For more information, see *Configuring Dock Objects for Siebel Remote*.

If you do not reuse a table appropriately, then future reuse of that table for the original purpose of the table might be difficult. For example, assume you use the S_CALL_LST table to store data that is not related to a call list. If you later implement predefined list management, then Siebel CRM displays data that is not related to the call list in the list management views. Adding a search specification to remove this data might compromise performance, and adding an index might or might not correct this problem. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.

You must not modify any repository table. If the Type property of a table is Repository, then it is a repository table.

For more information, see *Reusing Predefined Objects*.

Guidelines for Overloading a Table

You might overload a table if you reuse it multiple times on different business components, and if each business component includes a search specification in a Type field. For example, if you use the S_EVT_ACT table to store regular activities, audit logs, error logs, messages, EAI logs, and so on.

If you overload a table, then to prevent Siebel CRM from display data from one business component in another business component, it is often necessary to add a search specification that queries a Type field.

Overloading a table can cause the following problems:

- The search specification that Siebel CRM uses to type the table into various business components might cause performance problems. Siebel CRM might not design the table to be overloaded. For example, it often uses the TODO_CD column of the S_EVT_ACT table to set the type of the table. Siebel CRM does not denormalize this table on to the S_ACT_EMP intersection table for activities and employees. A query that uses the sales representative visibility against a business component that references the S_EVT_ACT table might result in poor performance.
- No guarantee exists that adding indexes against these type columns will resolve a performance problem because adding an index might compromise performance elsewhere. The fact that Siebel CRM might not denormalize a type column onto a position, employee, or organization intersection table might affect a query in some views.
- Overloading a table increases the table size.

For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.

Oracle Designs Some Tables to Be Overloaded

Oracle designs some tables in the Siebel schema to be overloaded. For example, the S_ASSET table uses the TYPE_CD column to set the type for various business components. Oracle denormalizes and indexes this column onto the S_ASSET_POSTN and S_ASSET_BU intersection tables to improve performance in SalesRep and Organization visibility views. Some XM tables, such as S_ORG_EXT_XM, are designed to be overloaded.

Guidelines for Using an Extension Table as the Base Table of a Business Component

You must never use an Extension or Extension (Siebel) table as the base table of a business component. Siebel Enterprise Integration Manager and Siebel Remote assume that the PAR_ROW_ID and ROW_ID columns on these tables are equivalent and that the PAR_ROW_ID column references a valid parent table record. For more information, see *Extension Columns of a Siebel Table*.

Guidelines for Creating a New One-To-One Extension Table

In most situations, you must extend the base table or reuse an ATTRIB column on a one-to-one extension table.

Guidelines for Creating a New XM Table

In most situations you must reuse a predefined XM table to support a one-to-many relationship from a base table. If you use an XM table, then use the following guidelines:

- Very few situations exist that require you to create a new XM table. Siebel CRM already tunes XM tables to support large data volumes and multiple data types. Before you create a new XM table, make sure to examine the predefined XM tables to determine whether one meets your requirements.
- In some instances, a base table does not reference an XM table. Instead of reusing another unsuitable predefined table because it contains a foreign key to the base table, you must create a new table. For example, if you require a one-to-many business component from the S_EVT_ACT table, then you must create a one-to-many table rather than reuse a table that might be inappropriate, such as the S_ACT_TIMESTAMP table, provided that the business component does not store timestamp information.

For more information, see *How an Extension Table Stores Custom Data*.

Guidelines for Using a Table That Is Not an Intersection Table to Create an Intersection

You must reuse an intersection table only where it is a true intersection between two tables. The type of table must be Data (Intersection). Do not use a table that is not an intersection table as an intersection table only because it contains a foreign key to the table. This configuration can overload the table. For more information, see *Reusing Predefined Objects*.

Do not use a one-to-many XM table as an intersection table. Siebel CRM does not tune an XM table for this usage, and using it as an intersection table might cause poor performance. To support one side of the relationship, you must create a custom foreign key. This configuration might cause problems with Siebel Remote and Enterprise Integration Manager.

If no suitable intersection table exists between two tables and one is required, then you must configure the database. For more information, see *How an Extension Table Stores Custom Data*.

Guidelines for Using a Table That Is Not Licensed

Do not reuse an unused table that is not licensed for your configuration.

Guidelines for Using the S_PARTY Table to Support a Custom Party Type

Using a custom party type compromises access control and remote visibility. For more information, see *How the S_Party Table Controls Access*.

Guidelines for Reusing a Predefined Business Component

Inappropriately using a predefined business component can make it difficult to use the same business component for the intended purpose in a future release. For example, if you use the Service Request business component to store financial transactions in one release, then you might be prevented from using the same business component to store actual service requests in a future release. For more information, see *Guidelines for Creating a Business Component*.

if you reuse a predefined business component, then use the following guidelines:

- If you require a business component that is similar to a predefined business component, then do one of the following:
  - Create a new business component that references the predefined business component.
  - Modify the predefined business component.

  Oracle prefers that you modify a predefined business component because it minimizes the number of business components in your configuration. This situation leads to a smaller repository that is easier to maintain and upgrade because it is more closely aligned with the predefined Siebel application.
- Use the business component in a way that is consistent with the intended use of the business component. For example:
  - Use the Contact business component to store individual details for each contact at a customer site.
  - Use the Account business component to store details of the business relationship with the customer.
  - Do not use the Service Request business component to store information that is not related to a service request, such as a financial transaction or order history.
- If you reuse a business component, then configure it to be as flexible and reusable as possible. For example:
  - In one release you use the Service Request business component to store customer complaints.
  - In another release, you use the Service Request business component to store addresses for customers who changed their address.

  In these situations, you must use the Service Request business component rather than using the business component in a subsequent release for other service transactions. For example, you could use the SR Type field to distinguish between the two service transactions. Your business requirements must be as generic as possible to facilitate the use of a single business component.
- Always configure Siebel CRM in a way that allows you to reuse a business component instead of creating a new business component. For example, Siebel CRM can allow one group of users to create new opportunities, but another group can only edit existing opportunities. Instead of creating a new business component and setting the No Insert property to TRUE, you can create a new applet and set the No Insert property to TRUE for the applet.

Guidelines for Copying a Predefined Business Component

If you copy a business component, then you must copy the links that this business component uses, and then update the copies with the new business component name. You can avoid errors if you copy and update links. For example, if you clone the Service Request business component and name the clone Service Request ABC, then you must copy the Service Request/Action link and name the copy Service Request ABC/Action.

Guidelines for Reusing a Predefined Business Object

It is recommended that you avoid copying a business object or business component, but copying might be appropriate in the following situations:

- When you must include a business component twice in a business object. For example, the Account business object must include the Account business component and Sub Account business component.
- When two business components contain different search specifications and predefault values for the Type field that differentiates the records of these two business components. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.

Guidelines for Reusing an Applet

To decide to reuse or create an applet, use the following guidelines:

- If your requirements closely align with the functionality of a predefined applet, and if this applet only requires minor modification, then modify it. For example, to modify a title, deactivate, or add a few controls or list columns, or to modify display labels.
- If the predefined applet meets your requirements for a relationship in the data model, such as between an opportunity and contacts, then copy this applet. You can then modify the new copy to make significant modifications, such as the applet layout, resequencing, inactivating objects, or adding many new controls and list columns. This configuration is easier to maintain and upgrade.
- If your requirements demand a different drilldown to a different view, then copy a predefined applet, and then modify this copy.
- If you cannot locate a suitable predefined applet, then create a new applet. For example, if your requirements demand that you display a new business component.
- If you copy an object, then use the Upgrade Ancestor property. For more information, see *Guidelines for Setting the Upgrade Ancestor Property*.

For more information, see the following topics:

- *About Applets, Controls and List Columns*
- *Guidelines for Creating an Applet*

Guidelines for Reusing a Predefined View

To decide to reuse or create a view, use the following guidelines:

- If the requirements for a new view closely align with a predefined view, but require simple modifications, such as modifying the view title, or moderate layout modifications, such as displaying a different applet or adding a toggle, then you can modify the predefined view.
- For a view that consolidates two predefined views, it is recommended that you modify one of these views, and then use the Responsibility Administration screen to remove visibility from the redundant view.
- If the requirements for a view do not align with a predefined view, and if the predefined view requires significant modifications, then you can create a new view. Typically these views implement new functionality that your implementation requires. For example, you might need a view to display new business objects or business components. In these situations, it is easier to maintain and upgrade a new custom view rather than modifying a predefined view.
- If you copy an object, then use the Upgrade Ancestor property. For more information, see *Guidelines for Setting the Upgrade Ancestor Property*.

For more information, see the following topics:

- *About Views*
- *Guidelines for Creating a View*

Guidelines for Reusing a Predefined User Interface Object

Sometimes it is appropriate to copy a user interface object. For example, if your business requirements demand a significant modification to the look and feel of the object, then copying the object and setting the Upgrade Ancestor property makes certain that Siebel CRM preserves the modified look and feel following an upgrade. For more information, see *Guidelines for Setting the Upgrade Ancestor Property*.

If you only require a minor modification to the user interface object, then it is recommended that you use the predefined object because this configuration reduces the time you spend to configure and maintain the Siebel repository. The following reasons describe when it is appropriate to copy a user interface object:

- When you require two different user interface objects to display different records that use different search specifications on applets. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.
- When you require different read and write properties between two objects. For example, one applet is read-only and the other applet is editable. In this situation, only copy the object if the dynamic read-only business component user property does not fulfill this functionality.
- When you require different drilldowns for different applets, depending on the view that contains them. In this situation, copy the object only if a dynamic drilldown does not fulfill this functionality.

If you copy an applet that uses a business component that references a specialized class, then use the following guidelines:

- You must use the copied applet with the original business component, not a copy of the original business component.
- To use a copied applet with a copied business component, you must modify the class of the copied applet.

For more information, see *Caution About Using Specialized Classes*.

Reasons Not to Reuse a Predefined Object

Reusing an object can result in problems.

Copying a Predefined Object Can Cause an Upgrade Problem

Copying an object can cause an upgrade problem that is difficult to debug. Functionality is often added to most of the predefined business components during major releases. This new functionality often depends on new fields, joins, and so on, that Siebel CRM adds to a predefined business component. During the upgrade, Siebel CRM adds these new fields only to the predefined business components that reside in your merged repository.

Copying an object can result in problems following an upgrade. These problems can be difficult to locate and debug. The errors often occur because some C++ code for the business component or applet class attempts to find a field that does not exist in your custom copy of that business component or applet. The only way to debug the problem is to compare your custom business component with the predefined business component, and to add any new fields and other child object definitions that Siebel CRM added in the new release. This work can be complex, requiring detailed knowledge of what Siebel CRM modified in the new release.

Copying a Predefined Object Can Cause Redundancy

Creating a new copy of a business component or applet can result in redundancy in your configuration. For example, if you create a copy of the Account business component and name it My Account, and use this copy on all of the views that reference accounts, then you must define copies of every applet that references accounts, and you must make sure each of these copies references the new My Account business component. It might be necessary for you to create a new business object, screen, and so on.

Copying a Predefined Object Can Increase Complexity

You might make a copy of an object in the belief that doing so will reduce problems during an upgrade. The assumption is that if the business component is named My Account, then the Application Upgrader will leave it alone during the upgrade, resulting in no problems after the upgrade. This assumption is misleading. The problems you might encounter with an upgraded configuration that contains a copied object might be more complex to solve than the problems that reusing a predefined object causes. It is easier to examine your Siebel application after an upgrade and remove various new controls and list columns from a predefined applet than it is to examine each custom business component and applet to identify the fields, joins, multi-value links, and so on, that you must troubleshoot.

Results of Reusing An Object Inappropriately

Reusing a Siebel module or repository object in a way that does not meet the original purpose of the object can cause the following problems:

- Performance might be degraded. Oracle tunes the performance of the component or object for a specific purpose.
- Future use of the object might be limited. For example, the Service Request business component stores a service interaction with a customer. If you modify this usage, you might limit your ability to use this business component for the original purpose in a future release.
- Large amounts of customization might be required to make the improper use of an object work correctly. For example, if you reuse a table for an alternative purpose, then you must configure Siebel CRM so that it updates all required columns and adds a search specification to the business component. If you use a predefined

business component for a purpose other than the original design intent, then specialized class behavior might exist that affects your ability to properly use the business component. It might be necessary for you to perform more customization to make the business component work so that it meets your requirements. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet* and *Caution About Using Specialized Classes*.

- The intended purpose of the reused object might not be clear to future developers. For example, if you use the S_SRV_REQ table to store financial transactions, then this configuration is not clear to future developers because the table bears no relationship to financial transactions.
- Upgradability might be decreased. Troubleshooting and reconfiguration might be required after an upgrade because the upgrade might revert an object to the original form. Modifications that Siebel CRM makes to objects during a repository upgrade depends on Siebel CRM using objects for their original purpose. An upgrade might include any of the following modifications:
  - Modify the table schema by adding or modifying unique indexes or required columns.
  - Modify the behavior of specialized classes.
  - Modify functionality. For example, access control, visibility, and so on.
  - Modify the party model.
- Objects that Siebel CRM does not include in your licensed configuration might be included in your deployment.

Process of Determining Whether You Can Reuse a Predefined Object

This task is a step in *Roadmap for Configuring a Siebel Application*.

To determine whether you can or cannot reuse a predefined object, do the following tasks:

1. *Determining Functional Fit for Reusing a Predefined Object*
2. *Determining Technical Fit for Reusing a Predefined Object*
3. (Conditional) *Determining Whether You Can Reuse a Predefined Table Column*
4. (Conditional) *Determining Whether You Can Reuse a Predefined Business Component Field*
5. (Conditional) *Determining Whether You Can Reuse a Predefined Business Component*

It is recommended that you use the predefined objects, but there might be situations when it is difficult to determine to reuse a predefined object or to create a new object. This situation occurs if predefined objects cannot meet your requirements. You must determine the functional and technical fit of the proposed use. If the fit is appropriate, then you can reuse the object. If it is not appropriate, then you must create a new object. You must not reuse an object merely because another Siebel application does not already use it.

If no predefined object is suitable, then you must consider configuring the data objects layer. For more information, see *Options to Configure the Data Objects Layer*.

To determine whether an object is a functional fit to your business requirement, you can examine the table or business component that you intend to use.

Determining Functional Fit for Reusing a Predefined Object

This task is a step in *Process of Determining Whether You Can Reuse a Predefined Object*.

To determine functional fit for reusing a predefined object

1. Determine whether you must reuse the object rather than copy the object.
   For more information, see *Reasons to Reuse or Not Reuse a Predefined Object*.
2. Determine whether you must copy the object rather than reuse the object.
   For more information, see *Guidelines for Reusing a Predefined Object*.
3. Make sure the original nature and purpose of the Siebel object is compatible with your proposed use.
   For example, storing customer complaints is compatible with the Service Request business component, but not for storing financial transactions.
4. Make sure relationships to other objects are compatible with your requirements.
   The fact that an object contains the correct relationships is not sufficient for reuse. For example, you must not use the S_EVT_ACT table as an intersection table only because it contains two of the foreign keys that you require. Doing so can cause Siebel CRM to overload the table and degrade performance.
5. Determine whether the visibility properties of the object are or are not compatible with your requirements.

If the object is not a good functional fit, then reusing the object for that purpose might be inappropriate. The following are examples of improper use:

- Using the S_PARTY table to store a nonparty entity.
- Using an unused table for a custom business component if the table does not possess a relationship to the intended usage of the business component.
- Using an unused table column or business component field that does not possess a relationship to the intended usage of the field.

Determining Technical Fit for Reusing a Predefined Object

This task is a step in *Process of Determining Whether You Can Reuse a Predefined Object*.

To determine technical fit for reusing a predefined object

1. Examine the following technical factors:
   - Performance factors
   - Size and type of columns and fields
2. Examine the following table schema factors:
   - Determine whether you must set columns to a default value.
   - Determine whether you must configure Siebel CRM to update the user key and unique index columns.

   If you must perform a large amount of customization to use an unused table, then technical fit diminishes.
3. Determine the affect that the foreign key relationships have on Siebel Remote.
   Foreign key relationships and Siebel Remote are closely interrelated. Simply using the correct foreign key might not make sure that Siebel CRM downloads the data to the Remote client. You must determine how reuse affects the dock objects and the rules that interact with the foreign keys. For more information, see *Configuring Dock Objects for Siebel Remote*.

4. Determine the affects that foreign key relationships have on visibility.

   Many columns that are not foreign keys can affect visibility. For example, S_PROD_INT.ENTERPRISE_FLG with a value of Y provides partial docking visibility to the product record. Misusing these columns can result in a significant negative affect on Siebel Remote.

Determining Whether You Can Reuse a Predefined Table Column

This task is a step in *Process of Determining Whether You Can Reuse a Predefined Object*.

Reusing a predefined table column if a good technical or functional fit does not exist can lead to a poor result. In this situation, it is better to create a new, custom field or column. For more information, see *Options to Configure the Data Objects Layer*.

To determine whether you can reuse a predefined table column

1. Consider the table columns that Siebel Remote uses:

   a. Do not use a field or column that controls Siebel Remote behavior.

      Siebel Remote uses the ENTERPRISE_FLG column to implement visibility on records that use normal visibility constraints. It might use other columns to control download behavior, such as the RTE_CONTACTS_FLG or RTE_PRSP_CONS_FLG column on the S_SRC table. For more information, see *Configuring Dock Objects for Siebel Remote*.

   b. Click the Flat tab in Siebel Tools, and then submit one of the following queries to determine whether a dock object visibility rule uses a particular column:
      - Query the SQL Statement and DBX SQL Statement properties of the Dock Object Visibility Rule object for the column you are considering.
      - Query the Filter Sql Statement property of the Dock Object Table object for the column you are considering.

      Note that you must display the object type known as the Dock Object and the children of this object type. For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.

   If you are in doubt about how a column might affect Siebel Remote, then see *Getting Help From Oracle*. For more information, see *Siebel Remote and Replication Manager Administration Guide* .

2. To reuse a table column or business component field that references a LOV (list of values) bounded column, make sure you use a bounded list that uses the same LOV type as the column object**.**

   If the LOV Bounded property of the column is TRUE, then Enterprise Integration Manager enters data into the column only if the corresponding LOV type and value exist in the S_LST_OF_VAL table. For more information, see *Creating a List of Values*.

3. Consider the column type and size.

   In most situations, you must not modify the type or size of the column. An exception is IBM DB2 for z/OS operating systems, which store the maximum size of a VARCHAR column in an index. For more information, see *Implementing Siebel Business Applications on DB2 for z/OS* .

4. Make sure a column you use is not obsolete.

   Examine the Comments property of the column to determine whether the column is obsolete. You must not use an obsolete column because Siebel CRM might delete it in a future release.

5. Examine the Foreign Key Table property of the Column object to determine whether the correct foreign key relationship exists:
   - If a field on the business component already uses the key, then reuse that field rather than creating a new field. The original purpose of the unused foreign key field or column must match your intended use.
   - Do not use a foreign key column that does not contain the correct foreign key relationships.

   For more information, see *Guidelines for Considering a Foreign Key Relationship*.
6. Use a one-to-one ATTRIB column.

   If no other suitable field or column exists, then some one-to-one tables contain generic ATTRIB columns that you can use. For example, the S_CONTACT_X table. If you use an ATTRIB column, make sure you do the following:

   a. Extend a base table with a custom extension column instead of using an ATTRIB column in the following situations:
      - For foreign keys and the Primary ID Field. For more information, see *About the Primary ID Field*.
      - For a column that Siebel CRM frequently queries or is always present in the result set. For example, a field that is in the initial list view of a screen, or a field whose Force Active or Link Specification property is TRUE.

   b. If you reuse a predefined ATTRIB column, then make sure another field does not use it. If another field does use it, then choose another unused ATTRIB column.

   For more information, see *Options to Use a Predefined One-to-One Extension Table*.
7. Reuse an unused column for a new business component field.

   You must verify that another field on the same business component does not already use the column. If more than one field references the same table column, then you might encounter a duplicate column insert error during a copy operation. In this situation, you must use the original Siebel field that references the column. Otherwise, use another appropriate column, such as a custom extension column or an unused ATTRIB column.
8. Use a user key column.

   Note the following:

   - Do not use a column that is part of the user key of a table for any other purpose than how Siebel CRM intends to use this column. Doing so might result in degraded performance. For more information, see *Siebel Performance Tuning Guide* .
   - Do not enter data into a nonforeign key value or map the foreign key to a different table as a way to map a user key column that is a foreign key to a table. Enterprise Integration Manager (EIM) uses the user key to identify a unique record. Inappropriately entering data into a user key column that references a foreign key might prevent Enterprise Integration Manager from working correctly.

   For more information, see *How a User Key Creates a Unique Set of Values*.

Guidelines for Considering a Foreign Key Relationship

You must not reuse a table only because it contains the necessary foreign key relationship. Instead, to add the required foreign key columns, you can do one of the following:

- Configure the database.
- Use the Siebel Dock Object Wizard.
- Use the EIM Table Mapping Wizard.

Do not use a predefined column that is not a foreign key to store a custom foreign key. Doing so can affect Siebel Remote and Enterprise Integration Manager. This situation can cause problems when Siebel CRM creates EIM mappings or routes data to the user.

Determining Whether You Can Reuse a Predefined Business Component Field

This task is a step in *Process of Determining Whether You Can Reuse a Predefined Object*.

This topic describes the factors you must evaluate when you consider reusing a predefined business component field. Reusing a field if a good technical or functional fit does not exist can lead to a poor result. In this situation, it is better to create a new, custom field. For more information, see *Process of Determining Whether You Can Reuse a Predefined Object*.

To determine whether you can reuse a predefined business component field

1. Examine the Primary ID Field.

   In some situations, you can configure the Primary ID Field for a multi-value link to improve performance. For more information, see *About the Primary ID Field*.

   For a custom multi-value link, you must attempt to reuse an unused column or the Primary ID Field before you create a new custom extension column:

   a. Make sure the Foreign Key Table property of the unused column references the table of the business component of the multi-value link.
   b. Make sure the Primary Child Col property is TRUE.
   c. Make sure the Primary Child Table, Primary Child Join Column, and Primary Join Column Name properties are set with an appropriate value.

      For a many-to-many relationship, the Primary Inter Table Name must reference the intersection table. You cannot set these values for a base table column that is predefined. You must make sure that the unused field or column is the appropriate Primary ID Field.
   d. If you cannot locate an appropriate unused primary field or column, then you must verify that another multi-value link does not already use it.

      For more information, see *Sharing a Primary for a Multi-Value Link*.
   e. If you cannot find a suitable unused Primary ID Field or column, then you must extend the base table and create a custom field.

      Note that the EIM Table Mapping Wizard does not create an EIM explicit primary mapping object for a custom Primary ID Field. For more information, see *Defining the Primary ID Field of a Multi-Value Link*.
2. Make sure the field you use is not inactive.

   Do not reactivate a field that is currently inactive in a predefined Siebel application for the following reasons:

   - The field might be obsolete. Siebel CRM might delete it in a future release.
   - The field might be part of future functionality that Siebel CRM has not yet implemented.

3. Make sure a specialized business component does not reference the field.

   You can use a field on a specialized business component only for the purpose that Siebel CRM intends to use this field. An unintended use of this field might affect specialized behavior. For more information, see *Caution About Using Specialized Classes*.
4. Use a user key field.

   For more information, see *Determining Whether You Can Reuse a Predefined Table Column*.

Examples of Reusing a Business Component Field or Table Column

The following are examples of how you might reuse a business component field or a table column:

- Use the Last Name or First Name field of the Contact business component to store the name of a contact.
- On the Account business component, instead of creating a custom extension column for a new field to store an email address, use the EMAIL_ADDR column of the S_ORG_EXT table.
- On the Order Entry business component, use the Revision field to store the revision number. Specialized behavior controls this field. For more information, see *Caution About Using Specialized Classes*.
- On the Campaign business component, use the Route Prospects field to store information that describes the campaign contacts that Siebel CRM routes to the Remote client. If you store other information, then Siebel Remote performance might be compromised.
- Use the WEIGHT column on the S_CONTACT table to store the weight of the contact. Do not use it to store information that is not related to the weight of the contact.

The Comment property of a column often describes the intended purpose of the column. Use comments to help you decide when to reuse a column.

Sharing a Primary for a Multi-Value Link

It is recommended that you do not share a primary for a multi-value link that returns different result sets because it can corrupt the primary. For example, assume two multi-value links reference business components that reference the same table but use different search specifications. Sharing the Primary ID Field for the two multi-value links causes Siebel CRM to set the Primary ID Field to the value of No Match Row Id because the value of the Primary ID Field might reference a record that one of the multi-value links does not return. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.

Determining Whether You Can Reuse a Predefined Business Component

Where possible, it is recommended that you reuse a business component in the predefined repository. You must only reuse a business component that provides a valid functional and technical fit with your requirements. If no suitable predefined business component exists, then it might be necessary to create a new one. It might be necessary to configure the database to create a new table.

To determine whether you can reuse a predefined business component

1. Make sure the business component does not use a specialized class.

   Instead of using a specialized business component, create a new business component. If you use a business component that references a specialized class for a purpose other than how Siebel CRM intends to use it, then

the class code might create problems that are difficult to debug. For example, if you use the eEvents Parent Event business component to store data that is not related to events management, then the specialized code might result in the creation of an undesirable subevent. For more information, see *Guidelines for Creating a Business Component That References a Specialized Class* and *Caution About Using Specialized Classes*.

2. Make sure the business component does not reference the S_PARTY table.

   You must never use a business component that references the S_PARTY table for a purpose other than how Siebel CRM intends to use the business component. A business component that references the S_PARTY table influences other areas, such as access control and Remote visibility.

3. Make sure the business component is licensed.

   You cannot use a predefined business component that Siebel CRM does not license for your configuration.

4. Make sure the business component is not a repository business component.

   Never configure a repository business component. The name of a repository business component begins with Repository.

11 Using the Entity Relationship Designer

Using the Entity Relationship Designer

This chapter describes how to use the Entity Relationship Designer. It includes the following topics:

- *About the Entity Relationship Designer*
- *Process of Creating and Binding an Entity Relationship Diagram*
- *Opening or Modifying an Entity Relationship Diagram*
- *Modifying Shapes and Lines in the Entity Relationship Designer*

About the Entity Relationship Designer

The *Entity Relationship Designer* is a visual design tool that you can use to create an entity relationship diagram (ERD). You then map the entities and relationships in the diagram to objects in the Siebel repository, such as business components, links, joins, and so on. The Entity Relationship Designer includes the following capabilities:

- An environment to create an ERD to relocate objects.
- Various edit and layout options, such as aligning shapes, moving shapes, and modifying text.

The Entity Relationship Designer provides the following benefits:

- Filters the list of objects that you choose when you bind entities and relations to Siebel objects. The list includes only the objects that support the context that the ERD represents. If no business components are suitable for binding, then you can open a wizard in the Entity Relationship Designer to assist you with creating a new business component.
- Allows you to use the crows feet diagraming format to define relationships between entities.
- Requires less work to define requirements for the data objects layer.
- Improves your ability to trace configuration modifications back to data object layer requirements.
- Creates a permanent record of entity relationship design in the Siebel repository.

Example of How the Entity Relationship Designer Filters Business Components

The following image shows an example entity relationship diagram that contains two entities (Entity A and Entity C) and one relationship.



The following table describes the business components that are available to bind to Entity C. The business components that are available to bind depends on how the Entity Relationship Designer filters them.

Entity A	Relationship	Entity C	Business Components Available for Binding
Unbound	Any	Unbound	All business components are available for binding.
Bound	one-to-one	Unbound	A business component that contains a join to the primary table of the business component that is bound to Entity A is available for binding.
Bound	one-to-many	Unbound	The following business components are available for binding: • A business component that contains a link to the business component that is bound to Entity A, where the business component bound to Entity A is the parent. • A business component that contains a join to the primary table of the business component that is bound to Entity A.
Bound	many-to-one	Unbound	The following business components are available for binding: • A business component whose primary table is the table that is joined to the business component that is bound to Entity A. • A business component that contains a link with the business component that is bound to Entity A, where the business component bound to Entity A is the child.
Bound	many-to-many	Unbound	A business component that is in the intersection of the one-to-many and many-to-one examples that this topic describes is available for binding.

Example of How the Entity Relationship Designer Filters Links and Joins

Example of How the Entity Relationship Designer Filters Business Components describes an example ERD that includes two entities and one relationship. Assume you bind entities A and C to business components and that you must bind the relationship AC to a link or join. The Entity Relationship Designer filters the list of links and joins that are available for binding that the context described in the ERD requires.

The following table describes the links and joins that are available to bind to relationship AC.

Relationship AC	Objects That Are Available to Bind
one-to-many	The following objects are available to bind: • A join whose source is the business component that is bound to Entity C, and whose destination is the primary table of the business component that is bound to Entity A. • A link between the business component that is bound to Entity A and the business component that is bound to Entity C, where the business component bound to Entity A is the parent and the business component bound to Entity C is the child.
many-to-one	The following objects are available to bind:

Relationship AC	Objects That Are Available to Bind
	• A join whose source is the business component that is bound to Entity A to the primary table of the business component that is bound to Entity C. • A link between the business component that is bound to Entity C and the business component that is bound to Entity A, where C is the parent and A as the child.
many-to-many	A link between business components that are bound to Entities A and C is available to bind.

Process of Creating and Binding an Entity Relationship Diagram

This task is a step in *Roadmap for Configuring a Siebel Application*.

To create and bind an ERD, perform the following tasks:

1. *Creating an Entity Relationship Diagram*
2. *Binding an Entity to a Business Component*
3. *Associating an Entity Attribute with a Business Component Field*
4. *Binding a Relationship to a Link or Join*

Two people are typically involved with creating and binding an ERD:

- A business analyst defines an ERD that represents your business model because the analyst possesses knowledge about the business model.
- A technical architect or developer binds the entities and relationships in the diagram to Siebel objects because the architect or developer possesses knowledge about the data objects layer.

Creating an Entity Relationship Diagram

This task is a step in *Process of Creating and Binding an Entity Relationship Diagram*.

An ERD can include business entities, entity properties, and the type of relationship that exists between entities, such as one-to-one, one-to-many, or many-to-many. Examples of common business entities include accounts, contacts, and addresses.

To create an entity relationship diagram

1. Open Siebel Tools.
2. Make sure object types for the Entity Relationship Designer are displayed.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
3. In the Object Explorer, click Entity Relationship Diagram.
4. In the Entity Relationship Diagrams list, right-click, and then click New Record.
5. Enter a name and associate a project to the new record.

   The project must be locked to enter it in the Project field. For more information, see *Using Siebel Tools* .
6. Right-click, and then click Edit Entity Relationship Diagram.

Siebel Tools displays the canvas of the Entity Relationship Diagram.

7. Relocate an entity from the ERD Palette to the canvas.
8. Click the entity, and then enter a value in the Name property in the Properties window.
9. Repeat step 7 through step 8 for the next entity in your diagram.
10. Relocate a relationship, such as ERD 1:1, from the pallet to the canvas.
11. Connect each end of the relationship to a connector point on one of the entities.

    Before you connect two entities, place them on a horizontal plane, with the preceding entity placed first and the subsequent entity placed next. Place the entities in close proximity to one another. When you drop the relationship, Siebel Tools connects the start point and the end point of the relationship.
12. Repeat step 7 through step 11 until you added and connected all entities and relationships.

Binding an Entity to a Business Component

This task is a step in *Process of Creating and Binding an Entity Relationship Diagram*.

After you create an ERD that includes entities, you can bind an entity to a business component. If you bind an entity, then the Entity Relationship Designer filters the list of business components so that they fit the context of the ERD. For more information, see *Example of How the Entity Relationship Designer Filters Business Components*.

To bind an entity to a business component

1. In the Entity Relationship Designer, right-click an entity, and then click Bind Business Component.
2. In the Bind Business Component dialog box, choose the business component that you must bind to the entity.

   If no business component meets your needs, then you can click New to start the New Business Component Wizard. For more information, see *Configuring a Business Component*.

   After you click OK, Siebel Tools displays the name of the business component and the underlying base table in the entity.

Associating an Entity Attribute with a Business Component Field

This task is a step in *Process of Creating and Binding an Entity Relationship Diagram*.

After you bind an entity to a business component, you can associate an entity attribute with a business component field.

To associate an entity attribute with a business component field

1. In the Entity Relationship Designer, choose an entity that is bound to a business component.
2. Right-click in the Multi Value Property Window, and then click New Record.
3. In the Business Component field, click the arrow, and then choose a field.

   Siebel Tools associates the attribute with the business component field. Note that the pick applet that displays if you click the arrow in the Business Component field is context-sensitive. It displays business component fields that Siebel CRM binds to the entity. If the entity is not bound to a business component, then Siebel Tools displays a set of default fields in the pick applet.

Binding a Relationship to a Link or Join

This task is a step in *Process of Creating and Binding an Entity Relationship Diagram*.

After you bind two entities to business components, you can bind the relationship between them. You can bind this relationship to a link or join. For more information, see *Example of How the Entity Relationship Designer Filters Links and Joins* and *About Links*.

To bind a relationship to a link or join

1. In the Entity Relationship Designer, choose a relationship between entities.

   Note that you must bind the two entities that the entity relationship joins before you can bind a relationship to a link or join.
2. Right-click, and then click Bind Entity Relation.

   In the Bind Relationships dialog box, Siebel Tools displays the joins or links that exist between the two business components.
3. Choose the join or link that best represents the relationship that your ERD describes.

   After you complete the bind, Siebel Tools bolds the relationship in the Entity Relationship Designer.

Opening or Modifying an Entity Relationship Diagram

This topic describes how to open or modify an Entity Relationship Diagram. It includes the following information:

- *Opening an Entity Relationship Diagram*
- *Viewing the Entities and Relations Lists of an ERD*
- *Modifying the Properties of a Relationship*
- *Copying the Drawing of an Entity Relationship Diagram*

Opening an Entity Relationship Diagram

You can open an entity relationship diagram.

To open an entity relationship diagram

1. In the Object Explorer, click Entity Relationship Diagram.
2. In the Entity Relationship Diagrams list, right-click the diagram you must open, and then click Edit Entity Relationship Diagram.

Viewing the Entities and Relations Lists of an ERD

You can toggle between the Entities List and the Relations List in the Object Explorer.

To view the entities or relations list of an ERD

- To view the entities list, in the Object Explorer, expand the Entity Relationship Diagram tree, and then choose the Entity tree.
- To view the relations list, in the Object Explorer, expand the Entity Relationship Diagram tree, and then choose the Entity Relation tree.

Modifying the Properties of a Relationship

You can use the Entity Relationship Diagrams list or the Properties window to modify the properties of a relationship. For example, you can modify the text that Siebel Tools displays at the end points of a relationship.

You cannot use the Entity Relationship Diagrams list or the Properties window to modify the type of relationship, such as modifying a one-to-one relationship to a one-to-many relationship. To modify the type of relationship, you delete the old relationship, and then move the new relationship from the ERD Palette to the canvas.

Using the Properties Window to Modify the Properties of a Relationship

You can use the Properties window to modify the properties of a relationship.

To use the Properties window to modify the properties of a relationship

1. Open an entity relationship diagram.

   For more information, see *Opening an Entity Relationship Diagram*.
2. In the Entity Relationship Designer, choose the relationship you must modify.
3. In the Properties window, edit the property you must modify.

   If you modify the value for the Name, End Name 1, or End Name 2 property, then Siebel Tools updates the labels in the diagram.

Using the Entity Relations List to Modify the Properties of a Relationship

You can use the Entity Relations list to modify the properties of a relationship.

To use the Entity Relations list to modify the properties of a relationship

1. In the Object Explorer, click Entity Relationship Diagram.
2. In the Entity Relationship Diagrams list, locate the entity relationship diagram you must modify.
3. In the Object Explorer, expand the Entity Relationship Diagram, and then click Entity Relations.
4. In the Entity Relations list, locate the record you must modify.
5. In the Properties window, edit the property you must modify.

   If you modify the value for the Name, End Name 1, or End Name 2 property, then Siebel Tools updates the labels in the diagram.

Copying the Drawing of an Entity Relationship Diagram

You can copy the drawing of an ERD and paste it into a third-party application, such as Microsoft Word or Outlook. You cannot copy a drawing from one ERD and paste it into another ERD.

To copy the drawing of an entity relationship diagram

1. In the Entity Relationship Designer, right-click the canvas, click Copy, and then click Drawing.
2. In another application, such as Word or Outlook, choose Paste from the Edit menu.

Modifying Shapes and Lines in the Entity Relationship Designer

This topic describes how to modify the appearance of shapes and lines in the Entity Relationship Designer. It includes the following information:

- *Modifying Shapes in the Entity Relationship Designer*
- *Modifying Relationships in the Entity Relationship Designer*
- *Moving Shapes in the Entity Relationship Designer*
- *Resizing Shapes in the Entity Relationship Designer*
- *Zooming, Displaying, and Snapping the Grid*

Modifying Shapes in the Entity Relationship Designer

This topic describes how to modify shapes in the Entity Relationship Designer. You perform all tasks that this topic describes in the Entity Relationship Designer.

To modify shapes in the Entity Relationship Designer

1. To modify the appearance of a shape in an ERD:
   a. Choose an entity or relationship.
   b. Right-click, and then click Shape Properties.
   c. Modify the properties in the Item Properties dialog box, and then click OK.
2. To choose multiple objects in an ERD:
   a. Choose an entity.
   b. Hold down the shift key.
   c. With the shift key still depressed, click another entity.
   d. Release the shift key.
3. To align shapes relative to each other:
   a. Choose multiple objects in the Entity Relationship Designer.
   For more information, see the following.

b. Right-click the canvas, click Layout, Align, and then click one of the following menu items:

   - Lefts
   - Centers
   - Rights

4. To make shapes the same size:

   a. Choose multiple objects in the Entity Relationship Designer.

      For more information, see *Modifying Shapes in the Entity Relationship Designer*.

   b. Right-click the canvas, click Layout, Make Same Size, and then click one of the following menu items:

      - Width
      - Height
      - Both

Modifying Relationships in the Entity Relationship Designer

This topic describes how to modify relationships in the Entity Relationship Designer. You perform all tasks that this topic describes in the Entity Relationship Designer.

To modify relationships in the Entity Relationship Designer

1. To add a point to a relationship:

   a. Right-click a relationship, click Edit, and then click Add Point.
   b. Grab the point, and then move it to a new position on the canvas.

   You can modify the shape of a relationship. For example, you can add a ninety degree angle. This configuration helps to avoid overlapping lines in a complex diagram.

2. To hide the text labels of a relationship, right-click a relationship, click Edit, and then click Hide Text.

   You can hide the text label of a relationship, including the Relationship Name, End Name 1, and End Name 2.

3. To move the name of a relationship, right-click a relationship, click Edit, and then click Move Text Back or Move Text Forward.

   You can modify where Siebel Tools displays the text label for the name of a relationship. Note that you cannot modify the location of the text labels for the End Name 1 and End Name 2 properties.

4. To return text labels of a relationship to the default setting, right-click a relationship, click Edit, and then click Move Text to Default.
5. To display a connection point, right-click the canvas, and then click Connection Points.

   To hide connection points, choose Connection Points again to remove the check mark.

   A *connection point* is the point on an entity where the relationship connects. You can display or hide a connection point. For example, you can display connection points when you create an ERD, and then hide them when you print an ERD.

Moving Shapes in the Entity Relationship Designer

To move a shape in the Entity Relationship Designer canvas, you can relocate it or use the menu items in the Layout menu.

To move shapes in the Entity Relationship Designer

- In the Entity Relationship Designer, do one of the following:
  - Move the shape to another position on the canvas.
  - Right-click the shape, click Layout, Move, and then click one of the menu items described in the following table.

Menu item	Description
Left by 1	Moves the shape in the direction you choose by 1 pixel.
Right by 1	
Up by 1	
Down by 1	
Left by X	Moves the shape in the direction you choose according to the number of pixels that one cell on the canvas contains. The number of pixels can vary depending on the resolution setting of your monitor.
Right by X	
Up by X	
Down by X	

You can use the shortcut keys that Siebel Tools displays in the Move submenu of the Layout menu.

Resizing Shapes in the Entity Relationship Designer

To resize a shape, you can move it in the canvas or use the Resize menu item in the Layout menu.

To resize shapes in the Entity Relationship Designer

1. In the Entity Relationship Designer, choose the shape you must resize.
2. Do one of the following:
   - Click one of the connection points on the entity, and then move it to a new position.
   - Right-click the entity, click Layout, Resize, and then click one of the menu items described in the following table.

Menu item	Description
Height by 1	Resizes the dimension you choose by 1 pixel.
Height by -1	
Width by 1	
Width by -1	
Height by X	Resizes the dimension you choose according to the number of pixels that one cell of the canvas contains. The number of pixels can vary depending on the resolution setting of your monitor.
Height by -X	
Width by X	
Width by -X	

You can use the shortcut keys in the Expand submenu of the Layout menu.

Zooming, Displaying, and Snapping the Grid

You can zoom, display the grid, or snap an object to the grid in the Entity Relationship Designer.

To zoom, display, and snap the grid

1. To zoom in the Entity Relationship Designer, right-click the canvas, click Zoom, and then click one of the following menu items:
   - Zoom In.
   - Zoom Out.
   - Choose a percentage.

You can choose a default zoom amount or enter a percentage to zoom in and out of an ERD.

2. To display the grid, right-click the canvas, and then click Show Grid.

   To hide the grid, click Show Grid again, which removes the check mark.

   The grid helps you align entities and relationships in an ERD. It is useful to display the grid when you work in the canvas, and then hide it when you print the ERD.
3. To turn on the Snap to Grid feature, right-click the canvas, and then click Snap to Grid.

   The Snap to Grid feature helps you keep entities and relationships aligned while you define your ERD.

12 Configuring Tables

Configuring Tables

This chapter describes tasks you perform to configure tables. It includes the following topics:

- *Using the New Table Wizard to Create a New Table*
- *Creating a Custom Index*
- *Adding an Extension Column to a Base Table*
- *Configuring Objects to Use a One-To-Many Extension Table*
- *Configuring an Extension Table*
- *Applying a Data Layer Customization to the Server Database*
- *Downloading a Data Layer Customization to Remote Users*

For more information, see *Options to Configure the Data Objects Layer*.

Using the New Table Wizard to Create a New Table

The New Table Wizard allows you to create a new stand-alone table, extension table, or intersection table. It includes lists that display choices for each type of table and that makes sure you use the correct naming formats. For more information, see *Guidelines for Creating a New Table*.

To use the New Table Wizard to create a new table

1. In Siebel Tools, click the File menu, and then click New Object.
2. On the General tab of the New Object Wizards dialog box, click Table, and then click OK.
3. In the General dialog box, in the Enter a Name for the New Table field, enter a new table that begins with CX_.

   If you do not enter a name, then the New Table Wizard adds a prefix.

   You must enter the Table Name in uppercase. A mixed case or lowercase name might result in problems if you apply the modifications to some databases.
4. In the Choose a Project in Which You Wish to Create the Table field, choose a project.

   The New Table Wizard restricts the Project list to only locked projects. The wizard restricts all lists that display in the wizard to objects that belong to locked projects.
5. In the Select the Type of the Table field, choose from the following options:
   - A stand-alone Table
   - 1:1 Extension Table for a predefined Table
   - 1:M Extension Table for a predefined Table
   - Intersection Table between two existing Tables

If you choose 1:1 Extension Table for an Existing Table, then the New Table Wizard applies the _X suffix to the table name.

6. Click Next.

   The subsequent dialog box displays depending on the type of table you are adding.
7. If you are creating a stand-alone table, then click OK in the Finish dialog box.
8. If you are creating a one-to-one or many-to-one extension table, then choose the parent table in the Parent Table Specification dialog box, click Next, and then click Finish.

   The wizard restricts the list of available parent tables to Data (Public) tables.
9. If you are creating an intersection table, then do the following:

   a. Add the parent tables and names of foreign key columns to the parent table in the Parent Table Specification dialog box.

      The New Table Wizard restricts the lists for the Select the First Parent Table field and the Select the Second Parent Table field to all Data (Public) tables. The wizard verifies the names of the Foreign Key columns that you enter. This verification makes sure that they are unique and do not conflict with each other or with other system column names.
   b. Click Next.

      Siebel Tools displays the Finish dialog box that allows you to review the modifications before the wizard creates the objects.
   c. Click Finish to create the table.

      Siebel Tools displays the new table in the Tables list. The name is CX_*YOUR_CUSTOM_NAME*_X.
10. Test and then deliver your Workspace.

    For more information, see *Using Siebel Tools* .

Work That the New Table Wizard Performs

If you use the New Table Wizard to create a custom extension table, then this wizard adds a U1 index to the table. The User Key column is empty. Siebel CRM does not support creating a user key on a custom table.

For all tables, the New Table Wizard creates seven system columns and a P1 index on ROW_ID.

For a one-to-one extension table, the wizard sets the Type property to Extension and does the following work:

- Creates a column for PAR_ROW_ID
  - Sets the User Key Sequence property to 1
  - Sets the Foreign Key Table property to *BASE_TABLE_NAME*
- Creates a U1 index that includes PAR_ROW_ID(1) and CONFLICT_ID(2)
  - Sets the Unique and Cluster properties to TRUE
  - Sets the Type property to User Key
  - Sets the User Primary Key property to TRUE

For a many-to-one extension table, the New Table Wizard sets the Type property to Data (Public) and does the following work:

- Creates the following columns:
  - PAR_ROW_ID
  - TYPE
  - NAME
- Creates a U1 index that includes PAR_ROW_ID(1), TYPE (2), NAME (3), and CONFLICT_ID (4)
  - Sets the Unique and Cluster properties to TRUE
  - Sets the Type property to User Key
  - Sets the User Primary Key property to TRUE
- Creates an M1 index on TYPE (1) and NAME (2)
  - Sets the Unique and Cluster properties to FALSE
  - Sets the Type property to System

For an intersection table, the New Table Wizard sets the Type property to Data(Intersection) and does the following work:

- Creates a TYPE column for added user functionality
- Creates two Foreign Key columns using names you defined in the wizard
  - Sets the User Key Sequence property to 1 and 2
  - Sets the Foreign Key Table property to Parent Table
- Creates a U1 index on the two Foreign Keys (1, 2), TYPE (3), and CONFLICT_ID (4)
  - Sets the Unique and Cluster properties to TRUE
  - Sets the Type property to User Key
  - Sets the User Primary Key property to TRUE
- Creates an F1 index on the Foreign Key to the second parent table

Creating a Custom Index

This topic describes how to create a custom index. For more information, see *Guidelines for Creating a Custom Index*.

To create a custom index

1. In Siebel Tools, in the Object Explorer, click Table.
2. In the Tables list, locate the table where you must add an index.
3. In the Object Explorer, expand the Table tree, and then click Index.
4. In the Indexes list, add a new index.

   When you add a custom index to a table, Siebel Tools appends an _X to the index name. Do not use an index name that includes a word that is reserved on your server or client database. For more information, see *Indexes of a Siebel Table* and *Siebel Object Types Reference* .

5. In the Object Explorer, expand the Index tree, and then click Index Column.
6. In the Index Columns list, add a new record for each index column.

   For more information, see *Index Columns of an Index*.

Adding an Extension Column to a Base Table

This topic describes how to add an extension column to a base table. For more information, see *Guidelines for Adding an Extension Column to a Base Table*.

> **CAUTION:** Be extremely careful if you use a custom extension column to track a foreign key. If you use this configuration, then it is recommended that you consult with Oracle concerning the visibility rules that Siebel CRM applies to the foreign key table. To use Enterprise Integration Manager to load values into the column, you must set the Foreign Key Table Name property to NULL for that column. For information on creating a foreign key mapping for Enterprise Integration Manager, see *About Interface Tables*.

To add an extension column to a base table

1. In the Object Explorer, click Table.
2. In the Tables list, choose the table where you must add an extension column.
3. Make sure the Type property for the table is not Data (Private).
4. In the Object Explorer, expand the Table tree, and then click Columns.
5. In the Columns list, add a new record.
6. Apply the modifications to your local database.

Configuring Objects to Use a One-To-Many Extension Table

To use a one-to-many extension table, you must configure the objects that this topic describes.

> **CAUTION:** Do not use a one-to-many extension table as an extension to a predefined one-to-many extension table. This configuration causes problems with Enterprise Integration Manager and docking processes.

To configure objects to use a one-to-many extension table

1. Create a new business component and new fields in the business component that references columns in the one-to-many extension table that you use to store data.
2. Define the following business component fields:

   a. PAR_ROW_ID. References the foreign key field that the one-to-many link uses.
   b. NAME. Makes the record unique for each parent record.
   c. TYPE. Groups records in the extension table.

- Set a default value for the Type field.
- Define the search specification for the business component to search for records in the extension table that contain the default value. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.

  To satisfy the U1 index of the one-to-many extension table, the combination of NAME, TYPE, and PAR_ROW_ID must be unique.

3. Add a link and business object component that creates the parent-child relationship between the new, child business component and the parent business component.

Configuring an Extension Table

This topic describes options to configure an extension table. It includes the following information:

- *Manually Creating a One-to-One Extension Table*
- *Modifying a Custom Extension Table or Column*
- *Deleting a Custom Extension Table or Column*

Manually Creating a One-to-One Extension Table

You can manually create a one-to-one extension table, but it is recommended that you use the New Table Wizard. For more information, see *Guidelines for Creating a Custom One-to-One Extension Table* and *Using the New Table Wizard to Create a New Table*.

To manually create a one-to-one extension table

1. In Siebel Tools, in the Object Explorer, click Table.
2. In the Tables list, locate the base table where you must create an extension table.
3. Verify that the Type property for the table contains Data (Public).
4. Click Extend.

   The Database Extension Designer creates the required predefined columns and predefined indexes, and then Siebel Tools displays the extension table in the Tables list. If necessary, the designer creates temporary columns in an interface table that Siebel Tools imports to the base table for this extension table.
5. Optional. Define more extension columns on the custom extension table.

   For more information, see *Adding an Extension Column to a Base Table*.

Modifying a Custom Extension Table or Column

After you create a custom extension table or column, you can only modify the properties of the table or column. You can rename a column before you apply it to the Siebel Server. After you add the column or apply it to the Siebel Server, you cannot rename the column. Instead, you must deactivate the column and create a replacement extension column.

Be careful if you modify the Physical Type property of a column. Depending on existing data that resides in the column, it might not be possible to do this modification.

Siebel CRM does not support modifying a predefined base table or the columns of a predefined base table. You must not modify the extension tables that come predefined with Siebel CRM. For more information, see *How an Extension Table Stores Custom Data*.

To modify a custom extension table or column

1. Open Siebel Tools.
2. In the Object Explorer, click Table.
3. Optional. Modify a custom extension column:
    a. In the Tables list, locate the table that contains the extension column you must modify.

       If you are adding a new extension table to the EIM Table Mapping list, then make sure you click Activate to create all the temporary columns that Enterprise Integration Manager (EIM) requires.
    b. In the Object Explorer, click Column.
    c. In the Columns list, locate the extension column you must modify, and then modify the properties.
4. Optional. Rename a custom extension column:
    a. In the Tables list, locate the table you must modify, and then deactivate the unwanted column.
    b. In the Tables list, create a new table column.
    c. Export the data from the old column.
    d. Use ddlsync.ksh to synchronize the logical and physical schema and to import the data.
    e. Delete the column you deactivated.
5. Optional. Modify a custom extension table:
    a. In the Tables list, locate the extension table you must modify.
    b. Modify properties, as you require.

Deleting a Custom Extension Table or Column

You can delete from the logical schema a custom extension table or column that you defined. Deleting a table or column removes it from the logical schema in the Siebel repository, but it does not remove it from the physical schema of the Siebel database.

You can only delete a custom extension column or table. You cannot delete a predefined table or the columns of a predefined table.

After you delete an extension table, Siebel Tools does not delete any corresponding temporary columns in an interface table. You cannot use Siebel Tools to delete these columns. The columns will remain in the logical and physical schema.

If a column is empty at the database level, and if the column is in the Siebel repository, and if the column is in the development environment but is not in the production environment, then your database administrator can use a database tool to remove a column that you do not require. The administrator must do a full database backup before removing a column. The administrator must be careful when deleting a column because removing a column that Siebel CRM still references might require the administrator to revert to a full backup.

Using the Siebel Database Configuration Wizard to run the Synchronize Schema Definition (ddlsync) utility does not delete a column that is inactive or that was deleted from the Siebel Repository. For more information, see *Siebel Database Upgrade Guide* .

To delete a predefined extension column

1. In the Object Explorer, click Table.
2. In the Tables list, locate the table that contains the extension column you must delete.
3. In the Object Explorer, expand the Table tree, and then click Column.
4. In the Columns list, locate the extension column you must delete.
5. Click the Edit menu, and then click Delete.

   Siebel Tools does not cascade the deletion of an extension column. You must delete or deactivate the attribute map after you delete an extension column. To delete an attribute map, you can navigate to the Attribute Mappings list in Siebel Tools, and then delete the record.

To delete a predefined extension table

1. In the Object Explorer, click Table.
2. In the Tables list, locate the table you must delete.
3. Click the Edit menu, and then click Delete.

Applying a Data Layer Customization to the Server Database

You must apply your customization to the physical server database. Until you do this, Siebel CRM only updates the logical database schema, as stored in the repository tables of the Siebel database. You can use Siebel Tools or the Database Configuration Utility to apply a customization to the data objects layer.

> **CAUTION:** If a table is marked as Inactive in the Siebel Repository, and if you click Apply/DDL, then Siebel Tools removes the underlying table from the Siebel database.

To apply a data layer customization to the server database

1. Test your customization in the local environment.
2. Prepare the server database:
   a. Make sure all remote users synchronize.
   b. Make sure all connected clients are disconnected from the database server.
   c. After Siebel CRM merges and routes all transactions for remote users, stop all Siebel Servers.
   d. Do a full backup of the server database.
3. Connect to the Siebel Server.
4. Check your projects into to the server database.
5. In the Object Explorer, click Table.
6. In the Tables list, locate the table where you must apply a modification to the Siebel database.
7. In the Tables list, click Apply/DDL.

   Siebel Tools disables the Apply/DDL button for tables that contain External in the Type property. For more information, see *Overview: Siebel Enterprise Application Integration* .
8. In the Choose option dialog box, choose the Apply option, and then click OK.

9. In the Apply Schema dialog box, perform step 7.

   If you receive an error message and cannot apply your customization on the server database, then you must use the Database Server Configuration Utility. For more information, see *Downloading a Data Layer Customization to Remote Users*.
10. In the Apply Schema dialog box, click Apply.
11. In the Tables list, click Activate.

    Siebel Tools increases the version of the custom database schema and prepares the upgrade of the remote client. The customization now exists physically on the server database.
12. Restart the Siebel Server.

    Your customization tables and columns are now available to use in your configuration.

Downloading a Data Layer Customization to Remote Users

After you check in extensions to your server database and apply the physical database, you can download the schema modifications to remote users.

Note: As of Siebel CRM 20.8 Update, Oracle Database XE has been replaced with Oracle Database SE2 for the local database for Siebel Mobile Web Client. For more information, see *Siebel Installation Guide* .

To download a data layer customization to remote users

1. Make sure all remote users perform a full synchronization.
2. If you use Siebel Anywhere, then do the following:
   a. Create an Upgrade Kit on your Server database that includes the Siebel Database Schema as the upgrade kit component.

      For more information, see *Siebel Anywhere Administration Guide* .
   b. Click Activate on the Upgrade Kits View to make the upgrade kit available.
3. If you do not use Siebel Anywhere, then do the following:
   a. Log in to Siebel Tools while connected to the server database.
   b. In the Object Explorer, click Table.
   c. In the Tables list, locate the table that includes your customization.
   d. In the Tables list, click Activate.

      Siebel Tools increases the version of the custom database schema and prepares the upgrade of the remote client.
4. To recreate the template local database, run gennewdb.

   For more information, see *Siebel Remote and Replication Manager Administration Guide* .
5. Reextract remote clients.

   Each remote client must reinitialize the local database with the extracted data. This procedure differs depending on if you use Siebel Anywhere.
6. If you use Siebel Anywhere, then click Distribute in the Upgrade Configurations View.

   This step makes the new custom schema version available for a schema upgrade. You must manually set the Required flag. For more information, see *Siebel Anywhere Administration Guide* .

If you do not use Siebel Anywhere, then manually reextract and reinitialize all remote user databases.

13 Configuring Business Components, Links, and Business Objects

Configuring Business Components, Links, and Business Objects

This chapter describes how to configure business components, links, and business objects. It includes the following topics:

- *Configuring a Business Component*
- *Configuring a Business Component Field*
- *Configuring a Link*
- *Creating a Business Object*
- *Configuring a Searchable Virtual Business Component*

Configuring a Business Component

This chapter describes how to configure a business component. It includes the following information:

- *Creating a New Business Component*
- *Determining How a Business Component Sorts Records*
- *Determining How a Business Component Sequences Records*
- *Defining Read-Only Behavior for a Business Component*
- *Creating a Recursive Join on a Business Component*
- *Configuring a Business Component to Copy Child Records If the User Copies the Parent Record*
- *Allowing the User to Set a Primary Team Member*

Creating a New Business Component

You might need to create a new business component if no predefined business components exist that provide a good functional or technical fit for your business requirements. For example, if you must predefault record values to a different type as a way to differentiate some records from other records in the same table. For more information, see *Reusing Predefined Objects*.

To create a new business component

1. Make sure you cannot use a predefined business component.
   For more information, see *Determining Whether You Can Reuse a Predefined Business Component*.
2. In Siebel Tools, click the File menu, and then click New Object.
3. In the General tab, choose BusComp, and then click OK.

4. In the New Business Component Wizard, choose a project, enter a name for the business component, and then click Next.
5. In the Single Value Fields dialog box, choose a column in the Base table, and then enter a name for the field.
6. Click Add, and then click Finish.

   Siebel Tools displays the business component you just created in the Business Components list.
7. In the Business Components list, define the properties to meet your requirements.

For more information, see *Properties of a Business Component*.

Determining How a Business Component Sorts Records

You can create a sort specification on a business component to determine how a business component sorts records. For more information, see *How a Business Component Sorts Records*.

To determine how a business component sorts records

1. In Siebel Tools, in the Object Explorer, click Business Component.
2. In the Business Components list, locate the business component you must modify.
3. In the Sort Specification property, enter a sort specification, and then save your modifications.

   You must use a specific format. For more information, see *Guidelines for Configuring How a Business Component Sorts Records*.
4. Test and then deliver your Workspace.

For more information, see *Using Siebel Tools* .

Determining How a Business Component Sequences Records

Sequential numbering is not predefined in any system column in a predefined table in Siebel CRM. You can configure a sequence field in a child business component to determine how a business component sequences records. For more information, see *How a Business Component Field Sequences Records*.

To determine how a business component sequences records

1. Verify that the class of the child business component is CSSBCBase or a subclass of CSSBCBase.

   If it is not, then contact Oracle Global Customer Support for assistance with this procedure. For more information, see *Getting Help From Oracle*.
2. Verify that the business component where you must add a sequence field is the child business component in a parent-child relationship.

   This child is the numbered business component. Siebel CRM numbers child records beginning with 1 in each parent record.
3. Add a child field to the numbered business component using values from the following table.

Property	Value
Name	Enter text that identifies the data that Siebel CRM sorts, such as Line Number or Order.

Property	Value
Column	Enter a numeric extension column, such as ATTRIB_14.
Type	DTYPE_NUMBER

4. Add a child business component user property to the numbered business component using values from the following table.

Property	Value
Name	Sequence Field
Value	Enter the field name you defined in Step 3.

5. Create a business component using values from the following table.

Property	Value
Class	CSSSequence
table	Enter the name of the base table of the numbered business component.
Name	Enter a name using the following format: *name of the numbered business component . name of the sequence value field* (Sequence)

6. Set the Sort Spec of the business component you created in Step 5 to Sequence (DESCENDING).
7. Add a child field to the sequence business component using values from the following table.

Property	Value
Name	Sequence
Column	Enter the same value you entered for the column in Step 3.

8. Add a child field to the sequence business component.

   This field is the foreign key field that creates the parent-child relationship to the parent business component. Set the Column property to the same column as the corresponding field in the numbered business component.

9. Create a link that creates a parent-child relationship between the parent and sequence business components.

   For more information, see *About Links*.
10. Create a child business object component of the business object that uses the predefined link that exists between the parent business component and the numbered business component. Use values from the following table.

Property	Value
Link	Choose the link you defined in Step 9.
BusinessComp	Choose the sequence business component.

11. Display the sequence value field in applets that display records from the numbered business component.
12. Test and then deliver your Workspace.

    For more information, see *Using Siebel Tools* .

Defining Read-Only Behavior for a Business Component

In this example, if an account record includes a competitor, then the user must not choose any competitors for the account. If the Type field includes a value of Competitor, then Siebel CRM makes the Competitor field in the account record read-only. For more information, see *How Siebel CRM Defines Read-Only Behavior for a Business Component Field*.

To define read-only behavior for a business component

1. In the Object Explorer, click Business Component.
2. In the Business Components list, locate the Account business component.
3. In the Object Explorer, expand the Business Component tree, and then click Field.
4. In the Fields list, create a field using values from the following table.

Property	Value
Name	Competitor Calculation You can use any name.
Calculated	TRUE
Calculated Value	Iif([Type] = "Competitor", "Y", "N")

5. In the Object Explorer, click Business Component User Prop.
6. In the Business Component User Props list, add a new record using values from the following table.

Property	Value
Name	Field Read Only Field: Competitor
Value	Competitor Calculation

When you create a business component or field, make sure the values in the Name and Value properties use the correct capitalization, spelling, and empty spaces. Make sure that quotation marks are not present.

7. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Creating a Recursive Join on a Business Component

If you create a recursive join, then you must make sure the Alias name of the join is different from the Table Name. If you use the same name with an inner join, then Siebel Tools displays the following error message:

```
Table 'T1' requires a unique correlation name.
```

If you use the same name with an outer join, then Siebel Tools displays the following error message:

```
Table 'T1' is in an outer join cycle.
```

An *inner join* is a join that contains an Outer Join Flag property that does not contain a check mark. An *outer join* is a join that contains an Outer Join Flag property that does contain a check mark.

To create a recursive join on a business component

- Make sure the Alias name of the join is different from the Table Name.

Configuring a Business Component to Copy Child Records If the User Copies the Parent Record

Cascade copy is a feature on a business component that copies the child records of a business component record if the user copies a parent record. For example, if the user copies an opportunity to create a similar opportunity, then the user might require Siebel CRM to copy the list of contacts for that opportunity.

A multi-value link that Siebel CRM uses with a multi-value field copies the child records because the child records that constitute a multi-value group remain with the parent record. For example, the child records for account addresses, sales teams, and industry lists of a parent account remain with the account. Siebel CRM uses this capability for a different purpose if cascade copy is defined for a multi-value link, and if Siebel CRM does not use the multi-value link in a multi-value field. It is not necessary to reference the multi-value link to a field in the business component. For more information, see *How Siebel CRM Creates a Multi-Value Group*.

You can define cascade copy for a many-to-many relationship where the Inter Table property of the destination link is not empty. In this situation, Siebel CRM creates new intersection table rows rather than new child business component

records. It creates new associations rather than new records. These associations exist between the new parent and the existing child records.

Cascade copy might cause the values in an index to not remain unique. If copying child records causes an index to not remain unique, then Siebel CRM cancels the copy operation.

To create a business component to copy child records if the user copies the parent record

- Create a multi-value link using values from the following table.

Property	Value
Destination Link	The name of the link where Siebel CRM defines the parent-child relationship.
Destination Business Component	The name of the child business component.
No Copy	FALSE If the No Copy property is TRUE, then Siebel CRM disables cascade copy. An exception to this configuration occurs if the corresponding field is defined as the destination field in a link. In this situation, the link enters data into the field and ignores the value of the No Copy property.

Allowing the User to Set a Primary Team Member

You can allow the user to set a primary team member.

To allow the user to set a primary team member

1. In the Object Explorer, click Business Component.
2. In the Business Components list, locate the business component that the multi-value group applet references.

   For more information, see *Creating Multi-Value Groups and Multi-Value Group Applets*.
3. In the Object Explorer, expand the Business Component tree, and then click Business Component User Prop.
4. In the Business Component User Props list, locate the following business component user property:

   MVG Set Primary Restricted:*name of the multi-value link*
5. Set the Value property to FALSE.

   This configuration allows some users to set a primary. Setting this user property to FALSE allows someone other than the Manager or Siebel Administrator to modify the Primary team member. If this user property is not set, then only a Siebel Administrator working in Admin mode or a Manager working in Manager view mode can modify the Primary team member on an opportunity, account, or contact. For more information about user properties, see *Siebel Developer's Reference* .

6. Test and then deliver your Workspace.
   For more information, see *Using Siebel Tools* .

Configuring a Business Component Field

This topic describes how to configure a business component field. It includes the following information:

- *Creating a New Business Component Field*
- *Activating a Multi-Value Field*
- *Validating Data That the User Enters in a Business Component Field*
- *Creating a Business Component Field That Displays More Than One Currency*
- *Configuring Client-Side Import to Update a Business Component Field*
- *Creating a Joined Business Component Field*
- *Creating a Predefault Value for a Joined Business Component Field*

Creating a New Business Component Field

You can create a new business component field.

To create a new business component field

1. Make sure you cannot reuse a business component field.
   Before you create a new business component field, make sure a predefined field that meets your business requirements does not exist. For more information, see the following topics:
   - *Determining Whether You Can Reuse a Predefined Business Component Field*
   - *Guidelines for Creating a Business Component*
2. In Siebel Tools, in the Object Explorer, click Business Component.
3. In the Business Components list, locate the business component where you must add a field.
4. In the Object Explorer, expand the Business Component tree, and then click Field.
5. In the Fields list, add a new record, and then define properties for the new record.
   You must not map multiple business component fields to the same column in a table. If you do this, then the SQL query fails because it attempts to access the same column twice in the same query. This configuration might cause an error message when Siebel CRM updates data, can cause problems with data integrity, and can lead to data loss for denormalized columns that reference the column.
6. Test and then deliver your Workspace.
   For more information, see *Using Siebel Tools* .

Activating a Multi-Value Field

To refresh correctly in the master applet when the user closes the multi-value group applet, you must make sure the multi-value field is activated at the business component level. For example, if the Account Entry Applet displays

the Active Login Name multi-value field through the Position multi-value link, then the Force Active property of the Active Login Name field in the Position business component must equal TRUE, or the Active Login Name field must be included in the Position MVG applet that Siebel CRM uses to maintain the account Sales Team. This situation is true even if the field is not visible. For more information, see *About the Multi-Value Field* and *Creating Multi-Value Groups and Multi-Value Group Applets*.

To activate a multi-value field

1. Do one of the following:
   - Set the Force Active property of the multi-value field to True.
   - Include the multi-value field in the multi-value group applet.
2. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Validating Data That the User Enters in a Business Component Field

You can configure Siebel CRM to validate the information that the user enters into a field. You can configure the error message that Siebel CRM displays if the user enters information that does not meet the validation expression.

To validate data that the user enters in a business component field

1. Make sure Siebel Tools is configured to allow you to modify a text string.

   For more information, see *Setting Up the Configuration File for Siebel Tools*.
2. Create the symbolic strings that Siebel CRM displays in the error message.
3. In the Object Explorer, click Business Component.
4. In the Business Components list, locate the business component that contains the field that Siebel CRM must validate.
5. In the Object Explorer, click Field.
6. In the Fields list, locate the field that Siebel CRM must validate.
7. In the Validation property, enter an expression that does the validation.

   You can use the Expression Builder to build the expression. To display the Expression Builder, click the ellipsis (...) in the Validation property.
8. In the Validation Error Message - String Reference property, enter the name of the symbolic string you created in step 2 for this error message.

   As an alternative, you can use the Validation Message property to enter the error message without using a symbolic string. You can use the Validation Message - String Override property to override the error message.
9. In the Message Display Mode property, choose one of the following display modes for the error message:
   - **User Msg.** Displays only the error message that you provide.
   - **User Msg with Error Code Only.** Displays the error message that you provide and the system error code.
   - **User Msg with Error Code/Msg.** Displays the error message that you provide, the system error code, and the system error message.
10. Test and then deliver your Workspace.

For more information, see *Using Siebel Tools* .

How Siebel CRM Validates Start and End Dates

If the user sets the completion date for an activity to occur before the start date for the activity, then Siebel CRM displays an error that is similar to the following:

```
Wrong field values or value types detected in field End.
```

Siebel CRM ignores any configuration you define in the Validation property or the Validation Message property for these date fields. For example, assume the user navigates to the Activities screen, clicks Activity List, and then sets the value in the Start field to a date that is later than the value in the End field. This Start field references the Planned field of the Action business component. If you define a value in the Validation property or the Validation Message property for the Planned field, then Siebel CRM ignores it.

You cannot configure the predefined validation in the classes, but you can add script in the BusComp_PreSetFieldValue event for the business component. This script monitors updates to these fields, and then compares the field values. You can write a custom error message in this script. For more information, see the topic about the CSSBCActivity class in *Siebel Developer's Reference* .

Creating a Business Component Field That Displays More Than One Currency

You can configure a field to display data in more than one currency. For example, assume a global deployment occurs in the United States and in Japan, where the deployment in the United States requires data to display in dollars and the deployment in Japan requires data to display in yens.

You cannot configure a Forecast business component to display dual currency because the list columns that display monetary values do not reference fields. The list columns display values that reference buttons that use specialized methods to perform the calculation. For more information, see *Caution About Using Specialized Classes*.

To create a business component field to display more than one currency

1. In Siebel Tools, in the Object Explorer, click Business Component.
2. In the Business Components list, locate the Opportunity business component.
3. In the Object Explorer, expand the Business Component tree, and then click Field.
4. In the Fields list, create a new field using values from the following table.

Property	Value
Name	My_Currency
Type	DTYPE_TEXT
Join	S_OPTY_X
Column	ATTRIB_03

Property	Value
PickList	PickList Currency

Siebel Tools stores the field in an unused column in the S_OPTY_X extension table.

5. In the Object Explorer, expand the Field tree, and then click Pick Map.
6. In the Pick Maps list, create a new pick map using values from the following table.

Property	Value
Field	My_Currency
Pick List Field	Currency Code

7. In the Object Explorer, click Field, and then add a field to the Opportunity business component for the converted revenue using values from the following table.

Property	Value
Name	My_Cvt_Revenue
Calculated	TRUE
Calculated Value	[Revenue]
Currency Code Field	My_Currency The Currency Code Field property references the currency code field of *Creating a Business Component Field That Displays More Than One Currency*.
Exchange Date Field	Sales Stage Date
Type	DTYPE_CURRENCY

CAUTION: Make sure the Exchange Date Field property on the originating currency field is defined in a way that is similar to the converted currency field. If it is not, then Siebel CRM bases the exchange date that it uses to convert the currency on the exchange date of the originating currency field.

For more information, see *Requirements for the Field That Contains the Converted Currency Amount*.

8. In the Object Explorer, click Applet.
9. In the Applets list, locate the Opportunity List Applet.
10. In the Object Explorer, expand the Applet tree, expand the List tree, and then click List Column.
11. In the List Columns list, add a list column using values from the following table.

Property	Value
Field	My_Currency
Display Name	Converted Currency Code

12. In the List Columns list, add a list column using values from the following table.

Property	Value
Field	My_Cvt_Revenue
Display Name	Converted Revenue
Runtime	TRUE

    It is not necessary to create a pick or detail applet because Siebel CRM opens the default applet that matches the field type.
13. Test and then deliver your Workspace.

    For more information, see *Using Siebel Tools* .
14. Make sure the underlying currency business component contains a minimum number of valid values:
    a. In the Siebel client, navigate to the Administration - Application screen, and then the Currencies view.

       This view lists currencies, conversion dates, and exchange rates.
    b. Make sure each currency that is involved in the conversion is marked as active.
    c. Make sure at least one exchange rate is defined for each currency that is involved in the conversion.
    d. Make sure at least one of the exchange rates for an exchange direction includes a date that occurs at or before the date that Siebel CRM uses as the Exchange Date.
15. Test your modifications.

Requirements for the Field That Contains the Converted Currency Amount

The field in the business component that contains the converted currency amount must meet the following requirements:

- The Type property of the field must equal DTYPE_CURRENCY.
- The field must be a calculated field.

- The Type property of the field that the Calculated Value property references must equal DTYPE_CURRENCY. For example, if the expression in the Calculated Value property is `[Revenue]`, then the Type property of the Revenue field must equal DTYPE_CURRENCY.
- The Exchange Date Field property must reference a field that contains a Type property that is DTYPE_DATETIME.

Configuring Client-Side Import to Update a Business Component Field

You can use client-side import to update a business component field. Client-side import uses the import functionality of the applet menu in the Siebel client. You use the object type known as the Import Object in Siebel Tools to identify the business component fields into which Siebel CRM enters data.

For an example of how Siebel CRM configures client-side import, in Siebel Tools you can examine the predefined import object that is defined for the Contact business component. The Contact business component is defined as an Import Object and it contains fields that Siebel CRM defines as Import Field objects.

You cannot use client-side import with a specialized business component or specialized applet. For more information, see *Class Property of a Business Component* and *Siebel Object Types Reference* .

To configure client-side import to update a business component field

1. Display the object type known as the Import Object.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. Lock the project that the business component references.
3. In the Object Explorer, click Import Object.
4. In the Import Objects list, add a new record using values from the following table.

Property	Value
Business Component	Choose the business component into which Siebel CRM must import data. Make sure this business component is a parent business component. Siebel CRM only supports client-side import for a parent business component.

5. In the Object Explorer, expand the Import Object tree, and then click Import Field.
6. In the Import Fields list, add a new record for each business component field that Siebel CRM must update.

   Note that you can add an import field to an import object that already exists, such as Contact.
7. Make sure the *No Insert* property of the applet that is defined for client-side import is *False*.
8. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Siebel CRM displays the new fields in the Select a Siebel Field dialog box. You can map them to fields in the External Data Source Field dialog box when you import data.

Creating a Joined Business Component Field

You can add a join to a business component, and then reference the join in a field.

To create a joined business component field

1. In Siebel Tools, in the Object Explorer, click Business Component.
2. In the Business Components list, locate the business component where you must add a join.
3. In the Object Explorer, expand the Business Component tree, and then click Join.
4. In the Joins list, add a new record, using values from the following table.

Property	Value
Table	Name of the joined table. For example, enter **S_ADDR_ORG** to access address data.
Alias	Name of the join. For example: **Contact - S_ADDR_ORG** It is recommended that you define the alias so that it is different from the table.
Outer Join Flag	If you must get all the records in the business component even if the joined fields are empty, then set Outer Join Flag to TRUE.
Comments	Optional.

5. In the Object Explorer, expand the Join tree, and then click Join Specification.
6. In the Join Specifications list, add a new record, using values from the following table.

Property	Value
Name	Name of the join specification. For example, Primary Address Id.
Destination Column	Primary key column in the joined table. For example, ROW_ID. If you create a join on a column other than ROW_ID, then you must enter a value in the Destination Column property. An empty value in the Destination Column property indicates that the destination column is ROW_ID, which is typically the primary key. For a join to a party table, the destination column must reference the PAR_ROW_ID column in the joined table.
Source Field	Foreign key field in the business component. For example, Primary Address Id.

Property	Value
	If empty, then the Source Field references the Id field, which indicates a one-to-one relationship between the business component and the joined table.

7. Optional. Add a Join Constraint:
   a. In the Object Explorer, expand the Join Specification tree, and then click Join Constraint.
   b. In the Join Constraints list, add a new record, using values from the following table.

Property	Value
Name	Name of the join constraint. For example, Primary Address Id.
Destination Column	Column in the joined table where you must apply a search specification. For example, OU_ID.
Value	The search specification. For example: `GetProfileAttr("Primary Address Id")` For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.

8. In the Object Explorer, click the Field object type in the Business Component tree.
9. In the Fields list, add a new record, using values from the following table.

Property	Value
Name	Name of the joined field.
Join	Join alias for the table where this field gets data. For example, Primary Account Address.
Column	Column in the joined table where this join gets data. For example, ADDR_NAME.
Text Length	Same length as the column where this join gets data.
Type	Data type that is compatible with the column where this join gets data. For example, DTYPE_TEXT for a Varchar column.

10. Test and then deliver your Workspace.

    For more information, see *Using Siebel Tools* .

Creating a Predefault Value for a Joined Business Component Field

To make sure a field contains a value when Siebel CRM inserts a new record, you can use a predefault value for a joined field. Siebel CRM cannot update a joined field. If a field does not include a value when Siebel CRM inserts a record, then you cannot use a predefault value as a default field value. This topic includes an example that creates a predefault value for a joined business component field.

To create a predefault value for a joined business component field

1. Create a join on the S_OPTY table in the Opportunity Product business component.
2. Define two new fields that reference the join.

   One field displays the Opportunity Sales Stage. The other field displays the Name.
3. Add the two fields to the Opportunity Product applet.
4. Test and then deliver your Workspace.

   To test your modifications:

   a. In the Siebel client, use the predefined Opportunities - Products view to add a new product for an opportunity.

      Siebel CRM does not update the joined fields. Oppty Id contains the data that provides the source field for the join.
   b. Requery the applet.

      Note that Siebel CRM now displays the values.
5. In Siebel Tools, set the Predefault property of the Opportunity Name field using values from the following table.

Property	Value
Pre Default Value	Parent:'Opportunity.Name'

   You must use the following format:

   ```
   Parent:'Parent Business Component.Name of the Joined Field'
   ```
6. In Siebel Tools, set the Predefault property of the Opportunity Sales Stage field using values from the following table.

Property	Value
Pre Default Value	Parent: 'Opportunity.Sales Stage'

7. Set the Link Specification property of the Name and Sales Stage fields in the parent business component to TRUE.

8. Test and then deliver your Workspace.
9. Restart the Siebel client.
10. In the Siebel client, add a new product for an Opportunity.

    Note that Siebel CRM immediately enters data into the joined fields.

Configuring a Link

This chapter describes how to configure a link. It includes the following information:

- *Configuring a Link That Deletes Child Records If the User Deletes the Parent Record*
- *Configuring a Link That Creates a One-to-Many Relationship*
- *Configuring Two Links That Create a Many-to-Many Relationship*
- *Creating an Association Between One Parent and Multiple Child Records*

Configuring a Link That Deletes Child Records If the User Deletes the Parent Record

The Cascade Delete property of a link determines if Siebel CRM deletes a child record if the user deletes the parent record.

To configure a link that deletes child records if the user deletes the parent record

- Set the Cascade Delete property on the link using values in the following table.

Value	Description
Delete	If the user deletes the parent record, then Siebel CRM deletes all child records. Use Delete to delete values that Siebel CRM stores in a one-to-many extension table where the only related record is the parent record. Do not use Delete if the child business component in this link is a child business component in another link. In this situation, use CLEAR instead.
Clear	If Siebel CRM deletes the parent record, then it removes the foreign key reference and clears the value in the foreign key column. Use this setting if Siebel CRM might share a child record with another parent.
None	If Siebel CRM deletes the parent record, then it does not delete any records and does not clear the foreign key column. The default setting is None.

Guidelines for Using Cascade Delete

CAUTION: Be careful. If set incorrectly, the Cascade Delete property might cause data integrity problems or orphaned records.

If you use the Cascade Delete property, then use the following guidelines:

- Cascade Delete is not available for a many-to-many link. A child might be the child of more than one parent when Siebel CRM uses a many-to-many link, so Siebel CRM deletes the intersection record but leaves the child record intact.
- If you delete a record that a foreign key of another table references, then Siebel CRM might or might not delete the reference to the record. If it does not delete the reference, then row IDs might reference records that do not exist. If used with a multi-value group, then Siebel CRM might convert the foreign key to display No Match Row Id.
- The link applies to parent child relationships. Siebel CRM treats a one-to-one extension table as an extension of the parent, so it keeps the extension table synchronized with the parent.
- Use a link except for a one to many extension table that involves two different business components.
- To involve grandchild records, use the Deep Delete business component user property. For more information, see *Siebel Developer's Reference* .

Configuring a Link That Creates a One-to-Many Relationship

You can configure a link that creates a one-to-many relationship. The Account/Account Note link is an example of this type of link. One Account can include many note records. The ID is the source field of the parent Account record. Siebel CRM stores it in the Account Id field of the Note record, which is the destination field.

To configure a link that creates a one-to-many relationship

1. Define the child business component.
2. Define the destination field on the child business component.
3. Define the parent business component.
4. Define the source field on the parent business component.

   If you do not define the source field, then Siebel CRM defaults to the ID of the parent business component.

Configuring Two Links That Create a Many-to-Many Relationship

Siebel CRM uses two links with opposite parent-child settings to create a many-to-many relationship that reference an intersection table. The Opportunity/Account link is an example of this type of link, where:

- The intersection table is S_OPTY_ORG.
- The Inter Child Column is OU_ID, which is the ID in the Account business component.
- The Inter Parent Column is OPTY_ID, which is the ID in the Opportunity business component.

For more information, see *How an Intersection Table Defines a Many-To-Many Relationship*.

To configure two links that create a many-to-many relationship

- Set the Inter Table, Inter Parent Column, and Inter Child Column properties of the two links to create the connection between the links and the intersection table.

Creating an Association Between One Parent and Multiple Child Records

If you create a link and an intersection table that creates a many-to-many relationship between a parent business component and a child business component, then Siebel CRM can only associate two business component records at one time even if the unique keys in the intersection table allow multiple associations. The link between the two business components only considers the ROW_ID values of the parent and child records that Siebel CRM requires to maintain the many-to-many relationship. This behavior is expected.

An *intersection business component* is a type of business component that references an intersection table and a one-to-many link between the parent business component and the intersection business component. You can use it to create multiple associations between one parent and multiple child records. The child list applet or multi-value group applet references the intersection business component. To choose the child record, the user accesses a pick applet that references the child business component instead of using an association applet.

For more information, see *How an Intersection Table Defines a Many-To-Many Relationship*.

To create an association between one parent and multiple child records

1. In Siebel Tools, in the Object Explorer, click Business Component.
2. In the Business Components list, query the Name property for Account.

   This example describes how to configure an intersection business component with the Account business component.
3. In the Object Explorer, expand the Business Component tree, and then click Field.
4. In the Fields list, add a new field using values from the following table.

Property	Value
Name	Add any value.
Column	ACCNT_NAME This property must reference a denormalized column that resides in the intersection business component.
Pre Default Value	Parent: 'Account.Name'

For more information about how Siebel CRM uses this configuration, see *How Siebel CRM Uses Denormalized Columns*.

How Siebel CRM Uses Denormalized Columns

Siebel CRM comes predefined with denormalized columns that reside in intersection tables that affect visibility. The user can use a shuttle applet to create an association in a view that affects this visibility. For example, the user can navigate to the Accounts screen, and then use the Account Team view in the Accounts List to add a record in the Account Sales Team View. In this example, Siebel CRM creates an intersection record in the S_ACCNT_POSTN intersection table, and then populates the denormalized columns. It uses the Account/Position many-to-may link to create a many-to-may relationship between the parent Account business component and the child Position business component. The Account/Position link uses the S_ACCNT_POSTN table as the intersection table. The following table describes the properties of the ACCNT_NAME column that Siebel CRM uses in the S_ACCNT_POSTN table.

Property	Value
Name	ACCNT_NAME
Type	Denormalized
Denormalization Path	[OU_EXT_ID].[NAME]

Assume you use the Account business component as the parent, you add a custom child Account Position business component that references the S_ACCNT_POSTN intersection table, and you use a a one-to-many link. If you do this, then Siebel CRM does not populate the denormalized columns. It populates them only if it also automatically creates the intersection record for the many-to-many relationship. If the user manually creates a one-to-many relationship, then Siebel CRM does not populate the denormalized columns. Instead, you must add a field that references a denormalized column in the intersection business component and use a predefault value for this field.

Adding Fields That Reference Denormalized Columns

The example in this topic describes how to add the Account Name field so that it references the ACCNT_NAME denormalized column.

To add fields that reference denormalized columns

1. In Siebel Tools, click Business Component in the Object Explorer.
2. In the Business Components list, locate the intersection business component that you must modify.
3. In the Object Explorer, expand the Business Component tree, and then click Field.
4. In the Fields list, create a new record using values from the following table.

Property	Description
Name	Account Name
Column	ACCNT_NAME
Predefault Value	Parent: 'Account.Name'

Creating a Business Object

This topic describes how to create a business object. For more information, see *Guidelines for Creating a Business Object*.

To create a business object

1. In Siebel Tools, click Business Object in the Object Explorer.
2. In the Business Objects list, create a new record using values from the following table.

Property	Description
Name	Enter a name for the business object that is unique among business objects in the Siebel repository. Siebel CRM uses the name to reference the business object.
Query List Business Component	The default value is Query List. It identifies the business component that stores predefined queries for the business object.
Primary Business Component	You cannot define this property until after you define the business object components.

3. In the Object Explorer, expand the Business Object tree, and then click Business Object Component.
4. In the Business Object Components list, create a new record using values from the following table.

Property	Description
Bus Comp	Choose the business component that the business object references.
Link	Optional. Create a link relationship between two business components.

5. Repeat step 4 for each business component that you must reference in the business object. You must define each of the following business components as a business object component:
    - Any business component whose data displays in an applet on a view that references the business object
    - Any business component whose data Siebel CRM exports in a report from a view that references the business object
6. Define the Primary Business Component property, as described in step 2.

Configuring a Searchable Virtual Business Component

You can configure a Virtual Business Component (VBC) based on a new class to enable search in any pop-up, list, or form applet. Using the class simplifies querying external data using Siebel workflow and integration objects instead of writing a script for the query method. For more information about Virtual Business Components, see *Integration Platform Technologies: Siebel Enterprise Application Integration* . This topic describes how to configure a VBC to search data in a pop-up, list, or form applet. It includes the following information:

- *Creating a New Class Object*
- *Creating a New Virtual Business Component*
- *Creating a New Business Component User Property*
- *Creating a New Applet*
- *Creating a Workflow*

Note: You cannot use a Searchable Virtual Business Component (SVBC) for a Hierarchical List Applet.

Creating a New Class Object

You can create a new class object for a VBC.

To create a new class object

1. Log in to Siebel Tools or Web Tools.
2. Create or open a workspace and go to the Object Explorer.
3. In the Object Explorer, click Class.
4. In the Classes list, add a new record using values from the following table.

Property	Value
Name	CSSSearchableBCVRec
Dll	Set the DLL to SSCABCBC
Object Type	Set the object type to Business Component
Super Class	Set the super class to CSSBCVRec
Project	Choose the required project name

Creating a New Business Component

You can create a new virtual business component for the newly created class object.

To create a new virtual business component

1. In the Object Explorer, click Business Component.
2. In the Business Components list, create a new record using values from the following table.

Property	Value
Name	Enter the name for the business component
Class	CSSSearchableBCVRec

3. Add the required fields to the created Business Component.
4. Optional. For each field, add a new field user property using values from the following table.

Property	Value
Name	IC Field Name
Value	Enter the name of the Integration Component field from which you need to populate this Business Component field.

Note: If a new field user property is not added, then, by default, the Business Component field takes the value from Integration Component field which has the same name. For example, if the Business Component field is “Comments”, then it will use the value obtained from the Integration Component field “Comments”.

Creating a New Business Component User Property

You can create a new business component user property for a workflow.

To create a new business component user property

1. In the Object Explorer, click Business Component User Prop.

2. In the Business Component User Props list, add a new record using values from the following table.

Property	Value
Name	Source Workflow
Value	Enter the name of the source workflow which will be used to populate the Virtual Business Component.

Optional. If you need a different name for the process property, add another business component user property using values from the following table.

Property	Value
Name	Response Property
Value	Enter the name of the custom process property. For example, the workflow can have an output process property of Type "Hierarchy" with name "ResponsePayload" and the property can contain an Integration Object structure.

3. Use the values from the following table to create a new business component user property for the workflow that can be invoked on clicking Submit or OK.

Property	Value
Name	Submit Workflow
Value	Enter the name of the submit workflow

Note: This business component user property supports post processing capabilities.

Creating a New Applet

You must create a new Applet for the newly created Business Component and configure the required user properties to display the Applet.

Optionally, you can also add a new control for the Submit button using values from the following table.

Property	Value
Name	Submit
Caption	Submit
Method Invoked	Submit

Also, if you want the Applet to support multi-select capability, you can add the following Applet user property.

Property	Value
Name	Multi Row Select Check box Display
Value	NONTOUCH-SHOW

For more information on creating Applet, configuring Controls, and defining Applet user properties, see *Configuring Applets*.

Creating a Workflow

You can create a workflow to enable the Virtual Business Component. This topic provides details of sample workflow that can be customized as per your requirements.

- **Source Workflow**. Workflow used to populate the Virtual Business Component. The following table shows the Process Property that should be added.

Property	Value
Name	ResponsePayload
Data Type	Hierarchy

  This process property must contain an Integration Object. The contents in the primary Integration Component will be used to populate the Fields. If any other Integration Components are available, then they will not be considered. There is no specific Type check to validate the Integration Component. The process will pick the first Integration Component found in the hierarchy.

  > **Note:** If a different name is used for the process property, add the respective user property name to the Business Component. For example, if the custom user property value is “Response Property” in the Business Component, then use this value.

- **Submit Workflow**. Workflow invoked on clicking Submit. The following table shows the process property that should be added.

Property	Value
Name	InputIO
Data Type	Hierarchy

This process property will contain the details of selected records in the Virtual Business Component. All the fields and its corresponding values will be available in this process property.

14 Configuring Views, Screens, and Applications

Configuring Views, Screens, and Applications

This chapter describes tasks you perform to configure views, screens, and applications. It includes the following topics:

- *Process of Creating a View*
- *Configuring a View*
- *Process of Creating a Screen*
- *Process of Creating a Screen Home Page View*
- *Creating and Deploying an Application*

Process of Creating a View

To create a view, do the following tasks:

1. *Creating a View*
2. *Editing the Layout of a View*
3. *Registering and Associating a View with a Responsibility*

Creating a View

This task is a step in *Process of Creating a View*.

You typically create a new view to display a new business component, business object, or applet. The New View Wizard assists you with creating a view.

> **Note:** Views and their relationship to Responsibilities can be Workspace enabled in your development environment. If they have been Workspace enabled in your development environment and you are working in that environment, then you can only modify them in an editable Workspace.

To create a view

1. Make sure the ClientConfigurationMode parameter is not set to All.

   For more information, see *Setting Up the Configuration File for Siebel Tools*.
2. In Siebel Tools, click the File menu, and then click New Object.
3. In the New Object Wizard, in the General Tab, click View, and then click OK.
4. In the New View dialog box, do the following, and then click Next:
   a. Choose the project.
   b. Enter a unique name for the new view.

c. Choose the business object whose data the view displays.
d. Enter the title for the view.

5. In the View Web Layout - Select Template dialog box, choose the template you must use for your new view, and then click Next.

   For more information, see *About Siebel Web Templates* and *Siebel Developer's Reference* .
6. In the Web Layout - Applets dialog box, choose the applets that Siebel Tools must include in the web layout, and then click Next.
7. In the Finish dialog box, review your choices, and then click Finish.

   Siebel Tools displays the Web Layout Editor. It allows you to edit the view layout if necessary. For more information, see *Editing the Layout of a View*.

Editing the Layout of a View

This task is a step in *Process of Creating a View*.

You edit the layout of a view in the View Web Template Editor. This editor allows you to edit the mapping between applets in the view and placeholders in the template.

Note: The feature described in this topic is available in Siebel CRM 18.10 Update and later releases.

Note: Views and their relationship to Responsibilities can be Workspace enabled in your development environment. If they have been Workspace enabled in your development environment and you are working in that environment, then you can only modify them in an editable Workspace.

To edit the layout of a view

1. In Web Tools, create or open a workspace and go to the Object Explorer.
2. Click View and search for a view where you want to add a new applet.
3. Locate the view web template you must modify in the View Web Template list and then click the preview button.

   If a template is associated with the view, then Web Tools displays the View Web Template Editor. The View Web Template Editor displays mapped and unmapped placeholders from the underlying view web template.

   If Web Tools does not display the applets properly in the editor, then exit the editor and make sure that the Applet Mode property for each view web template item includes a valid value. To do this, open the pick applet for the property of each applet. Open the editor again to make sure the applets display properly.

4. Optional. Do the following:
    - To add an applet to the View Web Template, select an applet from the Applets window and, with the mouse button depressed, move the selected applet onto an applet placeholder in the template then release the mouse button.
    - To delete an applet from the View Web Template, click the applet, and then press the Delete button at the far end of the applet.
    - To preview the view, select an application from the Applications drop-down list.

      In the preview pop-up window, Web Tools simulates how Siebel CRM displays the view in the Siebel client, which is configured using the application's user property. For more information on configuring the view, see *Using Siebel Tools* .
5. Deliver your modifications from Workspace Dashboard in the application.

Registering and Associating a View with a Responsibility

This task is a step in *Process of Creating a View*.

Registering a view adds a new record for the view in the Siebel database and associates a responsibility with the view. The user who is assigned this responsibility can access the new view the next time the user logs in to Siebel CRM. In a development environment, a developer typically registers the view. In a production environment, the administrator typically registers the view.

If you define a view but do not provide the user access to the view, then Siebel CRM does not display the view in the Siebel client for that user.

To register and associate a view with a Responsibility

> **Note:** Responsibilities can be Workspace enabled in your development environment. If they have been Workspace enabled in your development environment and you are working in that environment, then you can only modify them in an editable Workspace. You do not need an editable Workspace to create and edit Responsibilities in your Production environment.

1. In the Siebel client, navigate to the Administration - Application screen, Views view.
2. In the View Name field, click the down arrow.
3. In the View list, choose the name of the view, and then click OK.
4. Navigate to the Administration - Application screen, Responsibilities view.
5. Choose the responsibility that you must associate to the view.

   You cannot modify the SADMIN responsibility that Siebel CRM provides as seed data.
6. In the Views list, enter a new record for the view.

7. Depending on the nature of the new view and the users who access it, you might need to do the following:
   a. Add new responsibilities.
   b. Add employees to the new responsibilities.
   c. Make views read-only for a responsibility.

      The read-only feature allows you to use the Siebel client to define a read-only view instead of creating objects in the Siebel repository.

   For more information about responsibilities and employees, see *Siebel Security Guide* .

Configuring a View

This topic describes options for configuring a view. It includes the following information:

- *Using the Views List to Create a View*
- *Configuring the Thread Bar*
- *Defining the Drilldown Sequence to Configure Search for an Account*
- *Creating an Applet Toggle*
- *Defining Whether a View Is Cachable*
- *Creating a Secure View*
- *Creating a View That Requires an Explicit User Login*
- *Restricting Access to Records in a View*
- *Troubleshooting a View That Siebel CRM Does Not Display in the Siebel Client*

Using the Views List to Create a View

You can use the Views list to create a new view. It is recommended that you use the New View Wizard because it prompts you for all necessary information and it defines all the required objects. For more information, see *Creating a View*.

> **Note:** Responsibilities, Views, and their relationship can be Workspace enabled in your Development environment. If you have Workspace enabled Responsibilities, the changes below must be done in an editable Workspace while in your Development environment. If you are editing Responsibilities, Views, and their relationship in your Runtime Repository environment, there is no need for an editable Workspace.

To use the Views list to define a view

1. In the Object Explorer, click View.
2. In the Views list, add a new view, using values from the following table.

Property	Description
Name	Required. Enter the name of the view. References to the view are defined through the name.

Property	Description
Business Object	Required. Enter the name of the business object that the view references. The business object determines the relationship between business components that the applets reference.
Screen Menu	If TRUE, then Siebel CRM includes the view in the Site Map.
Title	Enter a text string. Siebel CRM displays this string in the window title when it displays the view in the Siebel client.

3. In the Views list, right-click the record, and then click Edit Web Layout.
4. In the Select Template dialog box, choose a view web template, and then click Next.
5. In the Applet dialog box, choose the applets that Siebel CRM must display on the view, and then click Next.
6. Review your choices, and then click Finish.

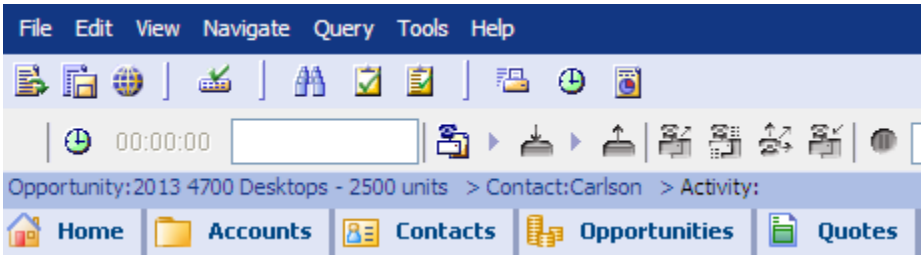
For information about configuring visibility, see *Siebel Security Guide* .

Configuring the Thread Bar

The *thread bar* is a navigation device that Siebel CRM displays immediately after the application toolbar. It helps the user track the navigation path among views. For more information, see *Using Web Templates to Configure the Thread Bar*.

An example of a thread bar (also shown in the following image) is as follows

```
Opportunity:2013 4700 Desktops - 2500 units > Contact:Carlson > Activity
```



To configure the thread bar

1. In the Object Explorer, click View.
2. In the Views list, locate the view you must modify, and then set properties for the view using values from the following table.

Property	Description
Thread Applet	The applets in the view that supply a value for the thread field.

Property	Description
Thread Field	The name of the field that Siebel Tools displays after the greater than sign (>) in the thread bar. This field is in the business component that the Thread Applet references.
Thread Title	The text that Siebel Tools displays before the greater than sign (>) in the thread bar. This text identifies the view. For example, the Thread Title property is Acct for most views that display accounts, such as Account List view and Account Detail - Contacts view.

3. Repeat the last step for each view that requires a thread bar.
4. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Defining the Drilldown Sequence to Configure Search for an Account

If the user sends a query for an Account from the Account Home page, then the visibility that the user possesses determines how Siebel CRM does the search in the following sequence:

1. All Across
2. All
3. My Team
4. My

The Sequence property of the drilldown objects that Siebel CRM defines for the Account Home Search Virtual Form Applet determines the sequence.

To define the drilldown sequence to configure search for an account

1. In Siebel Tools, in the Object Explorer, click Applet.
2. In the Applets list, locate the Account Home Search Virtual Form Applet.
3. In the Object Explorer, expand the Applet tree, and then click Drilldown Object.
4. In the Drilldown Objects list, modify the Sequence property according to your required search sequence.

   Siebel Tools lists several predefined drilldowns, such as Account List View, All Account List View, and All Accounts across Organization. Siebel CRM begins the search in the view that is defined for the drilldown object that contains the highest sequence.

   It is not necessary to define other properties, such as Hyperlink Field.
5. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Configuring Search When No Drilldown Object Is Defined

If no drilldown object is defined, then the Mirror Search GotoView applet user property on the Account Home Search Virtual Form Applet determines the view that Siebel CRM searches. Siebel CRM predefines the value for this user property to Account List View, but you can modify it to configure search if no drilldown object is defined. For example,

modifying the value of the Mirror Search GotoView applet user property to All Account List View causes Siebel CRM to search all accounts.

Creating an Applet Toggle

Assume you use the Contact business component to store information about the preferred payment method that your customers use. Payment methods are cash, credit card, or check.

The following table describes the data requirements for each payment method.

Payment Method	Data Requirement
Cash	No special data requirements exist. The default payment method is Cash.
Credit Card	The following data is required: • Credit Card Type • Credit Card Number • Expiration Date
Check	The following data is required: • Checking Account Number • Routing Number • Driver License Number • Driver License State

You can use a static toggle applet or a dynamic toggle applet for this example:

- To allow the user to toggle between the different applets, a static toggle applet requires the user to choose the applet from the Show list.
- A dynamic toggle toggles between applets that reference the value in the Payment Type field.

For more information, see *Options to Toggle Between Applets in a View*.

To create an applet toggle

1. In Siebel Tools, display the Applet Toggle object type.

   For more information, see *Displaying a System Field in an Applet*.
2. Create the following fields in the Contact business component:
   - Payment Method. Use a static, bound list that contains Cash, Credit Card, and Check.
   - Credit Card Type.
   - Credit Card Number.
   - Expiry Date.

- Checking Account Number.
- Routing Number.
- Driver License Number.
- Driver License State.

3. Display the Payment Method Field in the Contact Form Applet.
   This applet is the default applet that Siebel CRM uses if the preferred payment method of the contact is Cash.
4. Create two copies of the Contact Form Applet.
   Name one applet Contact Form Applet - Credit Card and the other applet Contact Form Applet - Check.
5. Display the following fields in the Contact Form Applet - Credit Card applet:
   - Credit Card Type
   - Credit Card Number
   - Expiry Date BC Fields

   If the preferred payment method of the contact is Credit Card, then the Contact Form Applet - Credit Card applet allows the user to enter credit card information for the contact.
6. Display the following fields in the Contact Form Applet - Check applet:
   - Checking Account Number
   - Routing Number
   - Driver License Number
   - Driver License
   - State BC Fields

   If the preferred payment method of the contact is Check, then the Contact Form Applet - Check applet allows the user to enter checking account information for the contact.
7. In the Object Explorer, click Applet, and then locate the Contact Form Applet in the Applets list.
8. In the Object Explorer, expand the Applet tree, click Applet Toggle, and then create a new applet toggle in the Applet Toggles list using values from the following table.

Property	Value
Applet	Contact Form Applet - Check
Auto Toggle Field	Payment Method
Auto Toggle Value	Check
Name	Contact Form Applet - Check
Parent Name	Contact Form Applet

To create this example with a static toggle applet, leave the Auto Toggle Field and Auto Toggle Value properties empty.

9. Create another new applet toggle in the Applet Toggles list using values from the following table.

Property	Value
Applet	Contact Form Applet - Credit Card
Auto Toggle Field	Payment Method
Auto Toggle Value	Credit Card
Name	Contact Form Applet - Credit Card
Parent Name	Contact Form Applet

To create this example with a static toggle applet, leave the Auto Toggle Field and Auto Toggle Value properties empty.

10. Set the Immediate Post Changes property of the Payment Method business component field to TRUE.

    If the Immediate Post Changes property is FALSE, then the toggle does not occur until the user saves the record.

11. Test and then deliver your Workspace.

    For more information, see *Using Siebel Tools* .

Defining Whether a View Is Cachable

In order for Siebel CRM to be able to display a view, the High Interactivity Enabled property of the underlying class of an applet in the view must be set to 2, 3, 4, or 5. Values 2 or 4 specify that views with applets that use this class are cachable. Values 3 or 5 specify that views with applets that use this class are not cachable. The value 1 has been deprecated.

To define whether a view is cachable

1. Display the Class object type.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. In the Object Explorer, click Applet.
3. In the Applets list, locate an applet that Siebel CRM displays in the view.
4. Note the value in the class property.
5. In the Object Explorer, click Class.
6. In the Classes list, locate the class you noted in step 4.
7. Set the High Interactivity Enabled property of the class to one of the appropriate values.

8. Repeat step 3 through step 7 for each applet that Siebel CRM must display in the view.
9. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Creating a Secure View

You can use the HTTPS protocol to define a secure view for your Siebel application. If a view is marked as secure, then the Siebel Web Engine verifies that the current request uses the HTTPS protocol, thereby preventing a user from entering `HTTP` into the browser to access a secure view instead of entering `HTTPS`.

Note: Oracle always recommends using HTTPS for security of the user interface and, when you use Siebel CRM with iframes, to use the same type for both applications: either HTTP or HTTPS. Browsers often require this configuration by default.

To create a secure view

1. In Siebel Tools, click View in the Object Explorer.
2. In the Views list, locate the view you must modify.
3. Set the Secure property to TRUE.

   For the Siebel client, the Siebel Web Engine specifies the HTTPS protocol when it creates URLs to the view.

The implementation of the HTTPS protocol is external to the Siebel Web Engine. The browser and the Siebel Application Interface negotiate the HTTPS. The Siebel Web Engine only specifies that HTTPS must be used for a specific view. HTTPS must be allowed on any server that provides a secure view.

Creating a View That Requires an Explicit User Login

The user can log in to a Siebel Web Engine application in the following ways:

- Explicitly entering the username and password in the login dialog box.
- With a cookie after the user logs in and if this user chooses Save My Username and Password.

To display a view that provides access to a sensitive part of the web site, you can require the user who logs in with a cookie to explicitly supply the user name and password.

To create a view that requires an explicit user login

Note: Views can be Workspace enabled in your Development environment. If you have Workspace enabled Views, the steps below must be done in an editable Workspace while in your Development environment.

1. In Siebel Tools, click View in the Object Explorer.
2. In the Views list, locate the view you must modify.
3. Set the Explicit Login property to TRUE.

   If a user logs in with a cookie, and if this user attempts to access this view, then Siebel CRM prompts the user to enter the user name and password. Siebel CRM requires this user to perform this login only one time for each

session. All subsequent visits that the user makes to this view during the session do not require the explicit login.

Restricting Access to Records in a View

The Read Only View field of the Responsibilities list in the Administration - Application screen allows you to restrict access to records in a view according to responsibility. If Read Only View contains a check mark, then Siebel CRM does the following:

- Disables New and Delete buttons in the view. Only the Query button remains active. Siebel CRM does not disable other buttons that it might display in the applet, such as New Contact Call or other custom buttons.
- For a Siebel Web Client or Siebel Developer Web Client connected to the server database, it makes the view read only.
- For a Siebel Mobile Web Client connected to the local database, it makes the view not read only.

To restrict access to records in a view

1. Navigate to the Administration - Application screen, and then the Views list.
2. In the Views list, locate the view where you must limit access.
3. In the Responsibilities list, locate the responsibility where you must limit access.
4. In the Responsibilities list, make sure the Read Only View field contains a check mark.

Troubleshooting a View That Siebel CRM Does Not Display in the Siebel Client

To resolve a problem with a view that Siebel CRM does not display in the Siebel client, look for it in the list of Diagnostic Steps or Cause column in the following table.

Diagnostic Steps or Cause	Solution
The view does not exist in the Siebel runtime repository. The spelling of the view name does not match the view name in the Siebel runtime repository.	A view name that you do not spell correctly when you register the view in the Views view of the Administration - Application screen might cause this problem. Make sure the spelling is correct, then deploy your changes to the Siebel runtime repository.
The view belongs to a screen that is not included in the Siebel application that is currently running.	In Siebel Tools, make sure the screen is defined as a child screen menu item of the Siebel application. Make sure the name of the Siebel application is spelled correctly in the configuration file of the Siebel application.
The view is not included in one of the responsibilities for the user who is currently logged in.	Use the Administration - User screen, Employees view to identify the responsibilities that Siebel CRM assigns to the user. Use the Administration - Application screen, Responsibilities view to determine to include or not include the view.

Diagnostic Steps or Cause	Solution
The view is hidden using personalization rules.	Use the Administration - Personalization screen, Views view to determine whether the view is hidden. For testing purposes, you can switch off the EnablePersonalization parameter in the configuration file for Siebel Tools. For more information, see *Setting Up the Configuration File for Siebel Tools*.
Siebel CRM does not display the view in the menu or in the view tabs. The user must drill down from another view to access the view.	In Siebel Tools, make sure the Screen Menu property of the View object is TRUE. It must be TRUE in order for Siebel CRM to include the view in the Site Map. Make sure the view is included in a screen and that the Viewbar Text property of the Screen View child object of the screen is set appropriately. Make sure the Visibility Applet and Visibility Applet Type properties of the view are set correctly. For more information, see *Siebel Security Guide* .
The view does not belong to the same business object as the default view for the screen.	Make sure the view references the same business object.
Siebel CRM does not translate the screen menu item or page tab into the appropriate language.	Make sure a translated string is available for each language for each screen menu item and each screen menu item locale. If a translated string is not available, then Siebel CRM does not display the screen in the Site Map. For a page tab to display, the page tab must include a translated string and a page tab locale that contains the appropriate language code. For example, if Siebel CRM runs in Norwegian, then the Language Code property of the screen menu item locale and page tab locale must be NOR. For more information, see *Localizing Siebel CRM*.
The view is not available because of an upgrade problem.	If you performed an upgrade, then examine the log files that Siebel CRM created during the upgrade to make sure the upgrade was successful. These log files are located in the **DBSRVR** *DB_PLATFORM* directory.
The view is not included in your license keys.	Make sure the view is included in your license keys. Send the license keys to Oracle for examination.

Process of Creating a Screen

To create a screen, perform the following tasks:

1. *Creating a Screen*
2. *Creating a Page Tab*
3. *Creating a Screen Menu Item*
4. *Creating a Screen View*
5. *Defining the Sequence That Siebel CRM Uses to Display Screen Views*

A screen includes groups of related views. A screen view identifies the views and categories that you must associate to the screen. You create a screen view for each category and each view that Siebel CRM must display in a screen.

Creating a Screen

This task is a step in *Process of Creating a Screen*.

You create a new screen in the Screens list in Siebel Tools.

To create a screen

1. In the Object Explorer, click Screen.
2. In the Screens list, add a new screen using values from the following table.

Property	Description
Name	Name of the screen. Other objects use this name to reference the screen.
Default View	View that Siebel CRM displays if the user clicks a page tab in the screen. You must add the view to the screen before you can define the view as the default view.

Creating a Page Tab

This task is a step in *Process of Creating a Screen*.

For more information, see *Page Tab* and *Troubleshooting a View That Siebel CRM Does Not Display in the Siebel Client*.

To create a page tab

1. In the Object Explorer, expand the Application tree, and then click Page Tab.
2. In the Page Tabs list, create a new record using values from the following table.

Property	Description
Screen	The screen you must display through a page tab.
Sequence	The order that Siebel CRM uses to display the page tabs in the Siebel client.
Text	Text string that Siebel CRM displays in a screen tab in the Siebel client. For more information, see *Modifying the Text Style of a Control or List Column in an Applet*.

Creating a Screen Menu Item

This task is a step in *Process of Creating a Screen*.

Siebel CRM does not display on the Site Map a view that filters data according to a visibility rule. Example visibility rules include My Accounts, My Team's Accounts, and so on. Siebel CRM displays screen menu items on the Site Map in alphabetical order. For more information, see *Screen Menu Item*.

To create a screen menu item

1. Make sure Siebel Tools is configured to allow you to modify a text string.

   For more information, see *Setting Up the Configuration File for Siebel Tools*.
2. In the Object Explorer, expand the Application tree, and then click Screen Menu Item.
3. In the Screen Menu Items list, add a new record using values from the following table.

Property	Description
Screen	The screen that Siebel CRM displays if the user clicks the menu item.
Text	The text string that Siebel CRM displays in the Site Map in the Siebel client. For more information, see the description for the Text property in *Creating a Page Tab*.

Creating a Screen View

This task is a step in *Process of Creating a Screen*.

If you define a screen view, then consider the following:

- Use categories to group views, where appropriate.
- Use the Screen View Sequence Editor to define the sequence. Do not edit the Sequence property of the screen view. For more information, see *Defining the Sequence That Siebel CRM Uses to Display Screen Views*.

For more information, see *About Screen Views*.

To create a screen view

1. In the Object Explorer, click Screen.
2. In the Screens list, locate the screen you must modify.
3. In the Object Explorer, expand the Screen tree, and then click Screen View.
4. In the Screen Views list, add a new record.

   When you create a new record, Siebel Tools sets the type property to Detail View and the Category Name and Category Default View properties to read-only.

5. Create a value for the Type property.
6. Define values for other properties.

   For more information, see *Properties of a Screen View*.

Defining the Sequence That Siebel CRM Uses to Display Screen Views

This task is a step in *Process of Creating a Screen*.

You can use the Screen View Sequence Editor to define the sequence that Siebel CRM uses to display views and categories in the Siebel client. The editor is a visual design tool that allows you to view and edit the hierarchy of screen views at each level of navigation. It displays the hierarchy in a tree format and allows you to move individual screen views or categories to different positions in the sequence. You cannot move an item out of the current category of the item or to another level in the hierarchy. If you do not define a sequence, then Siebel CRM orders views alphabetically.

The Sequence property of a screen view displays a number to indicate the sequence in the hierarchy. Siebel Tools updates this field if you open the Screen View Editor, make modifications, and then save these modifications. To view the hierarchy that Siebel CRM displays in the Siebel client, you can sort on the Sequence column in the Screen Views list. If the Sequence property is empty, then Siebel CRM displays that Screen View as the last item in the sequence.

A screen view sequence does not affect how Siebel CRM displays the view in the Site Map. It displays screen views in Site Map in alphabetical order after the screen name.

To define the sequence that Siebel CRM uses to display screen views

1. In the Object Explorer, click Screen.
2. In the Screens list, locate the screen you must modify.
3. Right-click, and then click Edit Screen View Sequence.

   Siebel Tools displays the Screen View Editor and uses a tree to display the screen and the child screen views. A screen view that Siebel CRM displays in bold indicates that it is the Category Default View for the category.
4. In the Screen View Editor, choose a screen view, then right-click and use options described in the following table to move the screen view up or down in the tree.

Option	Description
Move to Next Higher Position	Moves the screen view up one position.
Move to Next Lower Position	Moves the screen view down one position.
Move to Highest Position	Moves the screen view to the highest position in the current level of the hierarchy.
Move to Lowest Position	Moves the screen view to the lowest position in the current level of the hierarchy.

5. Save your modifications, and then exit the Screen View Editor.
6. Test and then deliver your Workspace.

For more information, see *Using Siebel Tools* .

Process of Creating a Screen Home Page View

To create a screen home page view, do the following tasks:

1. *Defining Business Components for the Screen Home Page View*
2. *Creating Links to Frequently Accessed Data*
3. *Determining How Siebel CRM Displays Recent Records*
4. *Defining the Business Object for the Screen Home Page View*
5. *Creating Simplified Screen Home Page Applets*
6. *Creating a Screen Home Page View*
7. *Adding the Screen View to the Screen*

This process describes guidelines that you can use to create a screen home page view. The actual tasks you perform and the sequence that you use to perform them will vary depending on your implementation requirements.

A screen home page view allows the user to access data in a screen. Typically, a screen home page view contains applets that help the user search and add records, and applets that display iHelp items, view links, and recent records. A screen home page view exists for various entities, such as accounts, contacts, opportunities, service, and households. You can define a screen home page view for other entities.

Defining Business Components for the Screen Home Page View

This task is a step in *Process of Creating a Screen Home Page View*.

The Rapid Search and Rapid Add applets reference virtual business components that reference the parent business component of a business object. For example, the Account Home Search Virtual and the Account Home Add Virtual business components reference the Account business component.

To improve performance, you can use a virtual business component for each applet. When Siebel CRM loads the screen home page view, it does not run an SQL query until the user submits a query or adds a record. It allows applets to access data from business components, and avoids display problems that might occur if the applets reference the same nonvirtual business component.

For more information, see *About Business Components, Fields, Joins, and Links*.

To define business components for the screen home page view

1. In the Object Explorer, click Business Component.
2. In the Business Components list, define the Home Search Virtual business component:
   a. Create a virtual business component using values from the following table.

Property	Value
Name	Use the following naming format *to* keep similar records in the Siebel repository consistent: *business component name* Home Search Virtual

Property	Value
	For example, Account Home Search Virtual.
Class	CSSBCVMirrorAdd. This class uses rapid add and rapid search to improve performance.

This virtual business component represents the data that the target business component presents. You must use the Business Components list in Siebel Tools to define a virtual business component. You cannot use the Business Component New Object Wizard because it forces you to associate the business component with a table.

b. In the Object Explorer, expand the Business Component tree, and then click Field.
c. In the Fields list, define the fields that represent fields from the target business components that Siebel CRM must display in the search applet on the home screen.

   The field names in the virtual business component must match the field names in the target business component. Siebel CRM does not support a multi-value group on a rapid search or rapid add applet.
d. In the Object Explorer, click Business Component User Prop.
e. In the Business Component User Props list, create a new record using values from the following table.

Property	Value
Name	Mirror Search Target BusComp
Value	Enter the name of the target business component. For example, Account.

f. In the Business Component User Props list, create a new record using values from the following table.

Property	Value
Name	Mirror Search Target BusObj
Value	Enter the name of the target business object. For example, Account.

3. Create the Home Add Virtual business components:

   a. Repeat Step 2a through Step 2c. Use *business component name* Home Add Virtual as the business component name.
   b. In the Object Explorer, click Business Component User Prop.
   c. In the Business Component User Props list, create a new record using values from the following table.

Property	Value
Name	Mirror Add Target BusComp
Value	Enter the name of the target business component. For example, Account.

   d. In the Business Component User Props list, create a new record using values from the following table.

Property	Value
Name	Mirror Add Target BusObj
Value	Enter the name of the target business object. For example, Account.

   e. Optional. In the Business Component User Props list, create a new record using values from the following table.

Property	Value
Name	Mirror Field *field name* For example, Mirror Field Account.
Value	Pick, *target field*, *mirror pick Id field* For example, Pick, Account, Account Id.

   You use this business component user property to display a dynamic list that does not use a list of values. It identifies a pick field, the corresponding field in the target business component, and the base table Id field in the virtual business component.

f. Optional. Complete this step only if you complete Step 2f. In the Business Component User Props list, create a new record using values from the following table.

Property	Value
Name	Mirror Add *mirror pick Id field name* For example, Mirror Add Account Id.
Value	Ignored Prevents Siebel CRM from adding the Mirror Pick Id Field to the target business component, which might cause a record insertion failure.

Creating Links to Frequently Accessed Data

This task is a step in *Process of Creating a Screen Home Page View*.

The View Links area on a screen home page displays links to the frequently accessed lists of data. Administrators and users can define view links. For more information, see *Siebel Fundamentals* and *Siebel Applications Administration Guide* .

To create links to frequently accessed data

1. Display the Business Component User Prop object type.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. In the Object Explorer, click Business Component.
3. In the Business Components list, locate the SRF Vlink Screen business component.
4. In the Object Explorer, expand the Business Component tree, and then click Business Component User Prop.
5. In the Business Component User Properties list, add a new record using values from the following table.

Property	Value
Name	VlinkScreen: *name of the screen home page* For example, VlinkScreen: Accounts Screen.
Value	Y

6. In the Business Components list, enter the following query in the Name property:

   ```
   *Private View Link
   ```
7. Choose the Public and Private View Link business component.

8. In the Business Component User Properties list, add a new record using values from the following table.

Property	Value
Name	Enter **`Vlink Bo Screen Map`** *name of the screen home page* For example, Vlink Bo Screen Map Account Home.
Value	Enter the *name of the screen home page*. For example, Accounts Screen.

   This user property is required. It uses the business object to associate the view link you define for a screen to the View Link Applet.
9. Make sure the Siebel application in which the view links are checked contains the home page that you defined in step 5.

Requirements for a View Link

A view link must comply with the following requirements:

- An aggregate category link can include multiple aggregate view, detail category, and detail view links.
- A detail category link can only include detail links.
- You can define an aggregate view link in the following ways:
  - Without a parent category
  - With an aggregate category as the parent category
- You cannot define a detail category as the parent category of an aggregate view link.
- You must define an aggregate category or detail category as the parent category of a detail view link.
- You cannot use a detail category as the parent category of an aggregate view.
- An aggregate category cannot define a parent category.

Determining How Siebel CRM Displays Recent Records

This task is a step in *Process of Creating a Screen Home Page View*.

The Recent Records area on a screen home page view displays a list of the last five records in the current screen that the user created, modified, or accessed.

Recent Records only works with business components that reference the CSSBCBase class or subclasses of the CSSBCBase class.

To determine how Siebel CRM displays recent records

1. In the Object Explorer, click Business Component.
2. In the Business Components list, locate the Recent Record business component.

3. In the Object Explorer, expand the Business Component tree, and then click Business Component User Prop.
4. In the Business Component User Props list, add a record using values from the following table.

Property	Value
Name	Recent Record Track BC-*screen home page business object name* For example, Recent Record Track BC-Account Home.
Value	*Name of the target business component.* For example, Account.

This business component user property associates a tracked business component to a screen home page view through the business object.

5. In the Business Components list, locate the target business component, such as Account.
6. In the Business Component User Props list, add a record using values from the following table.

Property	Value
Name	Recent Record Enabled
Value	Y

7. In the Business Component User Props list, add a record using values from the following table.

Property	Value
Name	Recent Record Name Field
Value	Enter the business component field that tracks recent records.

This business component user property specifies the field that tracks recent records. Siebel CRM displays this field in the Recent Record applet in the screen home page view. You can use a calculated field.

8. Optional. In the Business Component User Props list, add a record using values from the following table.

Property	Value
Name	Recent Record Type Field
Value	Enter the field that Siebel CRM uses in the dynamic drilldown.

Property	Value
	The value must reference a field in the parent business component. For example, Order Type LIC in Order Entry - Orders.

This business component user property specifies the field to track. Siebel CRM does not display this field in the Recent Record applet. You can use it to define a dynamic drilldown so that the user can navigate to a different view according to a value. If you configure dynamic drilldown in the recent record applet, then the dynamic drilldown destination objects must reference the Type field in the Recent Record business component.

For example, if the Recent Record Order Entry - Orders List Applet is the recent record applet, then the Sales Order and Web Order dynamic drilldown destination objects must reference the Type field in the Recent Record business component.

Defining the Business Object for the Screen Home Page View

This task is a step in *Process of Creating a Screen Home Page View*.

A screen home page view uses a separate, simplified copy of the business object that the other views in a screen reference. For example, in the Accounts screen the Account Screen Home Page View references the Account Home business object. Other views in the screen reference the Account business object. For more information, see *About Business Objects*.

To define the business object for the screen home page view

1. In Siebel Tools, create a new business object using information from the following table.

Property	Value
Name	*name of the target business component* Home For example, Account Home.
Project	ScreenHomePage

2. In the Object Explorer, expand the Business Object tree, and then click Business Object Component.

3. In the Business Object Components list, create a new business object component for each of the following business components:
    - *Name of the target business component*. For example, *Account.*
    - *Name of the target business component* Home Add Virtual. For example, Account Home Add Virtual.
    - *Name of the target business component* Home Search Virtual. For example, Account Home Search Virtual.
    - Recent Record.
    - Public View Link.
    - Private View Link.
    - Salutation (eApps).
    - Screen Home Task Assistant.

Associating iHelp Items to Business Objects

A screen home page view references a different business object than the other views in the screen reference, so you must associate iHelp items to the business objects. For example, to display the Create a New Account iHelp task in the Activities screen home page view and in the Activities screen views, you must associate the Action Home and the Action business objects with the iHelp item. My Activities, My Team's Activities, are examples of Activities screen views.

Creating Simplified Screen Home Page Applets

This task is a step in *Process of Creating a Screen Home Page View*.

An applet that Siebel CRM displays in a screen home page view is a simplified version of the same applet that Siebel CRM displays in another view. These simplified predefined applets result in a screen home page view that is simple to use and easy to manage. For more information, see *About Applets, Controls and List Columns*.

To create simplified screen home page applets

1. Copy a predefined banner applet.
2. Define properties for the new applet using values from the following table.

Property	Value
Name	*Name of the target business component* Home Screen Homepage Banner For example, Account Home Screen Homepage Banner.
Title	Name of the home screen. Siebel CRM displays this value in the middle area of the home screen and prior to the list of view links.

   The *Account Home Screen Homepage Banner applet is an example of a banner applet.*
3. Create a copy of a predefined rapid search applet, such as the *Account Home Search Virtual Form Applet*.
4. Define properties for the new applet you created using values from the following table.

Property	Value
Name	*Name of the target business component* Home Search Virtual Form Applet For example, Account Home Search Virtual Form Applet.
Business Component	*Name of the target business component* Home Search Virtual For example, Account Home Search Virtual. You created this business component in *Defining Business Components for the Screen Home Page View*.

5. Create a copy of a predefined rapid add applet, such as the Account Home Add Virtual Form Applet.
6. Define properties for the new applet using values from the following table.

Property	Value
Name	*Name of the target business component* Home Add Virtual Form Applet For example, Account Home Add Virtual Form Applet.
Business Component	*Name of the target business component* Home Add Virtual *For example, Account* Home Add Virtual. You created this business component in *Defining Business Components for the Screen Home Page View*.

7. Do the following for each applet you defined:
    a. Remove existing controls from the applet that represent fields from the original business component.
    b. Add new controls to represent fields from the target business component that you must display in the search applet.

       Make sure that the controls reference fields defined in the business component. You can reuse each control that does not represent a field. You do not need to remove them.

8. Do the following for each applet you defined:
    a. Remove existing web template items that represent controls from the original applet.
    b. Add new web template items to represent controls for the new applet.
    c. Repeat for each applet web template mode.

    For example, add and remove web template items for Base and Edit Mode. For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.

    For the Item Identifier property, use a number between 1300 and 1340. This range is available for Rapid Add and Rapid Search applets. For more information about the item identifier, see *Properties of the Applet Web Template Item*.
9. Optional. Identify the target view that Siebel CRM displays when the user clicks Go:
    - If a drilldown object is defined on the source applet, then modify the drilldown object in the new applet. For more information, see *Options to Drill Down to Another View*.
    - If the *Mirror Add GotoView* or the *Mirror Search GotoView* applet user property is defined on the source applet, then modify this value in the new applet.

Creating a Screen Home Page View

This task is a step in *Process of Creating a Screen Home Page View*.

After you define the business components, business objects, and applets, you can define the screen home page view to display the objects and data in the Siebel client. For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data* and see *Creating a View*.

To create a screen home page view

1. In Siebel Tools, click the File menu, and then click New Object.
2. In the New Object Wizards dialog box, click the General tab, click View, and then click OK.
3. Complete the New View dialog box using values from the following table, and then click Next.

Property	Description
Project	Choose ScreenHomePage.
Name	Use the following format: - *Name of the target business component* Screen Homepage View For example, *Account Screen Homepage View.*
Title	Use the following format: - *Name of the target business component* Home For example, *Account Home.*

Property	Description
Business Object	Choose the business object for the home page. For example, Account. For more information, see *Defining the Business Object for the Screen Home Page View*.

4. In the View Web Layout - Select Template dialog box, choose View 25 50 25, and then click Next.

   The View 25 50 25 view web template provides a three column layout.
5. In the Web Layout - Applets dialog box, move the following applets to the Selected Applets window:
   - Layout Controls Applet
   - *Name of the target business component* Home Screen Homepage Banner
   - Public and Private View Link List Applet
   - Recent Record *Name of the target business component* List Applet
   - Screen Home Task Assistant List Applet
   - *Name of the target business component* Home Search Virtual Form Applet
   - *Name of the target business component* Home Add Virtual Form Applet Rapid Search Virtual

   For more information about the Rapid Add and Rapid Search applets, see *Creating Simplified Screen Home Page Applets*.
6. Click Next, and then click Finish.
7. In the View Web Layout Editor, verify the layout of the screen home page view using values from the following table.

Applet	Location
Rapid Search	First quadrant of the screen
Rapid Add	Third quadrant of the screen
Homepage Banner	Upper middle of the screen
Public and Private View Link	Lower middle of the screen
Screen Home Task Assistant List Applet	Second quadrant of the screen
Recent Record	Fourth quadrant of the screen

8. Close the Web Layout Editor.

9. In the Object Explorer, expand the View tree, expand the View Web Template tree, and then click View Web Template Item.
10. In the View Web Template Items list, verify that the Applet Mode and Item Identifier properties are set correctly for each applet. Use values from the following table.

Applet	Applet Mode	Item Identifier
Rapid Search	Query	102
Rapid Add	Edit	103
Homepage Banner	Base	202
Public and Private View Link	Base	203
Screen Home Task Assistant List Applet	Base	302
Recent Record	Base	303

11. For more information, see *Properties of the Applet Web Template Item*.

Adding the Screen View to the Screen

This task is a step in *Process of Creating a Screen Home Page View*.

To display the new screen home page view in Siebel CRM, you must create a new Screen View object to represent it.

To add the screen view to the screen

1. In the Object Explorer, click Screen.
2. In the Screens list, locate the screen you must modify.
3. In the Object Explorer, expand the Screen tree, and then click Screen View.
4. In the Screen Views list, create a new record using information from the following table.

Property	Value
View	*Choose the name of screen view.* For example, Accounts Screen Homepage View.
Type	Aggregate View

Property	Value
Viewbar Text	*Enter the text that* Siebel CRM must display *in the viewbar.* For example, Accounts Home.

5. Right-click in the Screen Views list, click Edit Screen View Sequence, and then define the sequence for the screen view.

   For more information, see *Defining the Sequence That Siebel CRM Uses to Display Screen Views*.
6. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Creating and Deploying an Application

This topic describes how to define and deploy an application. It includes the following information:

- *Creating a New Application*
- *Configuring Keyboard Shortcuts for an Application or Applet*

Creating a New Application

You can modify a predefined application to meet most of your business requirements. You can create a new application, if necessary. For more information, see *About Applications* and *Reusing Predefined Objects*.

To create a new application

1. In Siebel Tools, click Application in the Object Explorer.
2. In the Applications list, create a new record.

   To create a new application, you can copy a predefined application, and then modify the properties and child objects of the new application. A field in an applet that is specific to the original application is not available in the new application, as determined by the application name.
3. Enter values for the Project and Name properties.

   For more information, see *Guidelines for Creating an Application*.
4. Add a page container template to the application for your home page.

   For more information, see *About the Container Page*.
5. For each of the following properties, choose a value from the list of available web pages:
   - Login Web Page
   - Error Web Page
   - Acknowledgement Web Page

   For more information, see *How Siebel CRM References Web Pages*.

6. Associate the application with screens.

Configuring Keyboard Shortcuts for an Application or Applet

This topic describes how to configure keyboard shortcuts. It includes the following information:

- *About Keyboard Shortcuts*
- *Guidelines for Creating Keyboard Shortcuts*
- *Creating a Keyboard Shortcut*
- *Modifying or Hiding the Key Sequence*

About Keyboard Shortcuts

A *keyboard shortcut* is a series of key sequences that Siebel CRM runs in reply to a user action. For example, the user can simultaneously press the CTRL and N keys to create a new record. For more information, see *Siebel Applications Administration Guide* .

Objects That Siebel CRM Uses with a Keyboard Shortcut

To create a keyboard shortcut, you configure an accelerator, which is a child of a command. Siebel CRM maps a shortcut directly to a command, so the scope of the actions that the shortcut represents applies to one of the following contexts:

- The active applet
- The entire application

For example, a shortcut that starts a new query uses a context on the current applet. A shortcut that calls the Site Map is independent of the current application context.

Siebel CRM must load commands into the active menu structure for the Siebel client. The command that each shortcut represents must be available to the user. For a command to be available to the user, it must be associated with the application menu or the applet menu for the currently active applet.

For more information, see *Creating a Command Object*.

Guidelines for Creating Keyboard Shortcuts

If you create a keyboard shortcut, then use the following guidelines:

- If the Siebel application runs in extended keyboard mode, then do not override browser functionality that the user already uses. For example, CTRL+C is a common shortcut that many users already use. For example, in Microsoft Internet Explorer, this shortcut copies a text string to the clipboard.
- Group related shortcuts according to the key sequence. For example, to assist the user in remembering shortcuts that perform similar roles, group key sequences that start with CTRL+ALT for query management.
- Do not map a frequently used command to a key sequence that is similar to a sequence that does a significant action that the user cannot reverse. For example, assume CTRL+SHIFT+X performs a log out. In this situation, do not map the CTRL+ALT+X sequence because the user might accidently press Crtl+SHIFT+X.
- You use the administration screens in the Siebel client to configure a keyboard shortcut that is related to the Siebel Communications Server. If you define a shortcut through the views in the Administration - Communications screen, and if this shortcut uses the same key sequence as a shortcut defined in Siebel Tools and compiled to the Siebel runtime repository, then the shortcut defined through the Administration - Communications screen takes precedence. For more information, see *Siebel CTI Administration Guide* .

Creating a Keyboard Shortcut

This topic describes how to create a keyboard shortcut.

To create a keyboard shortcut

1. In Siebel Tools, display the Command object type and all child objects of the Command object type.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. Make sure the command that you require for the shortcut exists.

   If the command does not exist, then you must add it. For more information, see *Creating a Command Object*.
3. Make sure the command is included as part of the active menu hierarchy at the application or the applet level for the application contexts where the shortcut is active.
4. In the Object Explorer, click Command.
5. In the Commands list, locate the command you must modify.
6. In the Object Explorer, expand the Command tree, and then click Accelerator.
7. In the Accelerators list, add a new record using values from the following table.

Property	Description
Name	Enter a name that describes the action that the shortcut performs.
Key Sequence	Enter the key sequence. For example, `Ctrl+Shift+O`.
Display Name	Enter the display name.
Browser Platform	Choose one of the following values: - **Extended.** For extended mode only. - **Basic.** For basic mode only. - **All.** For Extended and Basic modes.

8. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Modifying or Hiding the Key Sequence

You can modify or hide the key sequence for a shortcut.

To modify or hide the key sequence for a shortcut

1. In Siebel Tools, in the Object Explorer, click Command.
2. In the Commands list, locate the command you must modify.
3. In the Object Explorer, expand the Command tree, and then click Accelerator.
4. Optional. In the Accelerators list, modify the Key Sequence property.

   For more information, see *Guidelines for Creating Keyboard Shortcuts*.

5. Optional. Hide the key sequence:
    a. In the Object Explorer, expand the Accelerators tree, and then click Accelerator Locale.
    b. In the Accelerator Locales list, make sure the Display Name property is empty.

       You can hide the key sequence so that it does not display in the Siebel client. The Display Name property of the accelerator locale defines the key sequence for a shortcut. To hide the key sequence, leave this property empty.
6. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

15 Configuring Applet Layouts

Configuring Applet Layouts

This chapter describes how to use the Applet Web Template Editor to configure an applet. It includes the following topics:

- *Process of Using the Applet Web Template Editor*
- *Options for Configuring an Applet Layout*
- *Using Grid Layout for an Applet*

Process of Using the Applet Web Template Editor

To use the Applet Web Template Editor, perform the following tasks:

1. *Setting the Language Mode of the Applet Web Template Editor*
2. *Defining the Applet Mode*
3. *Adding a Control or List Column to an Applet Layout*
4. *Previewing the Applet Layout*

The *Applet Web Template Editor* is a visual editing tool that allows you to modify the layout of an applet, which includes adding and removing controls and list columns. It includes a canvas and a preview mode that allows you to view how Siebel CRM displays the applet in the Siebel client.

The constrain mode affects some text strings. For more information, see *Setting Up the Configuration File for Siebel Tools*.

Note: The feature described in this topic is available in Siebel CRM 18.10 Update and later releases.

Setting the Language Mode of the Applet Web Template Editor

This task is a step in *Process of Using the Applet Web Template Editor*.

Siebel Tools displays the current language at the end of the last quadrant of the Siebel Tools window. For example, Language:ENU. This language allows you to work with data in a language other than English. Example data includes a translatable text string. The language mode determines the records that are specific to a locale that Siebel Tools transfers during check in and check out and compiles to the Siebel runtime repository. You can set the language mode. For more information, see *Overview of Localizing a Siebel Application*.

To set the language mode

1. In Siebel Tools, click the View menu, and then click Options.
2. Choose the Language Settings tab.
3. Choose the appropriate language in the Language window, and then click OK.

For more information, see *Using Siebel Tools* .

Defining the Applet Mode

This task is a step in *Process of Using the Applet Web Template Editor*.

To define the applet mode

1. In the Mode list of the Controls/Columns window, choose the applet mode that you must edit.

   Make sure you choose an active web template. Siebel Tools displays active and inactive web templates. It uses the following term to label an inactive web template:

   inactive

   Siebel Tools does not apply modifications you make to an applet layout in one mode to the applet layout of another mode.
2. In the Application field of the Configuration Context toolbar, choose an application.

   Choose the following to apply modifications to all applications:

   All Applications

   Choose one application to apply modifications to only one application. For more information, see *Options to Determine How Siebel CRM Displays Controls and List Columns in a Siebel Application*.
3. Edit the applet layout.

   If you add a new control or list column to the applet layout, then you can use the Properties window to define an object property, such as Field or Name.
4. Save your modifications to the web layout.

Adding a Control or List Column to an Applet Layout

This task is a step in *Process of Using the Applet Web Template Editor*.

You can use the Applet Web Template Editor to add a control or list column to an applet:

- You can add a predefined control or list column that is a child object of the applet that exists in the Siebel repository but that Siebel CRM does not map to the applet web template.
- You can add a custom control or list column to an applet layout. For example, you can add a custom control to the applet layout that displays a custom business component field.

An applet header or footer can include a button control. You cannot place a nonbutton control, such as a field, in an applet header or footer.

For more information, see *About Applet Controls and List Columns*.

To add a control or list column to an applet layout

1. In Web Tools, open a workspace and then navigate to Object Explorer.

For information on using the workspace dashboard, see *Using Siebel Tools* .

2. Click Applet and then locate the applet you must modify.
3. Expand the applet list and click Applet Web Template and then Applet Web Template Item.
4. Select an applet web template and add a new control.
5. Preview your changes in the desired application.
6. To preview your changes, log in to the selected application, go to the workspace dashboard and then click Inspect.

How Siebel Tools Treats Labels and Controls in a Grid Layout

If an applet references an applet web template that uses a grid layout, then Siebel Tools treats the labels and controls as separate items. Siebel Tools does this to provide more flexibility when you design the layout. This functionality requires you to map the control and the label of the control onto the applet layout. A label includes the same name as the control, except that Siebel Tools appends the label with the word *label*. For more information, see *Using Grid Layout for an Applet*.

Deleting a Control or List Column

You can cut or delete a control or list column from an applet layout. It is not necessary to delete the object definition for the control or list column. For important caution information, see *Deleting a Control or List Column While in Language Override Mode*.

To delete a control or list column

1. In Web Tools, open a workspace and then navigate to Object Explorer.

   For information on using the workspace dashboard, see *Using Siebel Tools* .
2. Click Applet and then locate the applet you must modify.
3. Expand the applet list and click Applet Web Template and then Applet Web Template Item.
4. In Object List Editor, select an applet web template from the parent applet and then delete the required applet web template item from the child applet.
5. Click the preview (eye-shaped) icon in Applet Web Templates to verify that the web template does not use the control.

   The preview displays after the child applet.
6. Save your changes.

   Web Tools removes the item from the editor and deletes the corresponding applet web template object definition from the Siebel repository.

   > **Note:** You can delete a control only if it was created in the current version of the workspace. For a control created in a previous version, inactivate it to disable it in the object design.

Previewing the Applet Layout

This task is a step in *Process of Using the Applet Web Template Editor*.

You can preview the applet layout to view how Siebel CRM displays the applet in the Siebel client. You can preview the layout in the following ways:

- In different applet modes.

- For one application.

When working with the preview mode, consider the following:

- If Siebel Tools displays the layout of a grid applet in preview mode, then Siebel Tools might compress spaces between fields and the spaces in labels. Siebel Tools does not compress fields.

To preview the applet layout

1. In Web Tools, open a workspace and then navigate to Object Explorer.

   For information on using the workspace dashboard, see *Using Siebel Tools* .
2. Click Applet and then locate the applet you must preview.
3. Expand the applet list and click Applet Web Template and then Applet Web Template Item.
4. In the Applet Web Templates applet, click the preview (pencil-shaped) icon.

   The preview displays after the child applet as an approximation of how Siebel CRM displays the applet in the Siebel client. For more information on preview, see *Using Siebel Tools* .

Options for Configuring an Applet Layout

This topic describes options for configuring an applet layout. It includes the following information:

- *Configuring the Display Name for a Control Caption or List Column*
- *Displaying a Parent Applet Field in the Title of a Detail Applet*
- *Displaying a Subset of Fields or CRM Records*
- *Displaying a Field Only If the User Chooses Show More*
- *Setting the Input Method Editor Mode on a Control or List Column*

Note: The feature described in this topic is available in Siebel CRM 18.10 Update and later releases.

Configuring the Display Name for a Control Caption or List Column

You can configure the caption of a control or the display name of a list column.

For more information, see *Setting Up the Configuration File for Siebel Tools*.

To configure the display name for a control caption or list column

1. In Web Tools, click Applet in the Object Explorer.
2. In the Applets Web Template list, locate the applet web template you must modify, and then click the preview button.
3. In the canvas, double-click a control or list column.
4. Choose the text in the display name or caption, and then type new text.

   Web Tools searches for a symbolic string that is an exact match to the text you type, and that is unique, and then does the following:

- If Web Tools finds an exact match, then Web Tools references the symbolic string from the control or list column and enters the value of the current string in the Display Name or Caption field. After you save your work, Web Tools updates the Display Name property for the control or list column.

  > **Note:** You can select from Display - String Reference or enter in Display – String Override for controls. Likewise, you can select from Caption – String Reference or enter in Caption – String Override for list applets in the Property Pane of the Siebel IDE for layout editing.

- If Web Tools does not find an exact match, or if the match is not unique to a single symbolic string, then Web Tools displays an error message.

5. Optional. You can use the Controls or List Columns list to modify the control caption or list column display name:

   - Define the Caption property for a control and the Display Name property in the Property Pane of the Siebel IDE for layout editing.

Displaying a Parent Applet Field in the Title of a Detail Applet

You can display the value of a field from the parent record as the title of a detail form applet. Siebel CRM often uses a form applet as a detail applet. Displaying the title in this way helps the user to understand the relationship between the parent and child applet.

To display a parent applet field in the title of a detail applet

1. In Web Tools, click Applet in the Object Explorer.
2. In the Applets list, locate the detail applet web template you must modify, and then click the preview button.
3. Relocate a text control into the placeholder for the title that is positioned on the Siebel IDE editing layout.
4. Give the control a useful name, such as *business component name* Title.
5. Modify the HTML Type property of the control to PlainText in the Property Pane.
6. In the Field property of the control, choose the parent business component field whose value you must display, for example Name.
7. Preview your changes in the desired application.

   For more information, see *Using Siebel Tools* .

Displaying a Subset of Fields or CRM Records

You can configure Siebel CRM to display a subset of fields in a form applet or a limited number of CRM records in a list applet. If the user clicks Show More, then the applet displays more fields or records. The applet includes a Less mode and a More mode. The user can toggle between these modes to display more or fewer controls or list columns.

The Mode property of the applet web template item determines the mode that Siebel CRM uses to display a control or list column. For more information, see *Displaying a Field Only If the User Chooses Show More*.

If no web template item is defined in the More mode for an applet, then Siebel CRM does not display the Show More or the Show Less button. Siebel CRM does not support the more or less feature for a pop-up applet. For more information, see *Configuring Pop-Up Applets and Windows*.

To display a subset of fields or CRM records

1. In Web Tools, click Applet in the Object Explorer.
2. In the Applets list, locate the applet web template you must modify, and then click the preview button.
3. In the Applet Web Template Editor, move a Link control from the Palette window onto the placeholder at the second quadrant of the applet.
4. In the Properties window, set properties for the control using values from the following table.

Property	Value
HTML Bitmap	BTTNS_MORE
HTML Display Mode	EncodeData. For more information, see *Properties of a Control That Displays HTML Content*.
HTML Icon Map	Use one of the following values: o For a form applet, use ToggleLayout. o For a list applet, use ToggleListRowCount.
HTML Type	Link
Method Invoked	Use one of the following values: o For a form applet, use ToggleLayout. o For a list applet, use ToggleListRowCount.
Name	Use one of the following values: o For a form applet, use ToggleLayout. o For a list applet, use ToggleListRowCount.
Read Only	FALSE
Runtime	FALSE
Show Popup	FALSE
Sort	FALSE
Visible	TRUE

5. Close the Applet Web Template Editor.
6. Choose the applet, and then confirm that it now includes the ToggleLayout or ToggleListRowCount applet web template item.

   For more information, see *Properties of the Applet Web Template Item*.
7. Publish you changes in the desired application.

   For more information, see *Using Siebel Tools* .

Displaying a Field Only If the User Chooses Show More

You can define a control so that Siebel CRM only displays the field that references the control if the user chooses Show More.

To display a field only if the user chooses show more

1. In Web Tools, click Applet in the Object Explorer.
2. In the Applets list, locate the applet web template you must modify, and then click the preview button.
3. Click the pin icon that appears on the control/column after you move it to the form.

   > **Note:** The pin is a toggle to enable/disable a (red) border around the control and its label. The pin icon is available only in grid-based applets in Siebel CRM 18.10 Update and later releases.

   A border appears around the control/column to indicate that it will now display only during the Show More mode of the applet.
4. In the Siebel IDE editing layout, click the Show More button.

   Web Tools displays the control in the Siebel IDE editing layout based on the mode you selected in Step 3.

For more information, see *Displaying a Subset of Fields or CRM Records*.

Setting the Input Method Editor Mode on a Control or List Column

An *input method editor* (IME) is an editor that allows you to enter complex characters directly from the keyboard. For example, the characters in an Asian language. Several IME input modes handle different types of characters. For example, the Microsoft Windows Japanese IMEs include Hiragana, Katakana, English, Double-width English, and so on. You create a control or list column user property in Siebel Tools to set the IME mode for a control or list column.

To set the input method editor mode on a control or list column

1. In Siebel Tools, make sure the Control User Prop and List Column User Prop object types are displayed.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. Click Applet in the Object Explorer.
3. In the Applets list, locate the applet you must modify.

4. In the Object Explorer, expand the Applet tree, and then do one of the following:
   - Click Control
   - Expand the List tree, and then click List Column.
5. In the Controls or List Columns list, locate the control or list column you must modify.
6. In the Object Explorer, expand the Control or List Column tree, and then click Control User Prop or List Columns User Prop.
7. In the Control User Props list or in the List Columns User Props list, add the required records.

   The following table lists the values for several example records.

Name	Value
IME	E0010411:Hiragana
IME	E0010411:Full-Width Katakana
IME	E0010411:Half-Width Katakana
IME	E0010411:Full-Width Ascii
IME	E0010411:Half-Width Ascii
IME	E0010411:Direct
IME	E0010411:IMEOFF This setting can be useful for a field that must contain only numeric data, such as a phone number. In this situation, you can restrict the data the user enters to only numeric characters.

   The code that you use for the IME version might vary. For more information, see the topics that describe the following information at the Microsoft TechNet web site:
   - Default input locales
   - Locale IDs, input locales, and language collections for supported Microsoft Windows client and server operating systems
8. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Setting the Tab Order for Fields in an Applet

You can set the sequence of fields that Siebel CRM activates each time the user presses the tab button.

To set the tab order for fields in an applet

1. Open an editable Developer Workspace to adjust the Tab order.
2. In Siebel Web Tools, click Applet in the Object Explorer.
3. In the Applets list, locate the applet you wish to modify, click Applet Web Template.
    - Choose entry with grid type template.
    - Tab order can only be set for Applet Web Templates of type **Applet Template - Grid Layout**. An example of an Applet that can be edited is the *Campaign Template Form Applet*.
4. Click the pencil icon to edit the Web Template.
5. In the edit menu choose Set Tab Order icon

   Siebel Web Tools modifies the mode of the Applet Layout Editor to Set Tab Order and displays a number next to each control. If the user repeatedly presses the tab button, then the number indicates the sequence that the user uses to navigate through the controls.
6. To modify the tab order, click each control in the same sequence that the user must use to navigate through the controls.

   Siebel Web Tools assigns a sequence number to each control when you click the control. This number is visible in the upper right corner of each control. If you want to undo the changes you made, click the cancel icon.
7. After you assign a tab order to all the controls, click the save icon.

   Siebel Web Tools returns the Applet Layout Editor to normal edit mode.

   > **Note:** Save your work or cancel the Set Tab Order mode to use Web Tools, all other controls are read only while you are setting the tab order.
8. Repeat steps 3 through step 7 for each applet web template for the applet you are modifying.

Format Toolbar for Web Tools

Web Tools can format controls in Form applets. Formatting Applet Controls refers to their placement, alignment, size, and spacing.

To format the Controls for a Form Applet

1. Click the icon that opens the Format Toolbar. 
2. Select the Applet Controls that you wish to format and click the appropriate button in the Toolbar.

   > **Note:** You can only click one button at a time.

Icon	Behavior
	Aligns the left edges of controls.
	Aligns the centers of controls along a vertical axis.
	Aligns the right edges of controls.
	Aligns the top edges of controls.
	Aligns the middles of controls along a horizontal axis.
	Aligns the bottom edges of controls.
	Makes the controls the same width.
	Makes the controls the same height.
	Makes the controls the same size.
	Makes the horizontal spacing between controls equal.
	Increases the horizontal spacing between controls.
	Decreases the horizontal spacing between controls.

Icon	Behavior
	Removes the horizontal spacing between controls.
	Makes the vertical spacing between controls equal.
	Increases the vertical spacing between controls.
	Decreases the vertical spacing between controls.
	Removes the vertical spacing between controls.
	Aligns the labels to the left margin.
	Centers the labels.
	Aligns the labels to the far margin.
	Centers the controls horizontally using the default, visible canvas.
	Centers the controls vertically using the default, visible canvas.
	Centers controls vertically using the *entire* grid area. Web Tools allows for more canvas space than is initially displayed and you must scroll to see and use it all. Use this button to center selected controls using the entire vertical size of the grid in Web Tools.
	Centers controls horizontally using the *entire* grid area. Web Tools allows for more canvas space than is initially displayed and you must scroll to see and use it all. Use this button to center selected controls using the entire horizontal size of the grid in Web Tools.

Using Grid Layout for an Applet

This topic describes how to use grid layout for an applet. It includes the following information:

- *Accessing Grid Layout Web Templates*
- *Using the Conversion Wizard to Convert a Form Applet to Grid Layout*
- *Modifying the Web Template to Convert a Form Applet to Grid Layout*
- *Identifying an Applet or Applet Web Template That You Cannot Convert to a Grid Layout*
- *Modifying the Background Color of an Applet*
- *Troubleshooting a Grid Layout Conversion Problem*
- *Guidelines for Working with Grid Layout*

Grid layout is a design technology in the Applet Web Template Editor and some applet web templates that allow you to modify the layout of a form applet without having to directly modify the underlying applet web template. The work space is a grid canvas where controls snap to a grid. You use a palette of layout tools to define the layout of the form applet, such as resizing, aligning, and centering.

If you define a form applet, then it is recommended that you use a template that uses a grid. A template that uses a grid allows you to use the Applet Web Template Editor. This editor helps you to control the layout of the form applet.

For more information, see *About Grid Form Applet Templates*.

Note: The feature described in this topic is available in Siebel CRM 18.10 Update and later releases.

To use grid layout for an applet

1. In Web Tools, click Applet in the Object Explorer.
2. In the Applets list, locate the applet you must modify.
3. In the Object Explorer, expand the Applet tree, and then click Applet Web Template.
4. In the Applet Web Templates list, make sure the Web Template property references the appropriate template. For more information, see *Applet Web Templates That Support Grid Layout*.
5. In the Applets list, click, the preview button.
6. In the Applet Web Template Editor, add and delete controls, and then arrange controls, as necessary.
7. For more information, see *Guidelines for Arranging Controls in Grid Layout*.

Applet Web Templates That Support Grid Layout

The following table describes the applet web templates that support grid layout. For more information, see *About Grid Form Applet Templates*.

Web Template	Description
Applet Form Grid Layout	Use with all modes of form applets. This template includes buttons in the applet header.

Web Template	Description
Applet Popup Form Grid Layout	Use with all modes of popup form applets. This template includes buttons in the applet footer.

Accessing Grid Layout Web Templates

You can access grid layout web templates.

To access grid layout web templates

1. In Siebel Tools, click the View menu, Windows, and then the Web Templates menu item.
2. In the Web Template Explorer, expand the Siebel Web Templates tree, and then click CCAppletFormGridLayout.

   Siebel Tools displays the code for the CCAppletFormGridLayout file in the Web Template File window. You can use this template for a form applet.
3. To view the template, click CCAppletPopupFormGridLayout in the Siebel Web Templates tree.

   You can use this template for a popup form applet.

Using the Conversion Wizard to Convert a Form Applet to Grid Layout

The Applet Web Template Conversion Wizard allows you to convert an applet that does not use a grid layout to an applet that does use a grid layout. This conversion is useful in the following situations:

- You must convert a form applet to use grid layout, you did not previously convert the applet to use a grid layout, and you preserved the custom layout during an upgrade.
- You must convert a custom applet you defined that uses a template that does not use a grid.

To use the Conversion Wizard to convert a form applet to grid layout

1. Make sure the applet or applet web template you must convert can be converted.

   For more information, see *Identifying an Applet or Applet Web Template That You Cannot Convert to a Grid Layout*.
2. In the Configuration Context Toolbar, make sure the Application field contains the context you require.

   For more information, see *How the Siebel Application Context Affects Controls That the Applet Web Template Conversion Wizard Converts*.
3. If you work in Language Override mode, then make sure your Tools Language Mode is configured for the language you must convert.

   For more information, see *About Localization in the Development Environment*.
4. In Siebel Tools, click Applet in the Object Explorer.
5. In the Applets list, locate the applet you must convert.

6. Click the Tools menu, and then click Convert to Grid Layout.
7. In the Applet Web Template Conversion Wizard, move the applets you must convert from the Available Applets window to the Selected Window.
8. Choose more options:
   - It is recommended that you choose the Backup Existing Applet Web Templates option.
   - If you choose the *Label on the Left of the Fields* option, then the Conversion Wizard creates a new form template that does not use a grid, moves labels to the left, and then converts that template to grid layout.
   - If you choose the *Launch Web Layout Editor Upon Completion* option, then the editor displays the applet web template for the last applet that you chose in *Using the Conversion Wizard to Convert a Form Applet to Grid Layout*.
9. Click Next.

   The wizard converts the active web templates to grid layout web templates:
   - If no error occurs, then you can use the Applet Layout Editor to edit the layout of these applets. For more information, see *Process of Using the Applet Web Template Editor*.
   - If an error occurs, then the Applet Web Template Conversion Wizard displays the error in a dialog box. Siebel Tools stores this information in a log file. For more information, see step 7.

   If an item in an applet header or footer does not convert properly, then you might be required to manually modify the item after the conversion. This situation can occur if you map a field to a placeholder in an applet header or footer. You typically map a button control rather than a field to a header or footer.

How the Siebel Application Context Affects Controls That the Applet Web Template Conversion Wizard Converts

The Applet Web Template Conversion Wizard only converts controls that are valid in the current application context that is chosen in the Application field of the Configuration Context Toolbar. For example, if a particular Siebel CRM application is chosen, then Siebel Tools only converts the controls that are valid in the context of that Siebel CRM application. If a control is not valid in the chosen application context, then Siebel Tools displays a dialog box that allows you to cancel the conversion or to continue. If you choose continue, then Siebel Tools creates an entry in a log file for each control that it does not convert. For more information, see *Troubleshooting a Grid Layout Conversion Problem*.

Modifying the Web Template to Convert a Form Applet to Grid Layout

To convert the applet to a grid layout, you can modify the web template that the applet references. In Siebel Tools, you modify the web template file that is associated with each applet mode to a template that supports a grid layout. You manually perform this task for each applet you must convert.

To modify the web template to convert a form applet to grid layout

1. Make sure the applet or applet web template you must convert can be converted.

   For more information, see *Identifying an Applet or Applet Web Template That You Cannot Convert to a Grid Layout*.
2. In the Object Explorer, click the Applet object type.
3. In the Applets list, locate the applet web template that you must modify, and then click the preview button.
4. In the Web Template pane of the Applet Web Template Editor, choose the appropriate template.

For more information, see *Applet Web Templates That Support Grid Layout*.

5. Repeat step 5 for each applet mode.

   After you reference a grid layout template, you can use the Applet Web Template Editor to edit the applet. For more information, see *Process of Using the Applet Web Template Editor* and *Guidelines for Working with Grid Layout*.

Identifying an Applet or Applet Web Template That You Cannot Convert to a Grid Layout

You cannot convert some applets and applet web templates to a grid layout.

To identify an applet web template that you cannot convert to a grid layout

1. In Windows Explorer, navigate to the *SIEBEL_TOOLS_ROOT* `\BIN` directory.

   If you use another operating system, then use the appropriate navigation software.
2. Open the `awtcvtcfg.txt` file.

   The `awtcvtcfg.txt` file is the configuration file for the Applet Web Template Conversion Wizard. It lists applets and applet web templates that you cannot convert.
3. Make sure the applet or applet web template you must convert is not listed.

   Do not modify the list of applet classes and applet web template files that Siebel CRM lists in the configuration file for the Applet Web Template Conversion Wizard. Siebel CRM does not support modification of these classes or files.

You cannot convert the following web templates to grid layout:

- SWLS DetailApplet Template
- SWLS Edit Template

Modifying the Background Color of an Applet

To modify the background color of an applet that references a grid layout web template, you can modify the relevant selectors in the main.css (Cascading Style Sheet). For more information about main.css, see *Siebel Developer's Reference* .

To modify the background color of an applet

- If the applet is the parent applet, then modify the AppletStyle1 selector.

  The parent applet is the highest-level applet in a view. For example:

```
/*Parent Applet Style*/

.AppletStyle1{background-color:#f00000;color:#00f0ff;}
```

- If the applet is the child applet, then modify the AppletStyle3 selector.

The child applet is not the highest-level applet in a view. For example:

```
/*Child Applet Style*/

.AppletStyle3{background-color:#f0f000;}
```

Troubleshooting a Grid Layout Conversion Problem

This topic describes guidelines for troubleshooting a grid layout conversion problem.

To resolve a grid layout conversion problem, look for it in the list of Symptoms or Error messages column in the following table.

Siebel Tools displays these errors in a dialog box at the end of the conversion process. Siebel Tools creates an entry in a log file for each control that it does not convert or for errors that it encounters when it converts an applet to grid layout. The log file is named awtconversion.txt and is located in the `SIEBEL_TOOLS_ROOT\temp` directory.

Symptom or Error Message	Diagnostic Steps or Cause	Solution
Cannot map a control or label.	This problem might be due to an applet web template item that is not explicitly mapped to a control on the original applet web template. To display on the new grid applet web template, the Control property on each web template item must contain a value. For more information, see *Properties of the Applet Web Template Item*.	Use the Applet Web Template Editor to map a control to the applet layout. For more information, see *Adding a Control or List Column to an Applet Layout*.
Cannot convert an applet.	The following items might cause the problem: • The applet does not reference a web template. • Siebel CRM does not support an applet class or associated web template for grid layout. For more information, see *Identifying an Applet or Applet Web Template That You Cannot Convert to a Grid Layout*.	Use Edit Web Layout to associate a valid web template to the applet.
The applet web template is configured for more than one application context.	For more information, see *How the Siebel Application Context Affects Controls That the Applet Web Template Conversion Wizard Converts*.	Choose the appropriate application in the Application field of the Configuration Context toolbar and run the Conversion Wizard again. For more information, see *Using the Conversion Wizard to Convert a Form Applet to Grid Layout*.

Guidelines for Working with Grid Layout

If you use a grid layout, then use the following guidelines:

- Controls snap to a grid where each grid cell measures 8 pixels by 8 pixels.
- A grid layout only partially resizes a field according to the monitor resolution that is set on the computer where the Siebel client runs. For example, if a grid form applet is designed to run on a monitor with a resolution of 1024 by 768, and if Siebel CRM displays the applet on a monitor with a resolution of 800 by 600, then the user must scroll to view the farthest edge of the layout. If Siebel CRM displays the same applet on a monitor set to a resolution of 1280 by 1024, then the form does not occupy the entire width of the screen.

  To reduce the potential for horizontal scroll, the browser eliminates as much padding as possible. Siebel CRM does not modify field sizes, but it does modify empty characters that reside at the beginning or end of a label. For example, an applet that is 150 grid cells wide is 1200 pixels wide, and can display in the width of a screen that is set at a width resolution of 1028. The Tools Preview mode reflects this functionality. You can use the preview mode to track how many cells the form crosses, and to approximate how wide the form is when Siebel CRM displays it in the browser.
- If you configure a control in an applet that uses a grid layout, then place controls marked for More mode at the lower level of the applet. The grid layout applet does not compress empty space in the applet.
- In the Applet Web Template Editor, Siebel CRM displays a label as an item that is separate from the corresponding control. It allows you to independently position the label. Siebel CRM uses labels in the Applet Web Template Editor only. A label does not exist as a separate control in the Siebel repository. A label is defined as an applet web template item. It uses the Caption and Text Alignment- Label properties of the corresponding control. Other properties from the control do not apply.
- The Item Identifier property of the applet web template item indicates the position of the label or control in the grid. For example, in the Edit applet web template of the Account Form Applet, the value of the Item Identifier for Country is 8,072. The label is CountryLabel and is located at 8,064. For more information about the item identifier, see *Properties of the Applet Web Template Item*.
- If you save an applet layout, then the Applet Web Template Editor checks for overlapping controls. If an overlapping control exists, then Siebel Tools displays an error message and you cannot save the layout.
- If you use the alignment buttons on the Format toolbar, then the last item you choose on the canvas is the item that Siebel Tools uses to align, center, and space all other items.

Note: The Format toolbar is available only in Siebel Tools.

Guidelines for Arranging Controls in Grid Layout

The Format Toolbar includes several tools that help you arrange controls that Siebel Tools displays in the Applet Web Template Editor. If you arrange controls in grid layout, then use the following guidelines:

- To determine the action a tool performs, scroll over the tool and view the small pop-up label that Siebel Tools displays.
- A tool in the Format Toolbar is only active if the action that the tool performs in the Applet Web Template Editor is actionable. For example, the Align Lefts tool is only active if you choose more than one control.

- To resize or position a control, always use the Applet Web Template Editor. Do not modify the property of a control in the Controls list.
- You can use the arrow keys to move a control or controls to the position.

Note: The Format toolbar is available only in Siebel Tools.

16 Configuring Applets

Configuring Applets

This chapter describes how to configure an applet. It includes the following topics:

- *Creating an Applet*
- *Using the Applet Method Menu Item Wizard in Web Tools*
- *Configuring Pop-Up Applets and Windows*
- *Configuring Applet Buttons, Controls, and List Columns*
- *Configuring How Siebel CRM Displays Data in an Applet*
- *Process of Configuring Drilldown from the Calendar Applet*

Creating an Applet

This topic describes how to create an applet. It includes the following information:

- *Creating a List Applet*
- *Creating a Form Applet*

Web Tools provides a wizard to guide you through creating Applets. This allows you to create an applet without having to copy an existing applet. It includes the following information:

- *Creating a List Applet in Web Tools Using a Wizard*
- *Creating a Multi-Value Group (MVG) Applet in Web Tools Using a Wizard*
- *Creating a Pick Applet in Web Tools Using a Wizard*
- *Creating a Tree Applet in Web Tools Using a Wizard*
- *Creating a Form Applet in Web Tools Using a Wizard*

For more information, see the following topics:

- *About Applets, Controls and List Columns*
- *Configuring Applet Layouts*
- *Configuring Special-Purpose Applets*
- *Configuring Lists and Pick Applets*
- *Configuring Menus and Toolbars*

Creating a List Applet

You use the List Applet Wizard to create a list applet. The List Applet Wizard helps you configure the applet properties correctly. It also creates child objects, such as web template items. The List Applet Wizard does the following:

- Creates the list applet

- Creates the applet web template
- Creates the list, list columns, and controls
- Creates applet web template items

To create a new applet, you can manually add a record to the Applets list, and then define all the necessary properties and child objects.

To create a list applet

1. Make sure the ClientConfigurationMode parameter is not set to All.

   For more information, see *Setting Up the Configuration File for Siebel Tools*.
2. In Siebel Tools, click the File menu, and then click New Object.
3. In the New Object Wizards dialog box, click the Applets tab, click List Applet, and then click OK.
4. In the General dialog box of the List Applet Wizard, enter values using information from the following table, and then click Next.

Property	Example Value	Description
Project	Account	Choose the project to associate with this applet. Siebel Tools displays only locked projects in the list.
Applet Name	New Account List Applet	Enter a unique name for the applet. For more information, see *Guidelines for Naming an Applet*.
Display Title	Accounts	Enter the title that Siebel CRM must display in the Siebel client. For more information, see *Guidelines for Creating an Applet Title*.
Business Component	Account	Choose the business component that the applet references.
Upgrade Behavior	Preserve	Choose how Siebel CRM upgrades the applet during an upgrade.

   The wizard uses this information to create an applet and to define properties for the applet.
5. In the Web Layout - General dialog box, enter the web templates to use for each applet mode, and then click Next.

   The Web Template Type filters the web templates that the wizard displays. To display all templates, choose Show All Templates. For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.

   Siebel Tools displays a thumbnail image for most templates when you choose the template name. For more information about templates, see *Siebel Developer's Reference* .
6. In the Web Layout - Fields dialog box, choose the fields that Siebel CRM must display in the applet, and then click Next.

   Siebel Tools displays the fields for the business component you defined in *Creating a List Applet*. It displays these fields in the Available Fields window.

7. In the Web Layout - Fields dialog box, choose the controls in the Available Controls window that Siebel CRM must display in the applet, and then click Next.

   For more information, see *Configuring How Siebel Tools Enters Data Into the Selected Controls Window*.
8. Review the information the wizard displays in the Finish dialog box, and then click Finish.

   You can click Back to return to a previous dialog box, if necessary.

   The List Applet Wizard creates the applet and supporting object definitions according to the choices you made. Siebel Tools opens the Applet Web Template Editor and displays the layout of the new list applet ready for you to edit. For more information, see *Process of Using the Applet Web Template Editor*.
9. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Configuring How Siebel Tools Enters Data Into the Selected Controls Window

Siebel Tools adds all the available controls to the Selected Controls window, by default. It gets the available controls from the *Model HTML Controls* applet. This applet specifies the available controls and the template that Siebel CRM uses to map each control.

To configure how Siebel Tools enters data into the Selected Controls window

- Add controls to or remove controls from the *Model HTML Controls* applet.

Creating a Form Applet

You use the Form Applet Wizard to create a form applet. It helps you define the applet properties and create child objects, such as web template items. It does the following:

- Creates the form applet
- Creates a reference from the applet to an applet web template
- Creates the controls
- Creates applet web template items. This work creates a relationship in a control, which makes sure the control references a web template. For more information, see *Properties of the Applet Web Template Item*.

To create a form applet

1. Make sure the ClientConfigurationMode parameter is not set to All.

   For more information, see *Setting Up the Configuration File for Siebel Tools*.
2. In Siebel Tools, click the File menu, and then click New Object.
3. In the New Object Wizards dialog box, click the Applets tab, click Form Applet, and then click OK.
4. In the General dialog box of the Form Applet Wizard, enter values using information from the following table, and then click Next.

Property	Example Value	Description
Project	Account	Choose the project to associate with this applet. Siebel Tools displays only locked projects in the list.

Property	Example Value	Description
Applet Name	New Account List Applet	Enter a unique name for the applet. For more information, see *Guidelines for Naming an Applet*.
Display Title	Accounts	Enter the title that Siebel CRM must display in the Siebel client. For more information, see *Guidelines for Creating an Applet Title*.
Business Component	Account	Choose the business component that the applet references.
Upgrade Behavior	Preserve	Choose how Siebel CRM upgrades the applet during an upgrade.
Use Grid Layout	Check mark	Leave at the default setting, which includes a check mark. For more information, see *Using Grid Layout for an Applet*.

The wizard uses this information to create an applet object and to define the applet properties.

5. Do one of the following:
   - If you chose Use Grid Layout in step 4, then choose to display or not display the applet in Base mode. Siebel Tools displays the appropriate web template for Edit Mode.
   - If you did not choose Use Grid Layout in step 4, then choose the web template you must use for each mode.

   In most situations, it is recommended that you use Edit mode. You can use another mode. For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.
6. In the Web Layout - Fields dialog box, choose the fields that Siebel CRM must display in the applet, and then click Next.

   Siebel Tools displays the fields for the business component you defined in step 4 in the Available Fields window.
7. In the Web Layout - Fields dialog box, choose the controls that Siebel CRM must display in the applet, and then click Next.

   For more information, see *Configuring How Siebel Tools Enters Data Into the Selected Controls Window*.
8. Review the information displayed in the Finish dialog box, and then click Finish.

   You can click Back to return to a previous dialog box, if necessary. The Form Applet Wizard creates the applet and supporting object definitions according to the selections you made. Siebel Tools opens the Applet Web Template Editor and displays the layout of the new list applet ready for you to edit.
9. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Creating a List Applet in Web Tools Using a Wizard

To create a List Applet using the wizard

1. Open an editable Workspace.
2. Click the Magic Wand icon to open the wizard library.
3. Click the **A** icon with the caption **Applet**.
4. Click **Start**.
5. By default, the **Form Applet** radio button is selected. Select **List Applet**.
6. Click **Next**.
7. Enter the appropriate data in the Fields.

Field	Description
Name	This is the name for your new Applet.
Project	The Project to associate to your new Applet.
Business Component	The Business Component from which you will choose Fields and whose data the Applet will handle.
Upgrade Behavior	Choose how Siebel CRM upgrades the applet during an upgrade.
Display Name	Optional. This is how you wish the Applet name to appear when displayed in the user interface. By default, this value will be the name of the Business Component chosen.

8. Click **Next**
9. Choose the Business Component Fields you would like to display as List Columns in the Applet and move them to the right of the wizard's **Fields Applet**.
10. Click **Next**
11. Choose the Controls that you wish to have on the List Applet such as `PositionOnRow` for example. Move them to the right of the wizard's **Controls Applet**.
12. Click **Next**
13. Choose the Web Template that you would like to use for each Applet Mode. Usually, in a List Applet, **Edit List** is the most relevant.
14. Click **Next**
15. Review your choices in the Summary Applet. Click **Finish** if you are satisfied with the choices or **Previous** if you wish to change something.
16. When the wizard completes you can navigate to your Applet and further configure it.

Creating a Multi-Value Group (MVG) Applet in Web Tools Using a Wizard

Web Tools provides a wizard to guide you through creating MVG Applets. This allows you to create MVG Applets without having to copy an existing Applet.

To create an MVG Applet using the wizard

1. Open an editable Workspace.
2. Click the Magic Wand icon to open the wizard library.
3. Click the "A" icon with the caption *Applet*.
4. Click **Start**.
5. Select *MVG Applet* radio button.
6. Click **Next**.
7. Enter the appropriate data in the Fields.

Field	Description
Name	This is the name for your new Applet.
Project	The Project to associate to your new Applet. **Note**: This has been defaulted to *00Phoenix* for your convenience. You may choose any Project you wish or leave the default value if you do not use specific Projects.
Business Component	The Business Component from which you will choose Fields and whose data the Applet will handle.
Upgrade Behavior	Choose how Siebel CRM upgrades the applet during an upgrade.
Display Name	Optional. This is how you wish the Applet name to appear when displayed in the user interface. By default, this value will be the name of the Business Component chosen.

8. Click **Next**.
9. Choose the Business Component Fields you would like to display as List Columns in the Applet and move them to the right of the wizard's *Fields* Applet.
10. Click **Next**.
11. Choose the Controls that you wish to have on the MVG Applet such as *PositionOnRow* for example. Move them to the right of the wizard's *Controls* Applet.
12. Click **Next**.
13. Choose the Web Template that you would like to use for each Applet Mode.

> **Note:** By default, the most appropriate Web Templates will be displayed when you open the MVG Applet for each Applet Mode. But you can show all the Web Templates in the Repository by checking the *Show All Templates* checkbox. This setting can be done independently for each Applet Mode.

14. Click **Next**.

15. Review your choices in the Summary Applet. Click **Finish** if you are satisfied with the choices or **Previous** if you wish to change something.
16. A dialog displays asking if you would like to navigate to the Applet you just created. Click **OK** to navigate to the newly created Applet. Click **Cancel** to return to the view you were on before you started the wizard.

Creating a Pick Applet in Web Tools Using a Wizard

Web Tools provides a wizard to guide you through creating Pick Applets. This allows you to create Pick Applets without having to copy an existing Applet.

To create a Pick Applet using the Wizard

1. Open an editable Workspace.
2. Click the Magic Wand icon to open the wizard library.
3. Click the "A" icon with the caption *Applet*.
4. Click **Start**.
5. Choose the *Pick Applet* radio button.
6. Click **Next**.
7. Enter the appropriate data in the Fields.

Field	Description
Name	This is the name for your new Applet.
Project	The Project to associate to your new Applet. **Note**: This has been defaulted to *00Phoenix* for your convenience. You may choose any Project you wish or leave the default value if you do not use specific Projects.
Business Component	The Business Component from which you will choose Fields and whose data the Applet will handle.
Upgrade Behavior	Choose how Siebel CRM upgrades the applet during an upgrade.
Display Name	Optional. This is how you wish the Applet name to appear when displayed in the user interface. By default, this value will be the name of the Business Component chosen.

8. Click **Next**.
9. Choose the Business Component Fields you would like to display as Controls in the Applet and move them to the right of the wizard's *Fields* Applet.
10. Click **Next**.
11. Choose the Controls that you wish to have on the Pick Applet such as *NewQuery* for example. Move them to the right of the wizard's *Controls* Applet.
12. Click **Next**.

13. Choose the Web Template you wish to use for the Applet Modes such as Base and Edit List.

> **Note:** By default, the most appropriate Web Templates will be displayed when you open the Pick Applet for each Applet Mode. But you can show all the Web Templates in the Repository by checking the *Show All Templates* checkbox. This setting can be done independently for each Applet Mode.

14. Click **Next**.
15. Review your choices in the Summary Applet. Click **Finish** if you are satisfied with the choices or **Previous** if you wish to change something.
16. A dialog displays asking if you would like to navigate to the Applet you just created. Click **OK** to navigate to the newly created Applet. Click **Cancel** to return to the view you were on before you started the wizard.

Creating a Tree Applet in Web Tools Using a Wizard

Web Tools provides a wizard to guide you through creating Tree Applets. This allows you to create Tree Applets without having to copy an existing Applet.

To create a Tree Applet using the Wizard

1. Open an editable Workspace.
2. Click the Magic Wand icon to open the wizard library.
3. Click the "A" icon with the caption *Applet*.
4. Click **Start**.
5. Choose the **Tree Applet** radio button.
6. Click **Next**.
7. Enter the appropriate data in the Fields.

Field	Description
Name	This is the name for your new Applet.
Project	The Project to associate to your new Applet. **Note**: This has been defaulted to *00Phoenix* for your convenience. You may choose any Project you wish or leave the default value if you do not use specific Projects.
Business Component	The Business Component from which you will choose Fields and whose data the Applet will handle.
Upgrade Behavior	Choose how Siebel CRM upgrades the applet during an upgrade.
Display Name	Optional. This is how you wish the Applet name to appear when displayed in the user interface. By default, this value will be the name of the Business Component chosen.

8. Choose the Web Template you wish to use for the Applet Modes such as Base, Edit and Edit List.

> **Note:** By default, the most appropriate Web Templates will be displayed when you open the Tree Applet for each Applet Mode. But you can show all the Web Templates in the Repository by checking the *Show All Templates* checkbox. This setting can be done independently for each Applet Mode.

9. Click **Next**.
10. Review your choices in the Summary Applet. Click **Finish** if you are satisfied with the choices or **Previous** if you wish to change something.
11. A dialog displays asking if you would like to navigate to the Applet you just created. Click **OK** to navigate to the newly created Applet. Click **Next** to return to the view you were on before you started the wizard.

Creating a Form Applet in Web Tools Using a Wizard

To create a Form Applet using the wizard

1. Open an editable Workspace.
2. Click the Magic Wand icon to open the wizard library.
3. Click the **'A'** icon with the caption Applet.
4. Click **Start**.
5. By default, the **Form Applet** radio button is selected.
6. Click **Next**.
7. Enter the appropriate data in the Fields.

Field	Description
Name	This is the name for your new Applet.
Project	The Project to associate to your new Applet.
Business Component	The Business Component from which you will choose Fields and whose data the Applet will handle.
Upgrade Behavior	Choose how Siebel CRM upgrades the applet during an upgrade.
Display Name	Optional. This is how you wish the Applet name to appear when displayed in the user interface. By default, this value will be the name of the Business Component chosen.

> **Note:** You can uncheck the Use Grid Layout checkbox, but it is recommended that you leave it checked. The Grid Layout Web Template allows you to use such features as Tab Layout configuration and the Applet Format Toolbar. If you do not use the Grid Layout Web Template, you cannot use these features to configure the Applet.

8. Click **Next**.
9. Choose the Business Component Fields you would like to display as Form Controls in the Applet and move them to the right of the wizard's **Fields Applet**.
10. Click **Next**.
11. Choose the Controls that you wish to have on the Form Applet such as `NewQuery` for example. Move them to the right of the wizard's **Controls Applet**.
12. Click **Next**.

13. If you unchecked the **Use Grid Layout** checkbox, you will have to choose a Web Template. If you left the **Use Grid Layout** check box checked, a Web Template is automatically selected for you. Choose the Web Template that you would like to use for each Applet Mode. Usually, in a Form Applet, **Edit mode** is the most relevant. Base mode is to make the Applet appear read only. But you would then have to maintain two copies of the Applet and there are better, more dynamic ways to make an Applet read only without having to use the Base mode.
14. Click **Next**.
15. Review your choices in the Summary Applet. Click **Finish** if you are satisfied with the choices or **Previous** if you wish to change something.
16. When the wizard completes you can navigate to your Applet and further configure it.

Using the Applet Method Menu Item Wizard in Web Tools

The Applet Method Menu Item Wizard in Web Tools lets you update an applet's menu by adding, removing, or associating command/menu items. You can either create a new menu item by linking an existing Command object to the applet menu, or manage existing Applet/Class Method Menu Item records to control which commands appear in the applet's menu.

This wizard allows you to modify Applet Menus by adding or removing items in the Menu's list of commands. There are two paths in the wizard.

1. Create New Menu Item
2. Use existing Menu Item

Associating an Existing Command Object to an Applet Menu

This choice takes a Command object that has never been associated to an Applet's Menu and creates the Applet Method Menu Item record under a particular Applet. The "new" part of this is that this Command's association to the Applet's Menu has not previously been created as there are no Applet Method Menu Item records for that Applet using this Command.

To associate a Command to an Applet's Menu:

1. In an editable Developer Workspace start the wizard by choosing the **Applet Method Menu** icon and clicking **Start**.
2. Enter the name of the Applet you wish to modify.
3. Click **Next**.
4. Choose Create New Menu Item.
5. Click **Next**.
6. Choose an existing Command Object from the Pick Applet.
7. Give the Name that you would like to appear in the Applet Menu.
8. Click **Next**.
9. Review the brief summary to make sure all is as desired.
10. Click **Submit** when you are satisfied with your choices.
11. Navigate to the Applet and review the new Applet Method Menu Item records under **Applet** > **Applet Method Menu Item**.

Using Existing Applet Method Menu Item Records to Modify an Applet's Menu

This option will use existing records from two areas. You can add or remove Applet Menu Items from both during the wizard's execution. These records are those that already exist for the current Applet in the Applet Method Menu Item list in the Repository. When you remove an item (move a record from the Selected to the Available list) that record will be deleted from the current Workspace.

1. Applet Method Menu Item records for a particular Applet
2. Class Method Menu Item records for the C++ Class that backs the Applet.

To add or remove Applet Method Menu Items from an Applet's Menu:

1. In an editable Developer Workspace start the wizard by choosing the *Applet Method Menu* icon and clicking **Start**.
2. Enter the name of the Applet you wish to modify.
3. Click **Next**.
4. In the Applet Method Menu Item: Choose Commands Applet, you can add or remove Commands from the Applets currently defined Applet Method Menu Item records. To move items to the Selected Applet, choose them in the Available Applet and click the **>>** icon to move them from **Available** to **Selected**.

   > **Note:** Even though a Command may already be present in the Selected Applet, it may still show in the Available Applet. The two panes may have identical records. This is expected. The Available list shows all the defined records under this Applet's Applet Method Menu Item list. That does not mean that all the records are implemented and will be shown in the Menu. Only when you have moved them to the Selected list and finished the Applet will they be shown in the Applet's Menu. Those Commands in the Selected Applet are the ones that will be implemented in the Applet.

5. Click **Next**.
6. Choose the Commands from the C++ inheritance hierarchy that you wish to implement in the current Applet while in the Applet Method Menu Item: Choose Class Commands View. Command objects can define Class Method Menu Items. These are child records of the Class Object Type in the Repository. For the Applet's current C++ Class and any super classes that define Class Method Menu Items, you will see these records here. Defining a Class Method Menu Item allows you to use that functionality in any Applet that uses that Class or one of its super classes.
7. Click **Next**.
8. The Summary screen will display what you have added and removed from the Applet. If satisfied, click **Submit**.
9. Navigate to the Applet and review the new Applet Method Menu Item under **Applet** > **Applet Method Menu Item**.

Configuring Pop-Up Applets and Windows

This topic describes how to configure pop-up applets and windows. It includes the following information:

- *Guidelines for Creating a Pop-Up Applet or Window*
- *Creating a Pop-Up Control in an Applet*
- *Creating a Pop-Up Applet That Siebel CRM Opens from an Applet*
- *Creating a Pop-Up Applet That Siebel CRM Opens from a Menu Item*
- *Creating a Pop-Up Wizard*

Guidelines for Creating a Pop-Up Applet or Window

If you define a pop-up applet or window, then use the following guidelines:

- You must specify a class in the Class property of the pop-up applet that Siebel CRM derives from the CSSSWEFramePopup class.
- You are not required to specify a business component in the Business Component property of the pop-up applet.
- If you specify a business component for your pop-up applet, then you must specify a business component as a child of the business object of the view that contains the applet that Siebel CRM uses to open the pop-up applet.
- Siebel CRM supports one level of pop-up applet. If you activate a pop-up applet from a pop-up applet, then the most recently activated applet replaces the original pop-up applet.
- Siebel CRM does not support the more and less feature on a pop-up applet. For more information, see *Displaying a Subset of Fields or CRM Records*.

Creating a Pop-Up Control in an Applet

If the HTML Type property of a control or list column is set to Text or Field, then Siebel CRM allows some controls to pop-up in the Siebel client, depending on the data type of the field. Example controls include a calendar or a calculator.

The following table summarizes how the data type of the field affects the pop-up control that Siebel CRM displays. If you define a list for a field, then Siebel CRM pops up a list in the Siebel client instead of a calculator or calendar.

Field Data Type	Pop-Up Control That Siebel CRM Displays
DTYPE_DATE	Calendar
DTYPE_TIME	Time
DTYPE_DATETIME	Combination calendar/time
DTYPE_UTCDATETIME	Combination calendar/time
DTYPE_NUMBER	Calculator
DTYPE_INTEGER	

To create a pop-up control in an applet

1. In Siebel Tools, click Applet in the Object Explorer.
2. In the Applets list, locate the applet you must modify.
3. In the Object Explorer, expand the Applets tree, and then click Control to define the properties of the control.
   To define the properties of a list, expand the List tree, and then click List Column.

4. In the Controls or List Columns list, locate the control or list column you must modify.
5. Set the Read Only property of the control or list column to FALSE.
6. Set the Runtime property of the control or list column to TRUE.
7. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Creating a Pop-Up Applet That Siebel CRM Opens from an Applet

A pop-up applet that Siebel CRM opens from an applet occurs if the user clicks a button on an applet that calls a pop-up window. This window allows the user to edit a set of values, browse through a list, and so on.

To create a pop-up applet that Siebel CRM opens from an applet

1. Display the Control User Prop object type, which is a child of an applet.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. Create the pop-up applet:
   a. In the Object Explorer, click Applet.
   b. In the Applets list, create a new applet.
   c. In the Object Explorer, click Control.
   d. In the Controls list, create two new controls using values from the following table.

Name	Method Called
Cancel	CloseApplet or UndoRecord. This value causes the pop-up applet to close if the user clicks Cancel.
OK	Call a method. For more information, see *Calling a Method for an OK Control*.

3. In the Object Explorer, click Applet.
4. In the Applets list, locate the applet that Siebel CRM uses to open the pop-up applet.
5. In the Object Explorer, expand the Applet tree, and then click Control.
6. In the Controls list, create a control with the Method Invoked property set to ShowPopup.
7. In the Object Explorer, expand the Control tree, and then click Control User Prop.
8. In the Control User Props list, create three new user properties using values from the following table.

Name	Value
Popup	Name of the pop-up applet that you created in step 2. This applet must use a class that Siebel CRM derives from the CSSSWEFramePopup class.
Mode	Optional. Mode of the applet, which is Base or Edit.

Name	Value
	If you do not define this value, then the default is Base. For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.
Popup Dimension	Optional. Dimension of the pop-up window. The format is Height X Width. For example, 500 X 800. If you do not define this value, then Siebel Tools sets the dimensions to the value that the HTML Popup Dimension property of the pop-up applet contains. If the HTML Popup Dimension does not contain a value, then Siebel Tools sets the pop-up window dimensions to 600 X 600.

9. Optional. Add a radio button control for any field that references a static list.

   For more information, see *About Static Lists*, and the topics about the radio button and radio button group in *Siebel Business Process Framework: Task UI Guide* .
10. Test and then deliver your Workspace.

    For more information, see *Using Siebel Tools* .

Calling a Method for an OK Control

This topic describes how to call a method for an OK control.

To call a method for an OK control

1. Create a method on a business service.
2. Use code or some other mechanism to handle the method you defined. Make sure this code does the following:
   - Runs the specialized behavior. For more information, see *Caution About Using Specialized Classes*.
   - Calls the CloseApplet method to close the applet after the specialized behavior completes.

Creating a Pop-Up Applet That Siebel CRM Opens from a Menu Item

You can use the GotoApplet method of the command object to call a pop-up applet from a menu item. This configuration is similar to the ShowPopup method described in *Creating a Pop-Up Applet That Siebel CRM Opens from an Applet*. You can provide an argument through the Method Argument property of the command. The following examples use the GotoApplet method of the command object:

- The Spell Check feature that Check Spelling uses
- Add Items that Siebel CRM uses in the Quote Item List Applet

To view an example of this behavior, do the following in the Siebel client:

1. Navigate to the Quotes screen.

2. To drill down on a quote, click a link in the Name field.
3. Click Menu in the Line Items applet, and then click Add Items.

Siebel CRM sets the Method Argument property of the Add Items command to the following value, causing the applet to display if you click Add Items:

```
Applet=Product Popup Applet
```

To create a pop-up applet that Siebel CRM opens from a menu item

1. Display the Command object type, which is a child of the Applet object type.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. In the Object Explorer, click Command.
3. In the Commands list, create a command using values from the following table.

Property	Value
Method	GotoApplet
Method Argument	Applet=*name of pop-up applet*,ShowMode=*mode of pop-up applet* where: o *name of pop-up applet* is the name of the pop-up applet. Required. o *mode of pop-up applet* is the mode of the pop-up applet. Optional. This value can be Base, Edit, Edit List, or Query. If you do not include the mode, then Siebel CRM uses the default, which is Base. For example: `Applet=Product Popup Applet,ShowMode=Edit List` For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.
Show Pop-up	TRUE

4. In the Object Explorer, click Applet, and then locate the applet where Siebel CRM must open the pop-up.
5. In the Object Explorer, expand the Applet tree, and then click Applet Method Menu Item.
6. In the Applet Method Menu Items list, add a new command. Set the Command property for this new command to the command that you created in step 3.
7. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Creating a New Command Object using the Web Tools Wizard

To create a new command object

1. Log into the Web Tools client.
2. Open an editable Development Workspace.
3. Click the **New Object Wizard** button. It has a magic wand icon.
4. Choose the Command icon and click Start.
5. The first view that appears has three fields.
    a. **Name** (Required): This is the name for your new Command object. It must be a unique Name for a Command object in the Repository.
    b. **Project** (Required): This is the Project in which to put the new Command.
    c. **Target**: The two values are Server and Browser. Choose Server to send the Command to the Server to be handled. Choose Browser if you have written Browser Script to handle the Command.
6. When you have configured the three fields click the Next button.
7. The next view has two fields and a radio button group. The two fields depend on which radio button has been chosen. There are two radio buttons.
    a. **Application**: Choose Application if you want the command to be handled by the Application object. Either you have written script on the Application object to handle the Command's method or the Application itself handles the method in the C++ class. If you pick Application, type the method name in the *Method* field. If there are any method arguments, type them in the *Method Argument* field.
    b. **Business Service**: Choose Business Service if you want the command to invoke a method on a Business Service. If you choose the Business Service radio button, type the method name in the *Method* field. If there are any method arguments type them in the *Method Arguments* field.
8. Once you have configured how you want to handle the Command's method, click the Next button.
9. The next view allows you to configure three properties.
    a. **HTML Bitmap**: If you wish for your Command to be represented by a specific image such as a trash can for example, you can specify that in the HTML Bitmap field.
    b. **Tooltip Text**: If you want to add a Tooltip so that when a user hovers over the Command's control, specify the text here.
    c. **Show Popup**: If, when the Command is invoked, you want a popup to appear, check the Show Popup checkbox. You can then specify the dimensions of the popup box.
        i. **Height** (px): This is the height in pixels for the popup box.
        ii. **Width** (px): This is the width in pixels for the popup box.
10. Once you are satisfied with your configuration click the Next button.
11. The next view gives you a summary of your choices. If you are satisfied with the choices click the Finish button.

Creating a Pop-Up Wizard

You can create a set of pop-up applets that work like a wizard. The procedure you use to do this is similar to the procedure you use to define a dialog box that Siebel CRM opens from an applet. For more information, see *Creating a Pop-Up Applet That Siebel CRM Opens from an Applet*.

The parent applet must be in Edit mode.

To create a pop-up wizard

1. In Siebel Tools, click Applet in the Object Explorer.
2. In the Applets list, locate the pop-up applet you must modify.
3. Expand the Applet tree, and then click Applet Web Template.
4. In the Applet Web Templates list, add multiple templates:
   a. Add one template for each page in your wizard.

      Set the Type property to Edit for each template.
   b. Assign a different value to the Sequence property for each template.

      Use the Sequence property to define the order that Siebel CRM uses to display the templates when the user clicks through the wizard.
5. Add controls to the applet that allow the user to navigate between pages.

   For more information, see *Adding Navigation Controls to a Pop-Up Wizard*.
6. On the last template in the sequence, create a control named Finish that closes the applet, and then updates the parent applet.
7. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Adding Navigation Controls to a Pop-Up Wizard

You can add navigation controls that allow the user to navigate between the pages of a pop-up wizard.

To add navigation controls to a pop-up wizard

1. In Siebel Tools, click Applet in the Object Explorer, and then locate the pop-up applet you modified in *Creating a Pop-Up Wizard*.
2. In the Object Explorer, expand the Applets tree, and then click Control.
3. In the Controls list, add a new control for the Previous button using values from the following table.

Property	Value
Name	Previous
Caption	Previous
Method Invoked	PostChanges

   The Previous button saves the modifications that the user makes, and then navigates the user back to the page whose sequence number is one less than the current page.
4. In the Object Explorer, expand the Control tree, and then click Control User Prop.
5. In the Control User Props list, add a new record using values from the following table.

Property	Value
Name	Sequence
Value	-1

6. Repeat step 3 for the Next button using values from the following table.

Property	Value
Name	Next
Caption	Next
Method Invoked	PostChanges

The Next button saves the modifications that the user makes, and then navigates the user to the page whose sequence number is one greater than the current page.

7. Repeat step 5 for the Next button using values from the following table.

Property	Value
Name	Sequence
Value	1

Configuring Applet Buttons, Controls, and List Columns

This topic describes options to configure applet buttons, controls, and list columns. It includes the following information:

- *Configuring a Spell Check Button on an Applet*
- *Calling a Method from a Button in an Applet*
- *Identifying the Controls and List Columns That Siebel CRM Displays in the Siebel Client*
- *Modifying the Text Style of a Control or List Column in an Applet*
- *Displaying Totals for a List Column in an Applet*
- *Defining the Properties of a Control or List Column If HTML Type Is Text*

- *Using a Control to Allow the User to Click a Link to Activate a Record*
- *Displaying the Save Button*

Configuring a Spell Check Button on an Applet

Siebel CRM can call Siebel Spell Check from an applet menu item. To configure this applet menu item, you create a Check Spelling Field user property for the applet that contains the following objects:

- Check Spelling button
- Field where Siebel CRM does the spell check

Note: The feature described in this topic is available in Siebel CRM 18.10 Update and later releases.

To configure a spell check button

1. In Web Tools, display the following object types:
   - Control object and all child objects of the Control object
   - Applet User Prop, which is a child of the Applet object type

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. Create a check spelling button in the applet that contains the field where Siebel CRM must do the spell check:
   a. In Web Tools, in the Object Explorer, click Applet.
   b. In the Applets list, locate the applet you must modify.
   c. In the Object Explorer, expand the Applet tree, and then click Control.
   d. In the Controls list, add a new record using values from the following table.

Property	Value
Name	ButtonCheckSpelling
Caption	Check Spelling
Field	Do one of the following: - If this is a nonrequired field, then enter the field name. - If this is a required field, then leave *this property empty.*
HTML Type	MiniButton
HTML Only	Contains a check mark.
Method Invoked	ShowPopup

Property	Value
	If Web Tools does not display the Method Invoked in the list, then enter it in manually.

3. Define user properties for the spell check button:
    a. In the Object Explorer, expand the Controls tree, and then click Control User Prop.
    b. In the Control User Props list, create three new records using values from the following table.

Name Property	Value Property
Mode	Edit
Popup	Spell Checker Popup Applet
Popup Dimensions	560 X 350 This is the recommended initial size.

4. Add the spell check button to the web template:
    a. In the Object Explorer, click Applet.
    b. In the Applets list, locate the applet web template you modified in Step 2, and then click the preview button.
    c. In the Controls/Columns window, make sure the Mode list is set to Edit.
    d. In the Controls window, choose the Check Spelling control and, keeping the mouse button depressed, move it onto a placeholder in the Applet Web Template Editor and then release the mouse button.
5. Associate the Spell Check business component with the business object that the applet you modified in step 2 references:
    a. In the Object Explorer, click Business Object.
    b. In the Business Objects list, locate the business object where you must add the Spell Check business component.
    c. In the Object Explorer, expand the Business Object tree, and then click Business Object Component.
    d. In the Business Object Components list, add a new record using values from the following table.

Property	Value
BusComp	Spell Checker Applet VBC

6. Create a spell check menu item:
    a. In the Object Explorer, click Applet.
    b. In the Applets list, locate the applet you modified in step 2.
    c. In the Object Explorer, expand the Applets tree, and then click Applet Method Menu Item.

d. In the Applet Method Menu Items tree, add a new record using values from the following table.

Property	Value
Command	Check Spelling
Menu Text	&Check Spelling
Position	2

7. If the field you are configuring for spell check is a required field, then do the following:
   a. In the Object explorer, click Applet User Prop.
   b. In the Applet User Properties list, add a new record using values from the following table.

Field	Value
Name	Check Spelling Field
Value	Enter the name of the control or list column that is mapped to the field that will use spell check.

8. If you must configure spell check for multiple fields in an applet, then repeat steps 2 through step 4 for each additional field.

   You must create a button for each field.
9. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Calling a Method from a Button in an Applet

If Siebel CRM displays a button control, then this button can call a method that comes predefined with Siebel CRM, or a custom method that you define in Siebel Visual Basic, Siebel eScript, or browser script. The Method Invoked property specifies the name of the method that Siebel CRM calls if the user clicks the button control. It might be necessary for you to specify your own custom method in the Method Invoked property. This configuration is the only way to call a Siebel Visual Basic, Siebel eScript, or browser script on a button-click event.

The Runtime property must equal TRUE for a button control. If it does not, then Siebel CRM will not run the method you specify.

To call a method from a button in an applet

1. In Siebel Tools, in the Object Explorer, click Applet.
2. In the Applets list, locate the applet you must modify.

3. In the Object Explorer, expand the Applets tree, and then click Applet User Prop.
4. In the Applet User Properties list, add a new record using information from the following table.

Property	Value
Name	CanInvokeMethod: *Name of method* where: o *Name of method is the name of the method called.*
Value	Y You can also define an expression in the Value property. If Siebel CRM evaluates the expression to TRUE in the Siebel client, then it calls the method.

5. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Identifying the Controls and List Columns That Siebel CRM Displays in the Siebel Client

You can run a query that is similar to SQL to identify the controls and list columns that Siebel CRM displays in the Siebel client. This configuration can be useful to determine the combination of columns and values that Siebel CRM requires for the applet and web template. For example, an applet might include 30 fields and the applet web template item might include 28 fields, but only 19 fields are visible in the client. For more information, see *Properties of the Applet Web Template Item*.

To identify the controls and list columns that Siebel CRM displays in the Siebel client

- Run the following query against the Siebel database on the Siebel Server:

```
SELECT a.NAME, wtmit.*
FROM siebel.s_list_column a,
siebel.s_applet b,
siebel.s_list c,
siebel.s_appl_web_tmpl wtmp,
siebel.s_appl_wtmpl_it wtmit
WHERE b.NAME IN ('Contact List Applet')
AND c.applet_id = b.row_id
AND c.row_id = a.list_id
AND wtmp.applet_id = b.row_id
AND wtmit.appl_web_tmpl_id = wtmp.row_id
--and a.name in ('M/M','Birth Date','Suffix','Account','Postal Code'
```

Modifying the Text Style of a Control or List Column in an Applet

You can modify the style of a text string that Siebel CRM displays in a control or list column.

To modify the text style of a control or list column in an applet

1. Make sure Siebel Tools is configured to allow you to modify a text string.

   For more information, see *Setting Up the Configuration File for Siebel Tools*.
2. Embed an HTML tag in the Caption property of a control or in the Display Name property of a list column.

   For example, Siebel CRM uses the value in the HTML tags to display the following value for the Caption property:

   ```
   <font color="red" size=+2><b>Account Name</b></font>
   ```
3. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Guidelines for Modifying the Text Style of a Control or List Column in an Applet

If you modify the text style of a control or list column in an applet, then use the following guidelines:

- Siebel CRM supports an HTML tag that controls text style, such as size, color, italics, and bold.
- Siebel CRM does not support other HTML tags, such as tags that control alignment or position.
- You cannot use an HTML tag in a property that uses a string because the Siebel Web Engine interprets the tag as a literal value.
- You cannot use embedded HTML tags in the title text of a list column to set the text style. Siebel CRM displays these tags as part of the title text. They do not affect the appearance of the title. You can use these tags with text controls on a form applet.

Displaying Totals for a List Column in an Applet

This topic describes how to display totals of values that Siebel CRM displays in the list column of a list applet.

Displaying the Sum of Values That Siebel CRM Displays in a List Column

You can display the sum of values that Siebel CRM displays in a list column in a list applet.

To display the sum of values that Siebel CRM displays in a list column

1. In Siebel Tools, in the Object Explorer, click Applet.
2. In the Applets list, locate the applet you must modify.
3. Expand the Applet tree, and then click List.

4. In the Lists list, set the properties of the List object using information from the following table.

Property	Description
Total Displayed	Make sure the property contains a check mark.
Total Required	Make sure the property contains a check mark.

5. In the Object Explorer, expand the List tree, and then click List Column.
6. Make sure the Total Required property for each list column you must total contains a check mark.
7. In the Object Explorer, in the Applet tree, click Applet Web Template.
8. In the Applet Web Templates list, choose the Base or the Edit List web template.
9. Set the properties of the applet web template using information from the following table.

Property	Description
Web Template	Applet List Totals (Base/EditList)

10. In the list applet template file, set the property attribute of the `<div od-type="control">` tag to Total.

    For example, use one of the following code:

    ```
    <div od-type="control" id="XXX" property="Total"/>
    ```

    or

    ```
    <div od-type="control" id="XXX">
     <div od-property="Total"/>
    <!--od section control close-->
    </div>
    ```

    If the property attribute in the `<div od-type="control">` tag or in the tag is set to a value of total, and if the Total Required property for the list column contains a check mark, then Siebel CRM displays the total for the list column values. If the Total Required property does not contain a check mark, then Siebel CRM does not create an output. This property is valid only if Siebel CRM maps the tag to a list column. For more information, see *About Siebel Web Templates and Siebel Tags*.
11. Test and then deliver your Workspace.

    For more information, see *Using Siebel Tools* .

Displaying a Total That Siebel CRM Derives from an Expression in a Business Component Field

You can display a total that references an expression that is defined in a business component field. For example, the Revenue business component includes the following fields:

- Quantity
- Price

- Calculated Revenue

The Calculated Value property of the Calculated Revenue contains the following expression:

```
[Quantity]*[Price]
```

You can display the following values in a list applet that references this business component:

- **Total quantity.** The sum of all values in the quantity field.
- **Total revenue.** The product of the totals of the quantity and price columns.

To display a total that Siebel CRM derives from an expression in a business component field

1. In Siebel Tools, display the List Column User Prop child object type of the List object type.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. Make sure an expression is defined in the business component field that Siebel CRM maps to the list column.
3. Make sure the Total Required property of the list column contains a check mark.
4. In the Object Explorer, click Applet.
5. In the Applets list, locate the applet you must modify.
6. In the Object Explorer, expand the Applet tree, and then click List.
7. In the Object Explorer, expand the List tree, and then click List Column.
8. In the List Columns list, locate the column you must modify.
9. In the Object Explorer, expand the List Column tree, and then click List Column User Prop.
10. In the List Column User Props list, add a user property named TotalAsExpr.

    Adding this user property is sufficient to evaluate the totals as an expression. Siebel CRM ignores the properties of the field.
11. Set the property attribute of the `<div od-type="control">` tag in the template file to Total.
12. Test and then deliver your Workspace.

    For more information, see *Using Siebel Tools* .

Displaying Totals in a Separate Applet

You can display totals in a separate applet. For example, Siebel CRM displays a form applet after a list in the Quote Details View. This form contains totals of columns that it displays in the list.

To display totals in a separate applet

1. Create a form applet.
2. Place the form applet after the list applet in the view.
3. Create a field in the business component that the applet references.
4. Add the following expression to the Calculated Value property of the business component field:

   ```
   Sum([multi-value field])
   ```

   > **CAUTION:** Never define a Sum([multi-value field]) expression in a list column. This expression requires a separate query run for each record in the list. It can cause significant performance problems.
5. In the business component, create a multi-value link.
6. In the same business component, create a multi-value field that references the multi-value link.

   The multi-value link references the business component that supports the list of values that Siebel CRM sums.

7. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Defining the Properties of a Control or List Column If HTML Type Is Text

This topic describes how to define the properties of a control or list column if the HTML Type is Text.

To define the properties of a control or list column if the HTML Type is Text

1. In Siebel Tools, click Applet in the Object Explorer.
2. In the Applets list, locate the applet you must modify.
3. In the Object Explorer, expand the Applets tree, and then click Control to define the properties of a control. To define the properties of a list, expand the List tree, and then click List Column.
4. In the Controls or List Columns list, locate the control or list column you must modify.
5. Define the Field property.

   Specify the field in the business component that contains the data that the text control or list column displays.
6. Define the Display Format property.

   For more information, see *Defining the Display Format Property for Data That Is Not Text*.
7. Optional. If the Field property of the control or list column references a multi-value field, then do the following:
   a. In the MVG Applet property, specify the applet to use for the multi-value group applet.
   b. Set the Runtime property to TRUE.

      For more information, see *About the Multi-Value Field* and *How the Runtime Property Determines the Icon That Siebel CRM Displays with a Text Box*.
8. Optional. If the control or list column must reference a pick applet, then do the following:
   a. Define the Pick Applet property.
   b. Set the Runtime property to TRUE.

   The Pick Applet property identifies the pick applet to use for the list dialog box. You must define a list for the field that the control or list column references. For more information, see *How the Runtime Property Determines the Icon That Siebel CRM Displays with a Text Box*.
9. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

How the Runtime Property Determines the Icon That Siebel CRM Displays with a Text Box

If the HTML Type property of a control or list column is Text, then the Runtime property of the control or list column determines the icon that Siebel CRM displays with a text box. It uses the values in the following properties of the control or list column:

- If the MVG Applet or Pick Applet property is not empty, then it does the following:
  - If the Runtime property is TRUE, then it activates an icon or arrow after the text box.

- If the Runtime property is FALSE, then it does not display an icon or arrow, making the multi-value group applet or pick applet inaccessible.

- If the MVG Applet and Pick Applet properties are empty, and if the Runtime property is TRUE, then Siebel CRM uses the data type of the field that the Field property references to determine to display or not display an icon for a calculator, an icon for a calendar, or a currency pop-up applet.

Defining the Display Format Property for Data That Is Not Text

This topic describes how to define the Display Format property of a control or list column to display data that is not text.

To define the Display Format property for data that is not text

1. Determine the data type of the field that this control or list column references:
   a. In Siebel Tools, in the Object Explorer, expand the Business Component tree, and then click Field.
   b. In the Fields list, locate the field you specified in *Defining the Properties of a Control or List Column If HTML Type Is Text*.
   c. Examine the Type property to identify the data type for the field.
2. Define the Display Format property depending on the data type you identified in step c.

   For more information, see *Display Format Property of a Control or List Column*.
3. Optional. Format the postal code:
   a. Specify a DTYPE_TEXT data type.
   b. Create a format mask in the Display Format property that consists of number signs (#) and empty spaces. For example, ##### #### for a United States postal code that uses the zip code plus four format.

Using a Control to Allow the User to Click a Link to Activate a Record

You can use the PositionOnRow control to allow the user to click a link to activate a record.

To use a control to allow the user to click a link to activate a record

1. Add a control to the list applet that calls the PositionOnRow method.
2. Make sure the HTML Row Sensitive property of this control contains a check mark.
3. Place this control on the list applet where the link must choose the row.
4. Test and then deliver your Workspace.

   The user must be able to click the link to choose the record.

Displaying the Save Button

A Siebel application uses an implicit save, by default, so Siebel CRM does not display the Save buttons in the predefined application. You can configure these buttons so Siebel CRM displays them.

Displaying the Save Button in the Siebel Web Client

This topic describes how to display the Save button in the Siebel Web Client.

To display the Save button in the Siebel Web Client

- For the Application Object Manager component, set the ShowWriteRecord parameter to True.

Displaying the Save Button in the Siebel Mobile Web Client

This topic describes how to display the Save button in the Siebel Mobile Web Client.

To display the Save button in the Siebel Mobile Web Client

1. Use a text editor to open the configuration file for the Siebel application.
2. Set the ShowWriteRecord parameter to the following value:

```
ShowWriteRecord=TRUE
```

This parameter is located in the InfraUIFramework section of the configuration file. Note that the Save buttons use the WriteRecord method.

Configuring How Siebel CRM Displays Data in an Applet

This topic describes options to configure how Siebel CRM displays data in an applet. It includes the following information:

- *Controlling How the User Creates, Edits, Queries, and Deletes CRM Data*
- *Controlling Query Behavior If the User Presses CTRL+ENTER*
- *Filtering Data That Siebel CRM Displays in an Applet*
- *Displaying HTML Content in an Applet*
- *Displaying a System Field in an Applet*
- *Avoiding Losing Context During a Drilldown*
- *Configuring Quick Fill for a Custom Applet*

Controlling How the User Creates, Edits, Queries, and Deletes CRM Data

You can create an applet web template for each applet mode that determines if the user can create, edit, query, or delete Siebel CRM records in an applet. For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.

To control how the user creates, edits, queries, and deletes CRM data

1. Create a new applet web template.

   For more information, see *Adding a Web Template to an Applet*.

2. Enter text into the Name property to match one of the applet modes described in *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.
3. Set the Type property to one of the applet modes described in *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.
4. Repeat step 1 through step 3 for each applet mode that the applet must support.

   For example, create a separate applet web template in the following situations:

   - Create one applet web template for New and another applet web template for Query.
   - If the applet layout is different for New and Query modes compared to Edit mode, then create a separate web template for each mode.

Adding a Web Template to an Applet

If you must define another mode for an applet, then you must add a web template to the applet. For more information, see *About Applet Web Templates*.

To add a web template to an applet

1. In Siebel Tools, click Applet in the Object Explorer.
2. In the Applets list, locate the applet you must modify.
3. In the Object Explorer, expand the Applets tree, and then click Applet Web Template.
4. In the Applet Web Templates list, add a new record using information from the following table.

Property	Description
Name	Enter the applet mode for the applet web template, such as Edit. For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.
Type	Choose the applet mode of the applet web template.
Web Template	Choose the web template to associate to the applet.

Controlling Query Behavior If the User Presses CTRL+ENTER

The *default method* of an applet is the method that Siebel CRM runs if the user presses CTRL+ENTER. For an applet in query mode, this method is ExecuteQuery. The user can press ALT+ENTER to run the query. For other modes, you can set the DefaultMethod applet user property.

You must use a valid InvokeMethod for the applet, such as NewRecord or GotoNextSet.

To control behavior if the user presses CTRL+ENTER

1. In Siebel Tools, click Applet in the Object Explorer.
2. In the Applets list, locate the applet you must modify.
3. Expand the Applet tree, and then click Applet User Prop.
4. In the Applet User Properties list, add a new record using information from the following table.

Property	Description
Name	Enter **`Default Applet Method`**.
Value	Define the method you must call.

Filtering Data That Siebel CRM Displays in an Applet

To filter CRM data that Siebel CRM displays in the applet, you can define a search specification on an applet.

To filter data that Siebel CRM displays in an applet

1. In Siebel Tools, click Applet in the Object Explorer.
2. In the Applets list, locate the applet you must modify.
3. Create a search specification in the Search Specification property.

   For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.

Displaying HTML Content in an Applet

An *HTML content* control is a type of control that allows you to display HTML content in an applet in the Siebel client. HTML content can be static HTML or HTML that Siebel CRM gets from an external source.

To display HTML content in a field

1. in Siebel Tools, expand the Applet tree, and then click Control in the Object Explorer.
2. In the Controls list, locate the control you must modify.
3. Set properties for the control.

   For more information, see *Properties of a Control That Displays HTML Content*.
4. Test and then deliver your Workspace.
5. Open the Siebel client.
6. Administer host information:
   a. Navigate to the Administration - Integration screen, and then click the WI - Symbolic URL List link.
   b. Make sure the Host Administration visibility filter is chosen.
   c. Enter the name of the HTTP host in the Name field.
   d. Enter the virtual name and authentication parameters, as required for your configuration.

      For more information, see *Using the Host Administration View*.
7. Optional. Administer fixup information:
   a. Navigate to the Administration - Integration screen, and then click the WI - Symbolic URL List link.
   b. Choose Fixup Administration from the visibility filter.

c. Specify how to control the behavior of links that Siebel CRM embeds in the external content.

For more information, see *Using the Fixup Administration View*.

8. Optional. Administer symbolic URL information:

a. Navigate to the Administration - Integration screen, and then click the WI - Symbolic URL List link.
b. Choose Symbolic URL Administration from the visibility filter.
c. Define the content agent for the external host. Include the URL, host name, fixup name, and arguments.

9. Optional. Administer content sets information:

a. Navigate to the Administration - Content Center screen, and then click the Content Sets link.
b. Upload and manage web content that Siebel CRM displays.

For more information about content agents and symbolic URLs, see *Siebel Portal Framework Guide* .

Properties of a Control That Displays HTML Content

The following table describes the properties of a control that you must set to display HTML content.

Property	Description
ContentFixupName	Determines how to correct links after processing. You enter the name of a Fixup as displayed in the Fixup Administration View. Any value you enter does not work if the Field Retrieval Type property is HTML Attachment or Service.
Field Retrieval Type	Determines the type of HTML that Siebel CRM displays in the field. You can choose one of the following values: • **Field Data.** Stores the HTML content as data. • **HTML Attachment.** Displays an HTML attachment. The control displays the HTML Attachment that the field identifies. • **Service.** For more information, see the following section. • **Symbolic URL.** Siebel CRM gets content from an external host that references a symbolic URL. You must define the necessary information that Siebel CRM requires to access the external source. This includes the format for the request, the host name, necessary arguments, and so on. For more information, see *Siebel Portal Framework Guide* . • **URL.** Siebel CRM gets content from an external source. This source references the simple URL that is defined in the underlying field.
HTML Display Mode	Set the HTML Display Mode so that the HTML content displays properly in the browser. You can choose one of the following values: • **DontEncodeData.** Use this value if the field contains actual HTML text and you require Siebel CRM to display the content as HTML text. • **EncodeData.** Use this value if the field contains HTML reserved characters. If the field contains these characters, then Siebel CRM encodes them before it displays them correctly in the browser. Example reserved characters include angle brackets (< >), ampersand (&), and so on. It is recommended that you set the HTML Display Mode to EncodeData for security purposes. For more information about setting this property, see *Siebel Security Guide* .

Setting the Field Retrieval Type Property to Service

If you set the Field Retrieval Type property to Service, then Siebel CRM uses a business service to display the field, and you must do the following:

- Add a child control user property to the control.
- Set the Name property of the control user property to *Field Retrieval Service.*
- Enter the name of the business service into the Value property of the control user property.

For example, to define a control to display a Content Center asset, you do the following:

- Set the Field Retrieval Type to Service.
- Add a Control User Property child object with the Name property set to Field Retrieval Service and the Value property set to ContentBase - Asset Publish Service.

For more information about Content Center Assets, see *Siebel Applications Administration Guide* .

Using the Host Administration View

You use the Host Administration view to specify a host. Specifying a host allows you to do the following:

- Obscure the true server name in the created HTML.
- Specify a set of NCSA Basic Authentication credentials for a content host that requires authentication.
- Control fixup at the host level.

For each host, you must define an external content host server. You can only fix up links that are associated with a defined host.

To view the Host Administration list, navigate to the Administration - Integration screen, choose the WI - Symbolic URL List link, and then make sure the Host Administration visibility filter is chosen.

Using the Fixup Administration View

A *fixup* is a configuration that you use to control the behavior of links that are embedded in external content. A fixup includes a Link Context that corresponds to the fixup type. You can use the Fixup Administration view to administer a fixup. The following types of fixups are available:

- **Do Nothing.** Does not affect any of the links. The links remain as they are with the content that is passed back in the original form. This principle applies to relative and absolute links.
- **Outside Application.** Uses the host and path of the parent URL to convert the relative links to absolute links. Siebel CRM does not proxy any links.
- **Inside Application or Inside Applet.** Does the following:
  - Converts each relative link to an absolute URL link.
  - To maintain the Siebel Web Engine context, proxies any links that use a host that is specified in the Host Administration view. For more information, see *Using the Host Administration View*.

  Fixup is required for links in Siebel Business Application.

Default Link Targets

No default link targets exist that applied to a fixup. You can add a link target to a fixup.

Displaying Fields as HTML Content

Any business component can use the Web Content Assets feature to add fields that Siebel CRM displays as HTML content. For example, you can do the following:

- Display a static HTML message in the Partner Relationship Manager application.
- Display a product description as HTML content.

This topic uses the Partner Message business component as an example of how to configure the Web Content Assets feature.

To display fields as HTML content

1. In Siebel Tools, make sure the Control User Prop object type is displayed.
   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. In the Object Explorer, click Business Component.
3. In the Business Components list, locate the Partner Message business component.
4. In the Object Explorer, expand the Business Component tree, and then click Field.
5. In the Fields list, locate the Message field, and then set properties for the field using values from the following table.

Property	Value
Pick List	ContentBase Asset Hierarchical PickList

6. In the Object Explorer, click Applet.
7. In the Applets list, locate the Partner Message List Applet.
8. In the Object Explorer, expand the Applet tree, expand the List tree, and then click List Column.
9. In the List Columns list, locate the Message Body list column, and then set properties for the column using values from the following table.

Property	Value
Pick Applet	ContentBase Asset Hierarchical PickList

10. In the Object Explorer, click Applet.
11. In the Applets list, locate the Partner Message Entry Form Applet.
12. In the Object Explorer, click Control in the Applet tree.
13. In the Controls list, locate the Message Body Preview control.
    If necessary, add a new control with the Name property set to Message Body Preview.
14. Set properties for the control using values from the following table.

Property	Value
Field Retrieval Type	Service

15. In the Object Explorer, expand the Control tree, and then click Control User Prop.
16. In the Control User Props list, add a new record using values from the following table.

Property	Value
Name	Field Retrieval Service
Value	ContentBase - Asset Publish Service

17. Repeat step 6 through step 16, except this time do the following:
    a. Modify the properties for the MessageBody control of the Partner Message Form Applet (SCW) applet.
    b. Add the control user property to the MessageBody control.
18. Test and then deliver your Workspace.
    For more information, see *Using Siebel Tools* .

Displaying a System Field in an Applet

This topic describes how to display a system field in the list column of a list applet. You can display a system field in the control of a form applet. It is not necessary to define a system field as a child Field object of the underlying business component. For more information, see *System Fields of a Business Component*.

To display a system field in an applet

1. In Siebel Tools, click Applet in the Object Explorer.
2. In the Applets list, locate the applet you must modify.
3. In the Object Explorer, expand the Applet tree, expand the List tree, and then click List Column.
4. In the List Columns list, add a new record.
5. In the Field property of the list column, choose a system field.
6. In the Display Name property, enter a value that describes the data that the system field stores, such as Last Updated.

Avoiding Losing Context During a Drilldown

In some custom configurations, if the user navigates through some views, then Siebel CRM might ignore the search specification in the parent applet. Instead, it requeries the Siebel database. It does this because it detects that the current search specification that is defined on the destination applet is different from the bookmarked search specification. Siebel CRM displays the first record in the applet but loses the record context. If a search specification is defined on the parent applet in the destination view, then Siebel CRM ignores the bookmark.

Configuring Quick Fill for a Custom Applet

You can configure quick fill for a custom applet.

To configure quick fill for a custom applet

1. In Siebel Tools, in the Object Explorer, click Applet.
2. In the Applets list, locate the custom applet you must modify.
3. In the Object Explorer, expand the Applet tree, and then click Applet Method Menu Item.
4. In the Applet Method Menu Items list, create a separate record for each row in the following table.

Command	Menu Text	Menu Text - String Reference	Position
ApplyTemplate	Apply Template	SBL_APPLY_TEMPLATE-1004224602-029	20
SaveTemplate	Save as Template	SBL_SAVE_AS_TEMPLATE-1004224757-0HM	30
NewFromTemplate	New From Template	SBL_NEW_FROM_TEMPLATE-1004224722-0CD	40
NewFromLastTempla	New From Last Template	SBL_NEW_FROM_LAST_TEMPLATE-1004224721-0C(	50

   These values use the Opportunity Form Applet - Child applet as a model. You must adjust the values for the Position property to meet the requirements for your custom applet.
5. Test and then deliver your Workspace.

Process of Configuring Drilldown from the Calendar Applet

This topic describes how to define a destination view if the user clicks the Contact Icon in the calendar. To configure drilldown from the calendar applet, perform the following tasks:

1. *Preparing Siebel Tools to Configure Drilldowns*.
2. *Defining Fields in the Business Component*.
3. *Defining the Applet User Properties*.
4. *Creating the Drilldown Objects and Controls*.
5. Optional. *Configuring a Different Icon for the Dynamic Drilldown*.
6. Optional. *Configuring a Different Destination for the Dynamic Drilldown*.

Preparing Siebel Tools to Configure Drilldowns

This task is a step in *Process of Configuring Drilldown from the Calendar Applet*.

In this topic, you display object types and lock projects.

To prepare Siebel Tools to configure drilldowns

1. Display the following object types. These object types are children of an applet:

- Applet User Prop
- Control
- Dynamic Drilldown Destination

For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.

2. Lock the following projects:
   - HI Calendar
   - Activity
   - Activity HI Calendar

Defining Fields in the Business Component

This task is a step in *Process of Configuring Drilldown from the Calendar Applet*.

The fields you define in this topic are required to support drilldown on the contact icon and a dynamic drilldown on an activity. If you modify the drilldown definition, then you must modify the relevant fields.

To define fields in the business component

1. In the Object Explorer, click Business Component, and then locate the Action business component in the Business Components list.
2. Expand the Business Component tree in the Object Explorer, and then click Field.
3. In the Fields list, add a new field using values from the following table.

Property	Value
Name	Primary Contact Icon
Calculated	True
Calculated Value	IIF([Primary Contact Last Name] is NULL and [Primary Contact First Name] is NULL, "", "<img src='images/icon_copy.gif' border='0'>")

The Primary Contact Icon allows the user to start the drilldown.

4. In the Fields list, add a new field using values from the following table.

Property	Value
Name	Type Category
Calculated	True

Property	Value
Calculated Value	IIF([Type] = LookupValue ("TODO_TYPE", "Appointment"), "A", (IIF([Type] = LookupValue("TODO_TYPE","Presentation"), "P", "O")))

5. In the Fields list, add a new field using values from the following table.

Property	Value
Name	Open Bracket
Calculated	True
Calculated Value	IIF([Primary Contact Last Name] is NULL and [Primary Contact First Name] is NULL, "", "[")

You use the open bracket symbol ([) and close bracket symbol (]) in *Defining the Applet User Properties*. To enclose the corresponding last and first names of the contact with brackets in Siebel CRM, these fields are defined according to the conditions that are set for them.

6. In the Fields list, add a new field using values from the following table.

Property	Value
Name	Close Bracket
Calculated	True
Calculated Value	IIF([Primary Contact Last Name] is NULL and [Primary Contact First Name] is NULL, "", "]")

7. In the Fields list, add a new field using values from the following table.

Property	Value
Name	Contact Details
Join	S_CONTACT
Column	LAST_NAME

Property	Value
	Siebel CRM uses the LAST_NAME column to get details about the contact from the S_CONTACT table. To support drilldown on the contact icon and dynamic drilldown on the activity, details about the contact are required. If you modify the definition of the drilldown, then it might be necessary for you to also modify the relevant fields.

You must define the Join property before you define the Column property.

8. In the Fields list, verify that the Primary Contact Last Name field is defined using values from the following table.

Property	Value
Name	Primary Contact Last Name
Join	S_CONTACT
Column	LAST_NAME

If the field does not exist, then create it. If the field is not defined correctly, then modify it.

9. In the Fields list, verify that the Primary Contact First Name field is defined with values from the following table.

Property	Value
Name	Primary Contact First Name
Join	S_CONTACT
Column	FST_NAME

If the field does not exist, then create it. If the field is not defined properly, then modify it.

10. Create a file name for the icon that Siebel CRM uses with the Primary Contact Icon field.
    a. In the Fields list, locate the Primary Contact Icon field.
    b. In the Calculated Value property, replace icon_copy.gif with a file name that contains an image of the icon you must display.

       If a primary contact is associated to the calendar event, then Siebel CRM displays this icon on a calendar event.
11. Verify that the join to the S_CONTACT table is defined appropriately:
    a. Make sure the Action business component is chosen in the Business Components list.

b. In the Object Explorer, click Join, then query the Alias property of the Joins list for S_CONTACT.
c. Verify that the Table property contains S_CONTACT.
d. In the Object Explorer, expand the Join tree, and then click Join Specification.
e. In the Join Specifications list, verify that the join specification contains the following values.

Property	Value
Name	S_CONTACT
Destination Column	PAR_ROW_ID
Source Field	Primary Contact Id

f. If an S_CONTACT join with the alias S_CONTACT does not exist, then search for a join on the S_CONTACT table that contains the same definition.
   i. If a predefined join with this definition does not exist, then create a new join using values in step b through step e.
   ii. If a join does exist that contains a different alias that meets this definition, then modify the join values to match the values in step b through step e.

Defining the Applet User Properties

This task is a step in *Process of Configuring Drilldown from the Calendar Applet*.

In this topic, you define the links and tooltips for the Activity HI Calendar Applet.

To define the applet user properties

1. In the Object Explorer, click Applet.
2. In the Applets list, locate the Activity HI Calendar Applet.
3. In the Object Explorer, expand the Applet tree, and then click Applet User Prop.
4. In the Applet User Properties list, locate the Display Fields applet user property, and then define the Value property using values from the following table.

Property	Value
Value	Contact Details, Primary Contact Icon, Description

   Siebel CRM displays each field you define in the Value property as a separate link. The Contact Details and Primary Contact Icon fields provide contact details, and the Description field provides Activity information in the calendar. Siebel CRM can use an icon to represent the link.
5. In the Applet User Properties list, locate the Display Field Drilldown Object Names applet user property, then define the Value property using values from the following table.

Property	Value
Value	Contact - Detail, Contact - Detail, Action - Detail

This step defines the link for the drilldown object. The values must match the drilldown object.

6. In the Applet User Properties list, add a new applet user property using values from the following table.

Property	Value
Name	Contact Details.Detailed Description Fields
Value	Open Bracket, Primary Contact Last Name, Primary Contact First Name, Close Bracket

This step defines the display text for the Contact Details link that you defined in step 4. The text that Siebel CRM displays before the period in the name of the applet user property must match the Contact Details field that you defined in the value property in step 4.

7. In the Applet User Properties list, add a new applet user property using values from the following table.

Property	Value
Name	Contact Details.Tooltip Fields
Value	Primary Contact Last Name, Primary Contact First Name

This step defines the tooltip for the Contact Details link. The text before the period in the name of the applet user property must match the Contact Details field that you defined in the value property in step 4.

8. In the Applet User Properties list, add a new applet user property using values from the following table.

Property	Value
Name	Description.Tooltip Fields
Value	Type, Description, Planned, Planned Completion, MeetingLocation, Comment

This step defines the tooltip for the Description link. The text before the period in the name of the applet user property must match the Description field that you defined in the value property in step 4.

9. In the Applet User Properties list, add a new applet user property using values from the following table.

Property	Value
Name	Description.Detailed Description Fields
Value	Description

10. In the Applet User Properties list, add a new applet user property using values from the following table.

Property	Value
Name	Primary Contact Icon.Tooltip Fields
Value	Primary Contact Last Name, Primary Contact First Name

This step defines the tooltip for the Primary Contact Icon link that you defined in step 4. The text before the period in the name of the applet user property must match the Primary Contact Icon field that you defined in the value property in step 4.

Creating the Drilldown Objects and Controls

This task is a step in *Process of Configuring Drilldown from the Calendar Applet*.

In this topic, you define the drilldown objects and controls for the Activity HI Calendar Applet. This configuration allows the user to do a dynamic drilldown on an activity and a static drilldown on a contact. Note that if the Activity Type is Presentation, then the target view is the Activity Participants View. Otherwise, the target view is the eCalendar Detail View.

To create the drilldown objects and controls

1. Make sure Activity HI Calendar Applet is still chosen in the Applets list.
2. In the Object Explorer, in the Applet tree, click Drilldown Object.
3. In the Drilldown Objects list, add a new drilldown object using values from the following table.

Property	Value
Name	Action - Detail
View	eCalendar Detail View
Hyperlink Field	Id

Property	Value
Source Field	Id
Business Component	Action

4. In the Drilldown Objects list, add a new drilldown object using values from the following table.

Property	Value
Name	Action - Detail Participant
View	Activity Participants View
Hyperlink Field	Id
Source Field	Id
Business Component	Action

5. In the Drilldown Objects list, add a new drilldown object using values from the following table.

Property	Value
Name	Contact - Detail
View	Contact Detail View
Hyperlink Field	Primary Contact Last Name
Source Field	Primary Contact Id
Business Component	Contact

6. Make sure Action - Detail is chosen in the Drilldown Objects list.
7. In the Object Explorer, expand the Drilldown Object tree, and then click Dynamic Drilldown Destination.
8. In the Dynamic Drilldown Destinations list, add a new dynamic drilldown destination using values from the following table.

Property	Value
Name	Action - Detail
Field	Type Category
Value	P
Destination Drilldown Object	Action - Detail Participant

You must complete *Defining Fields in the Business Component* before you can add the dynamic drilldown.

9. In the Applet tree of the Object Explorer, click Control.
10. In the Controls list, add a new control using values from the following table.

Property	Value
Name	Primary Contact Id
Field	Primary Contact Id
Field Retrieval Type	Field Data

11. In the Controls list, add a new control using values from the following table.

Property	Value
Name	Primary Contact Last Name
Field	Primary Contact Last Name
Field Retrieval Type	Field Data

Each control references a drilldown object, so you must define the Hyperlink Field and Source Field property in the Drilldown Object before you define the control. The exception is if the value of the Hyperlink Field or Source Field is Id, then you can define the control first.

12. Test and then deliver your Workspace.

For more information, see *Using Siebel Tools* .

Configuring a Different Icon for the Dynamic Drilldown

This task is a step in *Process of Configuring Drilldown from the Calendar Applet*.

The optional configuration in this topic modifies the link feature so that if the primary contact is an employee, then Siebel CRM displays a different icon.

To configure a different icon for the dynamic drilldown

1. In the Object Explorer, click Business Component.
2. In the Business Components list, locate the Action business component.
3. In the Object Explorer, expand the Business Component tree, and then click Field.
4. In the Fields list, add a new field using values from the following table.

Property	Value
Name	Primary Contact Employee Flag
Join	S_CONTACT. As an alternative, you can use the name of the join specification that you use in *Defining Fields in the Business Component*.
Column	EMP_FLG

5. In the Fields list, modify a predefined field using values from the following table.

Property	Value
Name	Primary Contact Icon
Calculated Value	IIF([Primary Contact Last Name] is NULL and [Primary Contact First Name] is NULL, "", IIF([Primary Contact Employee Flag] = "Y", "<img src='images/icon_copy.gif' border='0'>", "<img src='images/icon_alarm.gif' border='0'>"))
Column	EMP_FLG

Configuring a Different Destination for the Dynamic Drilldown

This task is a step in *Process of Configuring Drilldown from the Calendar Applet*.

The optional configuration in this topic causes the dynamic drilldown to drill down to an employee view if the primary contact is an employee. Otherwise, the drilldown displays a contact view.

To configure a different destination for the dynamic drilldown

1. In the Object Explorer, click Applet.
2. In the Applets list, locate the Activity HI Calendar Applet.
3. In the Object Explorer, expand the Applet tree, and then click Drilldown Object.
4. In the Drilldown Objects list, add a new drilldown object using values from the following table.

Property	Value
Name	Contact - Detail Employee
View	Employee Activity (ERM - Help Desk)
Hyperlink Field	Primary Contact Last Name
Source Field	Primary Contact Id
Business Component	Employee

5. In the Drilldown Objects list, locate the Contact - Detail drilldown object.
6. In the Object Explorer, expand the Drilldown Object tree, and then click Dynamic Drilldown Destination.
7. In the Dynamic Drilldown Destinations list, add a new destination using values from the following table.

Property	Value
Name	Employee View Drilldown. Note that you can use any name.
Field	Primary Contact Employee Flag
Value	Y
Destination Drilldown Object	Contact - Detail Employee

8. If necessary, make sure that the Employee Activity (ERM - Help Desk) view is defined in the ERM Employee ReadOnly Screen.This view comes predefined. It is only necessary to do this step if the view is deleted or modified for some reason:
    a. In the Object Explorer, click Screen.

b. In the Screens list, locate the ERM Employee ReadOnly Screen.
c. Expand the Screen tree, and then click Screen View.
d. In the Screen Views list, locate the Employee Activity (ERM - Help Desk) screen view.

   The name of this screen view contains a special character, so you must enclose the name in double quotes when you create the query.
e. If the query returns an empty result, then add a new screen view using values from the following table.

Property	Value
Name	Employee Activity (ERM - Help Desk)
View	Employee Activity (ERM - Help Desk)
Type	Detail View
Parent Category	Employee List
Viewbar Text	Activities
Menu Text	Employee Activities

17 Configuring Special-Purpose Applets

Configuring Special-Purpose Applets

This chapter describes how configure special-purpose applets such as chart, tree, attachment, and pop-up applets. It includes the following topics:

- *Configuring a Chart Applet*
- *Configuring a Tree Applet*
- *Configuring a Hierarchical List Applet*
- *Configuring a File Attachment Applet*
- *Configuring an Organization Analysis Applet*

Configuring a Chart Applet

This topic describes how to configure a chart applet. It includes the following topics:

- *About Chart Applets*
- *Types of Charts*
- *How Siebel CRM Creates a Chart Applet*
- *Using the Chart Applet Wizard to Create a Chart*
- *Configuring Lists in Chart Applets*
- *Configuring a Chart That Includes Multiple Lines Against One Y-Axis*
- *Configuring a Chart That Includes Two Y Axes*
- *Limiting and Sorting Axis Points*
- *Defining the Physical Appearance of a Chart*
- *Making an X-Axis Label Vertical*
- *Defining the Size of a Chart Control*

Note: The JavaScript D3.js (Data Driven Documents) charting engine is supported in Siebel CRM 18.7 Update and later releases.

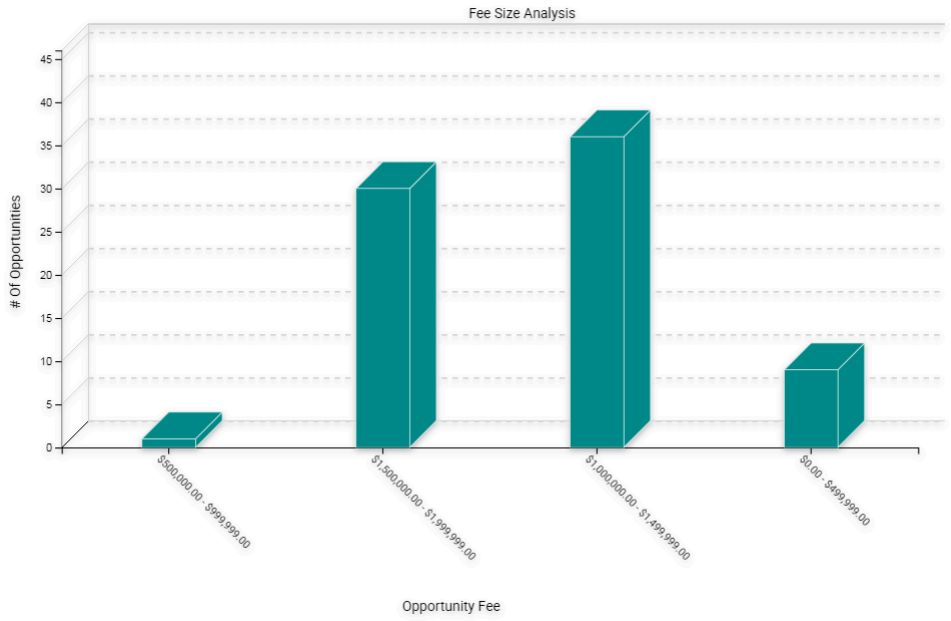
About Chart Applets

A *chart applet* is a type of applet that graphically displays data from a business component in various formats that allow the user to analyze trends, compare categories, and examine other relationships in the data. Note the following about charts:

- You can include any data in a business component in a chart. Aggregated data based on period, multiple attributes, a particular attribute, multiple value fields, a selected record, calculated field or ranges can all be displayed in chart format.

- The data in a chart applet reflects the current query for the business component. The following aggregated functions are supported when building your query: MIN, MAX, SUM, AVG, COUNT, and GROUPBY.
- You can click in the chart to update it with modifications to the query.
- The JavaScript D3.js (Data Driven Documents) charting engine renders charts in the UI.

The following figure shows a chart applet in a view.



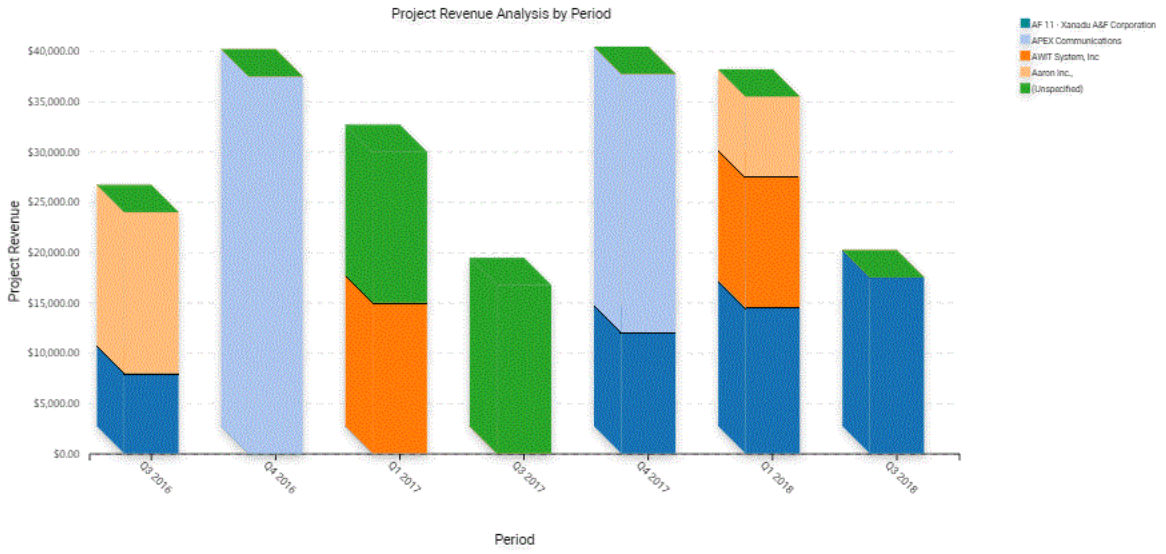
This view is named Opportunity Fee Size Analysis. The object definition for this view is named FINCORP Deal Chart View - Fee Size Analysis. It includes a list and a chart applet. It lists opportunities in the list applet and aggregates them by size in the FINCORP Deal Chart View – Fee Size Analysis chart applet. The chart applet in this view displays the data in the three dimensional bar chart format, by default. The user can choose different chart types from the Type list that Siebel CRM displays in the chart applet. To modify the size of the legend for a chart applet, the user can right-click the legend, and then choose one of the menu items. For more information, see *Types of Charts* and *Considering Factors That Affect Chart Performance*.

Example of a Chart That Includes Three Axes

The following figure displays the Project Revenue Analysis chart, which is an example of a chart that includes three axes. In this chart, Siebel CRM does the following:

- Plots the amount of revenue on the Y data values axis (Project Revenue)
- Displays quarters on the X category axis (Q3 2021, Q4 2021, Q1 2022, Q2 2022, Q3 2022, Q4 2022)
- To identify a different project, Siebel uses each bar color for Z, series, and axis.

In a chart that contains two Y axes, the first Y-axis refers to the vertical axis in the first quadrant of the chart, and the second Y-axis refers to the vertical axis in the second quadrant of the chart.



Axis Terminology

The following table describes each axis in a chart.

Axis	Name	Usage in Bar Chart	Usage in Line Chart	Usage in Pie Chart
X-axis	Category	The horizontal axis, except in a horizontal bar chart, where the X-axis is the vertical axis.	The horizontal axis.	The set of pie slice labels.
Y-axis	Data Values	The vertical axis, except in a horizontal bar chart, where the Y-axis is the horizontal axis along the lower level of the chart.	The vertical axis.	The percentage of the circle that each pie slice occupies, and the corresponding numeric value.
Z-axis	Series	A set of labels in the legend. In the stacked bar or cluster bar charts, each series label corresponds to a bar segment or bar of a color that Siebel CRM displays in each stack or cluster.	A set of labels in the legend. In a line chart, each series label in the legend corresponds to one line.	Siebel CRM charts only the first entry in each series. Do not use a series field with a pie chart.

Types of Charts

This topic describes different types of charts. It includes the following topics:

- *Bar Charts*
- *Line Charts*
- *Pie Charts*
- *Scatter Charts*

Siebel CRM does not support all styles for all chart applets. It uses data from the CHART_TYPE list of values to enter values in a chart type list.

The user can choose different chart types from the Type list that Siebel CRM displays in most chart applets. A chart type includes the following layout options:

- Horizontal bar
- Stacked bar
- Pie
- Line
- Scatter
- Spline
- Combo, which is a combination of a line chart and a bar chart

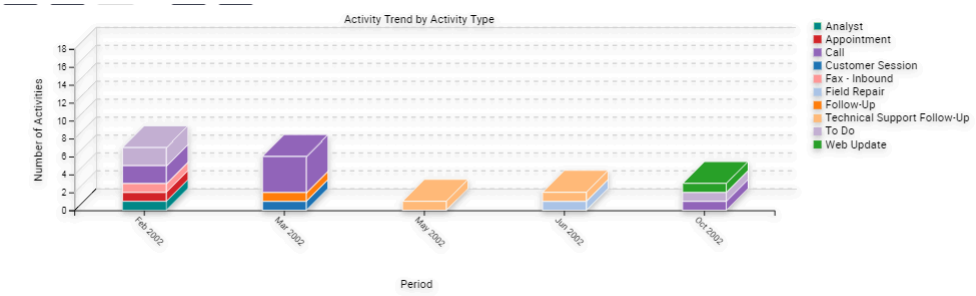
Several of these charts can display data in two or three dimensions. The functionality for a three dimension chart is the same as the functionality of the corresponding two dimensional chart except that the three dimensional chart displays thickness for the bar, line, or pie.

Bar Charts

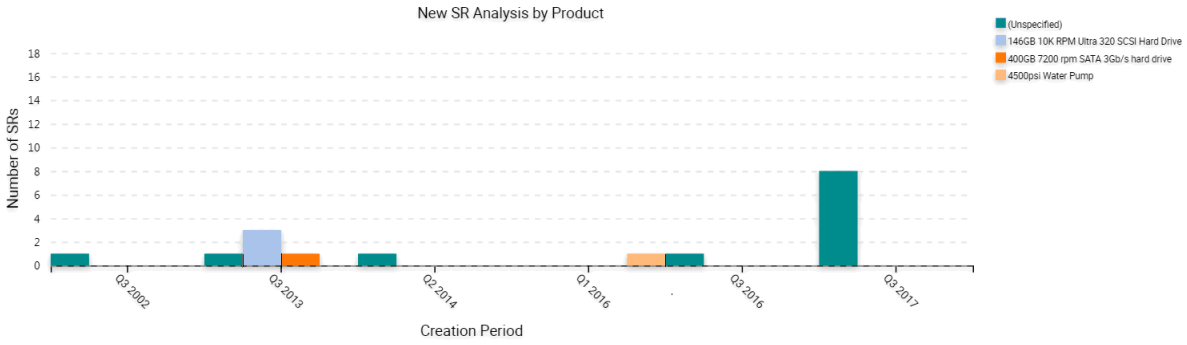
A bar chart compares the difference in data from one category to another category. This topic includes examples of different bar charts.

Three Dimensional Bar Chart

The following figure shows a three dimensional bar chart that divides data from source records into categories and displays the total for each category as a vertical bar.



The following figure shows a cluster of bars for categories rather than a single bar if the chart is configured with a Z series axis.

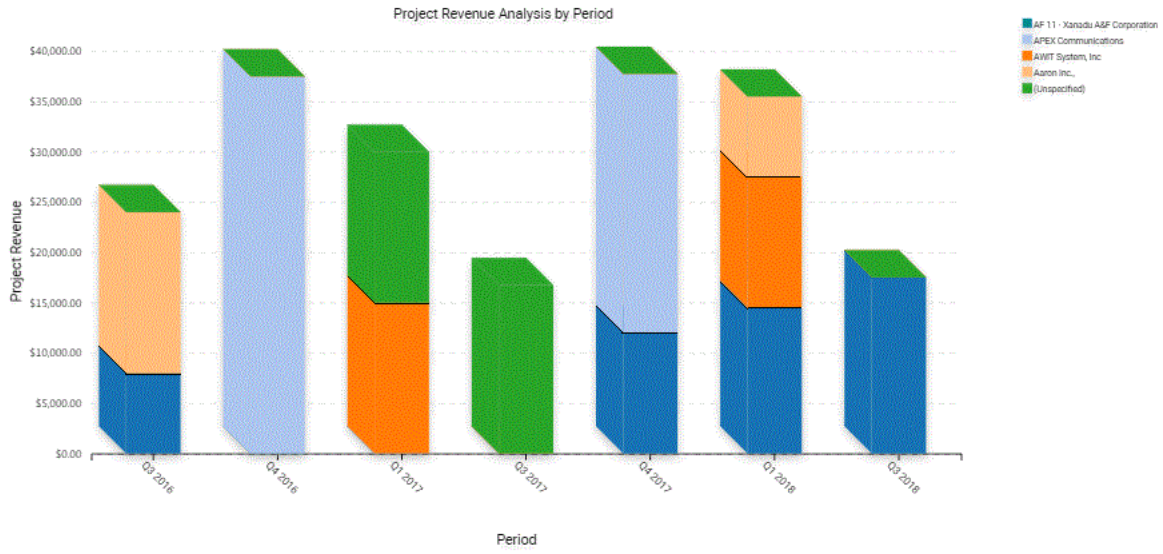


Three Dimensional Stacked Bar Chart

The following figure shows a three dimensional stacked bar chart that normally includes a series axis. The chart displays a single stack of bars for each category. A bar with a different color for each series displays in this stack of bars. A stacked bar chart displays the individual value for each series in the category and the total for the category. In this example, the Project Revenue Analysis chart displays data in the following ways:

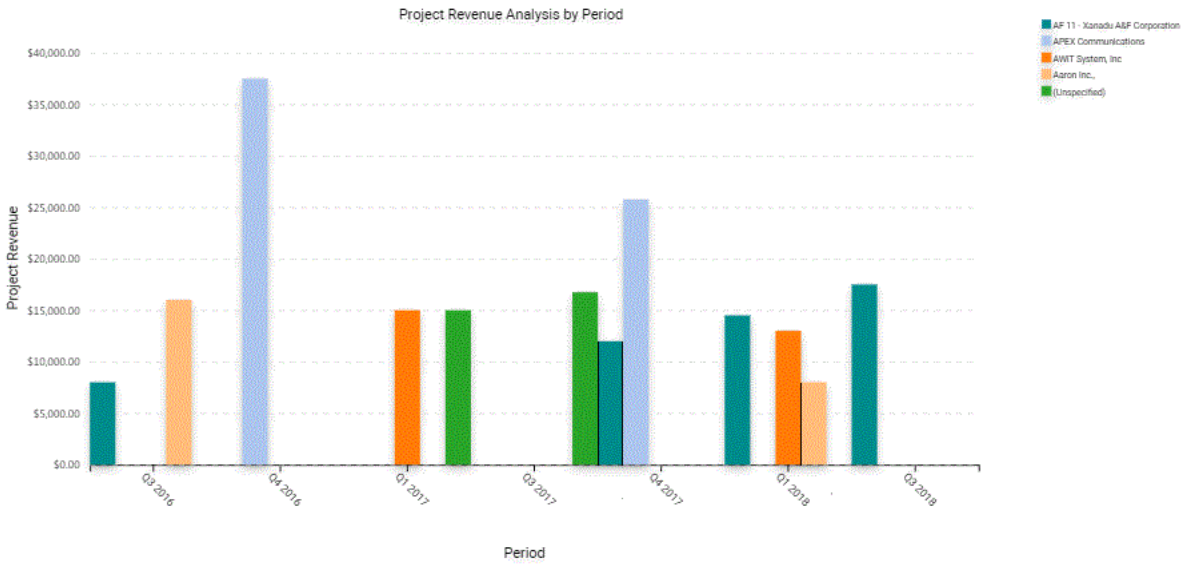
- Data in the values axis corresponds to project revenue.
- Data in the category axis corresponds to a quarter.
- Data in the series axis corresponds to the project name.

Each quarter along the X-axis includes a stack of bars. Each bar in the stack indicates the revenue reached in a quarter. The stacks in each bar indicate individual projects.



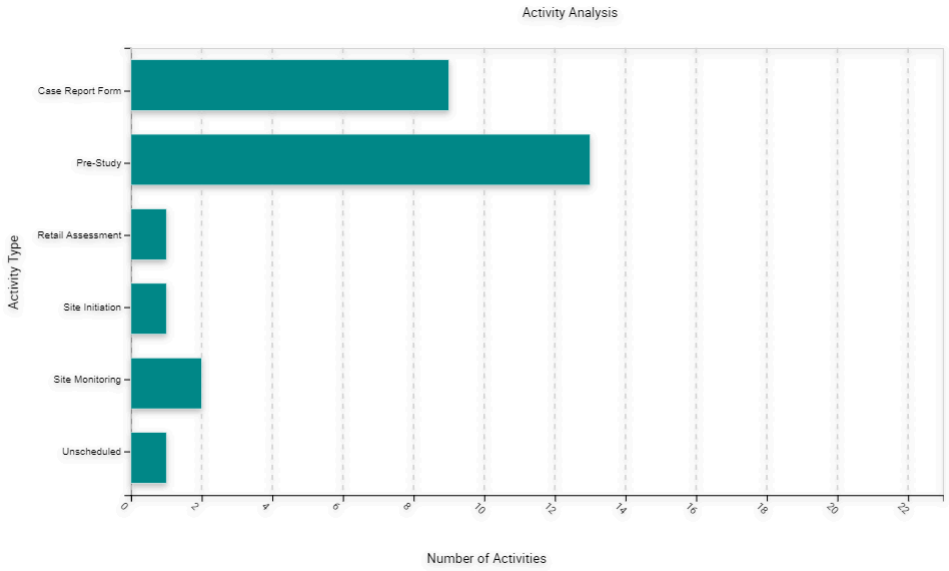
Two Dimensional Bar Chart

The following figure shows a two dimensional bar chart that is functionally equivalent to a three dimensional bar chart except it displays data without the illusion of depth. A two dimensional chart is typically easier to read accurately but might be less visually attractive than the three dimensional chart. If a series axis is present, then the two dimensional bar chart displays bars in a cluster.



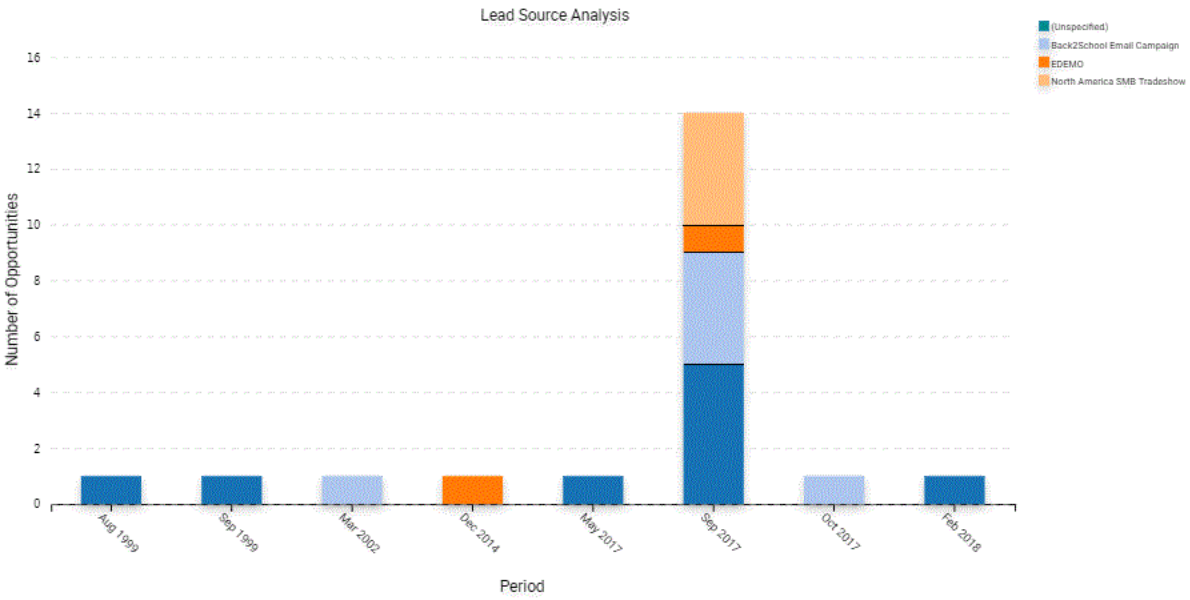
Two Dimensional Horizontal Bar Chart

The following figure shows a two dimensional horizontal bar chart that is functionally equivalent to a two dimensional bar chart except the X-axis and Y-axis are switched.



Two Dimensional Stacked Bar Chart

The following figure includes a two dimensional stacked bar chart that is functionally equivalent to the three dimensional stacked bar chart except it displays without the illusion of depth.

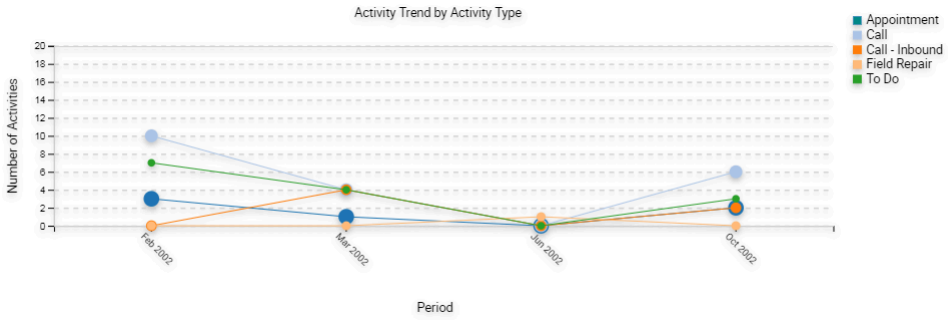


Line Charts

A *line chart* is a type of chart that displays trends across categories or over time. This topic shows examples of different line charts.

Two Dimensional Line Chart

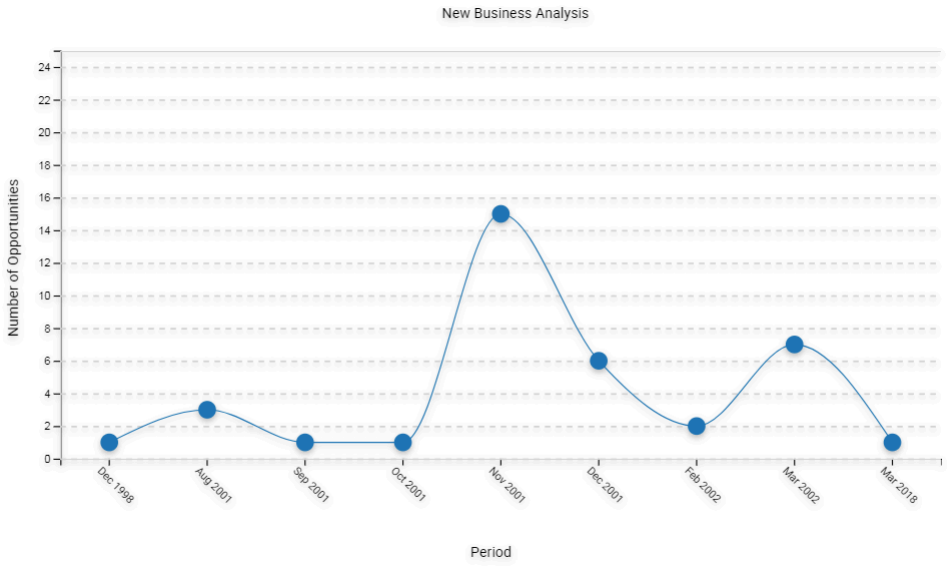
The following figure shows a two dimensional line chart that displays one or more lines plotted against an X-Y grid. If a series axis does not exist, then Siebel CRM displays a single line. If a series axis does exist, then Siebel CRM displays one line for each color in the legend.



Two Dimensional Spline Line Chart

The following figure shows a two dimensional spline line chart that displays one or more lines plotted against the X-Y grid with the points plotted accurately but the line between points smoothed mathematically:

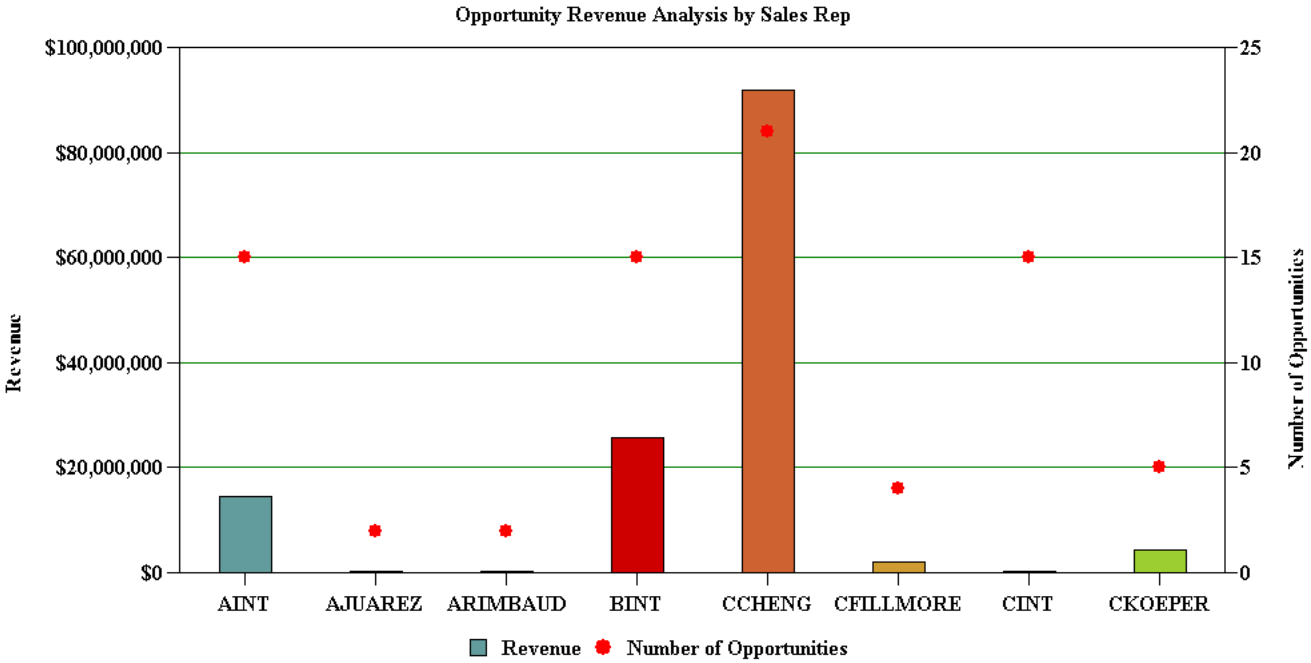
- If a series axis does not exist, then Siebel CRM displays a single line and set of points.
- If a series axis exists, then Siebel CRM displays one line and the corresponding set of points for each color in the legend.



Combo Line Chart

The following figure shows a Combo line chart that displays a single bar chart with superimposed dots. The two charts share the category axis but each chart includes separate data points axes that Siebel CRM displays in the following ways:

- On the start of the graph for the bar chart
- On the end of the graph for the line chart



Pie Charts

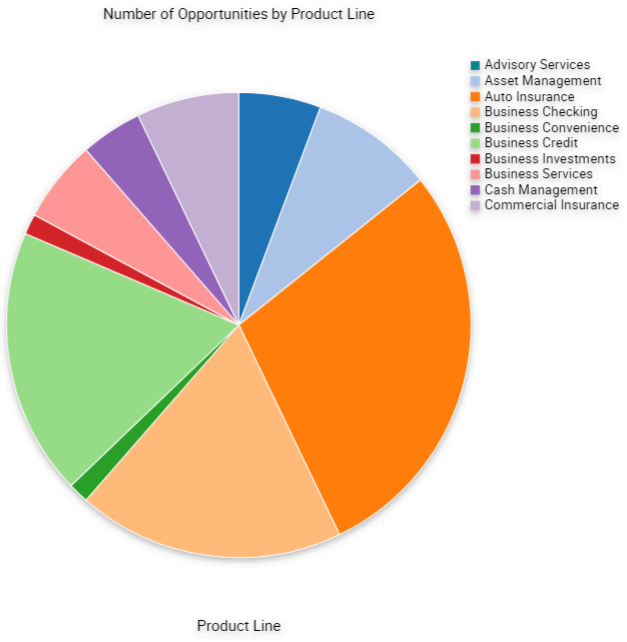
A *pie chart* is a type of chart that compares the relative difference across categories. It divides a circle into segments that represents the percentage of the whole for each category. This topic includes examples of pie charts.

Two Dimensional Pie Chart

The following figure shows a two dimensional pie chart that aggregates data in the records according to category and displays each category as a separate segment in the pie chart.

- The category constitutes the X-axis. It is the set of pie slices and corresponding labels.
- The data points constitute the Y-axis. It determines the relative size of each pie slice as a percentage of the total.

You cannot define a series axis for a pie chart.

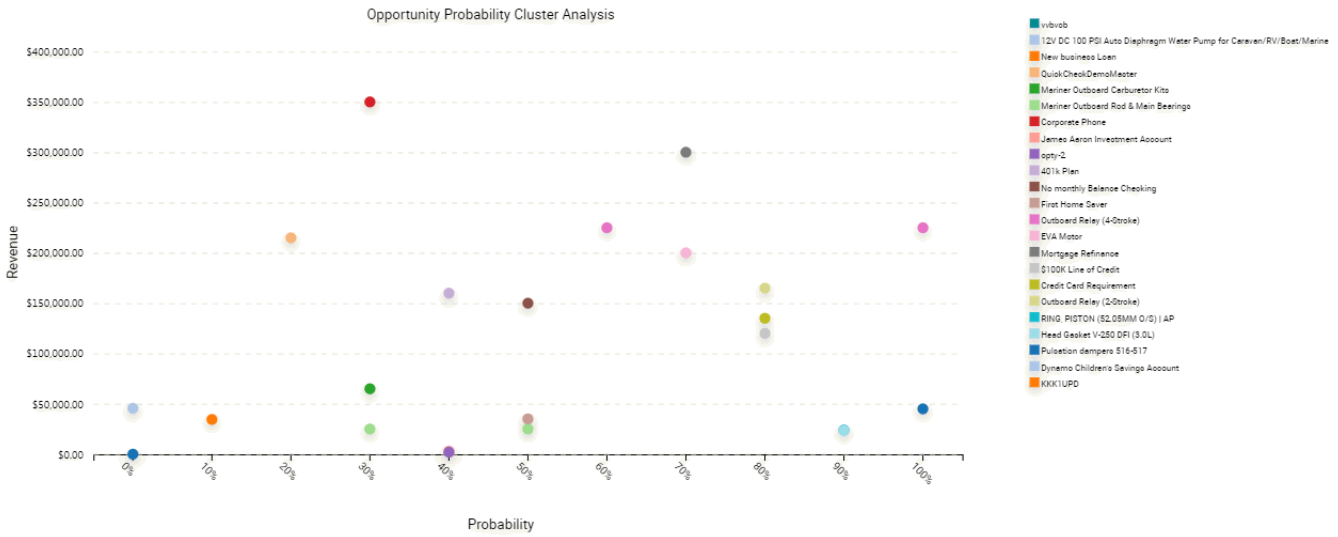


Scatter Charts

A *scatter chart* is a type of that displays the distribution of data across two dimensions, which is useful for probability distribution and other uses. The category axis must contain only numeric data, so you cannot convert the two dimensional scatter chart to other chart types, such as the bar chart, line chart, or pie chart. For this reason, the following conditions apply for the two dimensional scatter chart:

- Does not display in the Type list
- Does not include a Type list

The following figure includes a two dimensional scatter chart.

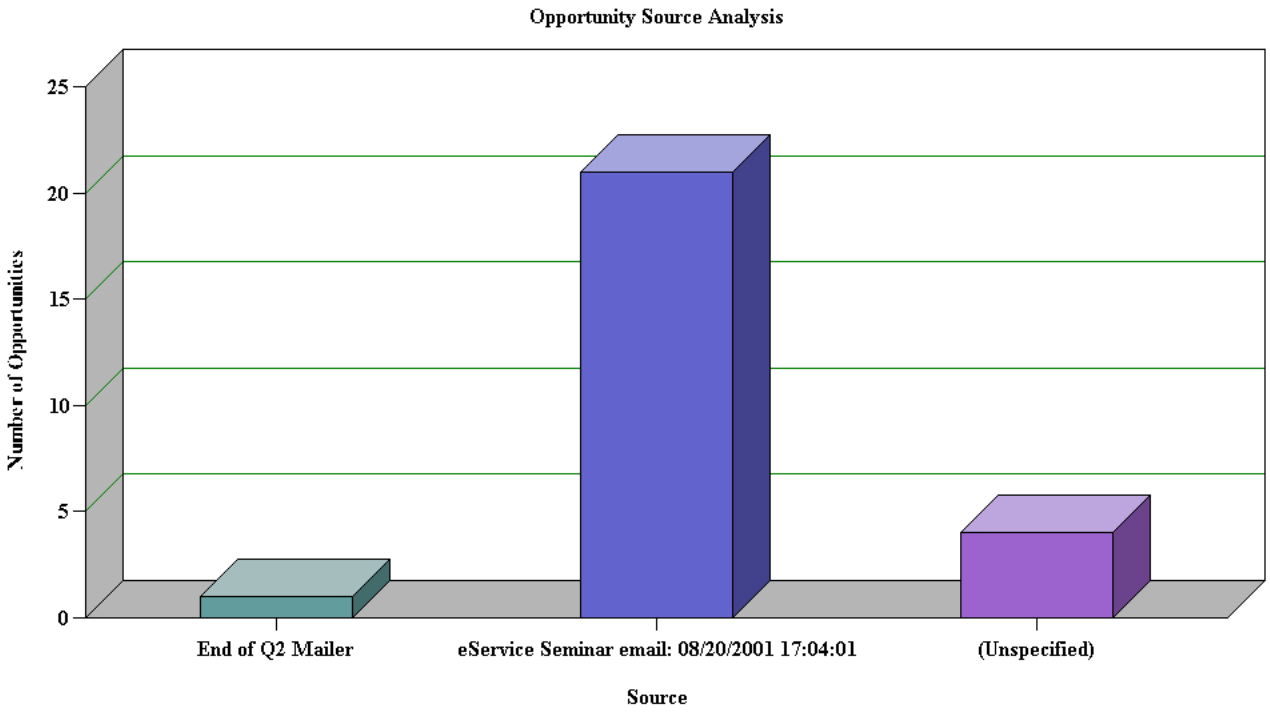


How Siebel CRM Creates a Chart Applet

Siebel CRM builds a chart as an applet that contains one or more Chart object definitions. A Chart is a child of an applet. The Business Component property of a chart applet identifies the business component that provides data that Siebel CRM displays in a chart applet. Records in this business component are subject to the current view, the current query, and visibility requirements. Business component fields provide the data for the category, data point, and series axes in a chart applet. The properties of the chart object define the relationship between axes and fields. Siebel CRM also supports the building of charts based on Virtual Business Components (VBC). This allows charts to be built using data from external data sources.

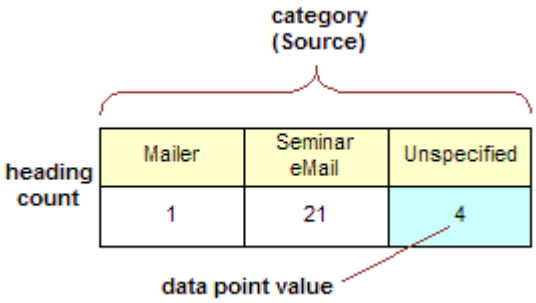
A single bar or line graph is the simplest form of chart applet. It contains no series axis and only a category field and a data point field are defined. Siebel CRM plots pairs of category and data point field values as points or bars. If multiple records use the same category value, then Siebel CRM adds together their data point values.

The following figure shows opportunities on the data point axis that Siebel CRM plots against the source of the opportunity on the category axis. Example sources include referral, magazine article, web site, and so on. To create the data required for the line, Siebel CRM checks the Source field in each record and tallies the number of opportunities for each distinct source value.

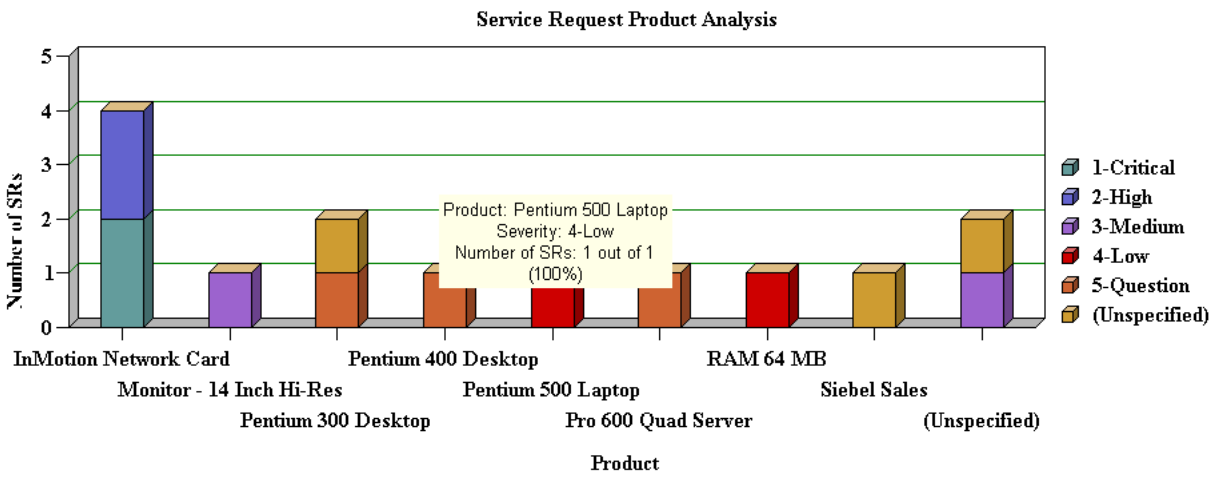


The following table and figure shows the result, which is a two row temporary table that includes a column for each source.

Mailer	Seminar eMail	Unspecified
1	21	4

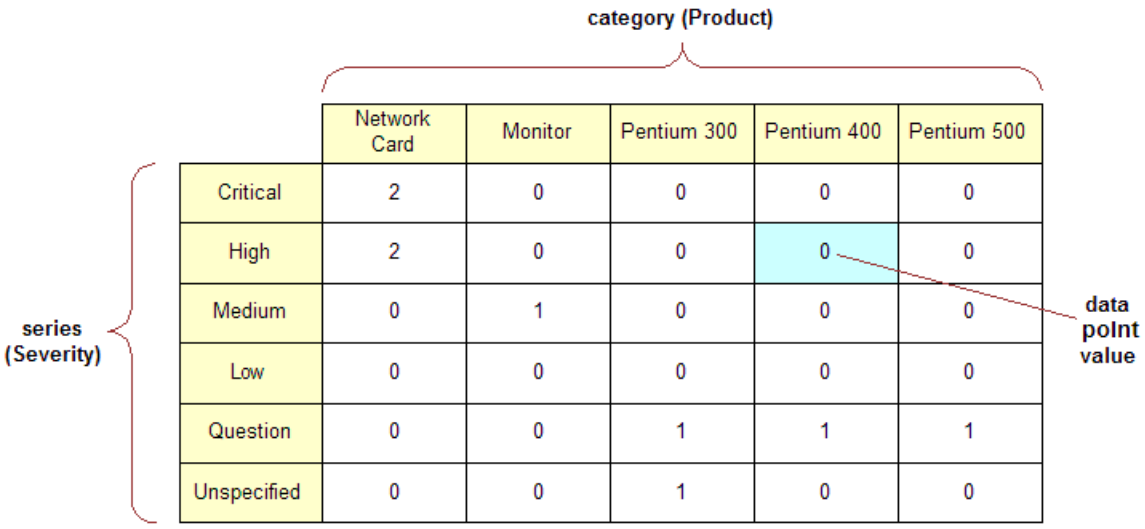


The following figure includes a multiple line chart where Siebel CRM adds a row to the temporary table for each line in the series.



The following table and figure shows the temporary table for a multiple line chart.

<Empty>	Network Card	Monitor	Pentium 300	Pentium 400	Pentium 500
Critical	2	0	0	0	0
High	2	0	0	0	0
Medium	0	1	0	0	0
Low	0	0	0	0	0
Question	0	0	1	1	1
Unspecified	0	0	1	0	0



Properties of the Chart Object

To create the data mapping from the business component to the chart applet, you must define the properties of the Chart object that this topic describes. For situations where these properties are configured differently, see *How Siebel CRM Creates a Chart Applet* and *Siebel Object Types Reference* .

The following table describes properties of the Chart Object.

Property	Description
Category Field	Contains the name of a text or date field in the business component except for a scatter chart that uses a numeric category field. When Siebel CRM scans the business component records, it maps the different values in this field to different categories. It displays values on the X-axis labels of the chart.
Data Point Field	Can contain the name of a numeric field in the business component or is not defined: • If it is defined, then Siebel CRM adds the value in this field in each record to the total for the value of the category field in the same record. • If it is not defined, then Siebel CRM increments the count for the corresponding category field. These counts or totals determine the height along the Y-axis of a bar or line point for each unique category field value in the line. Rather than a total or a count, some other function that is defined in the Data Function property can determine how to use the data in the Data Point Field property.
Series Field	Contains the name of a text field in the business component, or is not defined. When Siebel CRM scans the business component records, it maps the different values in this field to different lines. It displays these values on the legend labels of the chart. If the number of series exceeds 50 when the user runs the chart, then Siebel CRM displays an error message. The user might be required to run another query that results in a display that does not exceed 50 series.
Function	Determines how Siebel CRM converts data point field values into the cell values of the new table. The following values are available: • **Sum.** Simple addition. • **Count.** The number of occurrences of a cell value. • **Average.** The average value for each record. • **Plot.** Similar to Count except that if a cell is empty, then the value is NULL instead of 0.

Using the Chart Applet Wizard to Create a Chart

You can use the Chart Applet Wizard to create a new chart applet. For more information, see *Example of a Chart That Includes Three Axes*.

To use the Chart Applet Wizard to create a chart

1. In Siebel Tools, click the File menu, and then click New Object.
2. In the New Object Wizards dialog box, Click the Applets tab, click the Chart Applet icon, and then click OK.
3. In the General dialog box of the Chart Applet Wizard, define the following properties, and then click Next:

- Project
- Business Component
- Name
- Display Name

4. In the Y Axis dialog box, define the properties for the Y-axis, and then click Next.
5. Follow the instructions in the dialog box.

   When you define the Data point field, Siebel Tools enters values in the Titles section of the dialog box. For more information, see *Properties of the Chart Object*.
6. In the X-axis dialog box, define the properties for the X-axis, and then click Next.

   For more information, see *Properties of the Chart Object*.
7. Optional. In the Z Axis dialog box, define the properties for the Z-axis, and then click Next.

   For more information, see *Properties of the Chart Object*.
8. In the Chart Title dialog box, enter a title, and then click Next.
9. In the Web Layout - General dialog box, choose the Siebel Web Template to use for the base read-only mode, and then click Next.
10. In the Finish dialog box, review the information, and then click Finish.

    The Chart Applet Wizard creates the required object definitions and sets the property values according to the information you entered in the wizard. The Web Applet Layout Editor opens and allows you to map controls to placeholders in the web template.

    For more information, see *Editing the Layout of a Web Page*.
11. Add list controls to the web template for the applet.

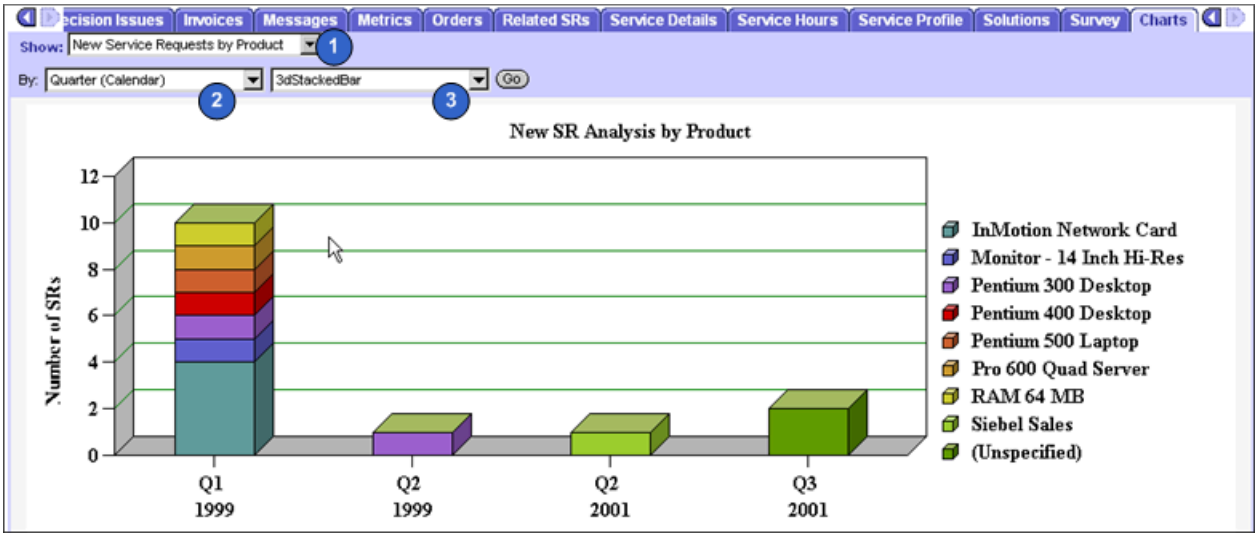
    Siebel CRM displays these controls in the chart applet in the Siebel client. For more information, see *Configuring Lists in Chart Applets*.
12. Add the applet to a view.

    For more information, see *Editing the Layout of a View*.
13. Test and then deliver your Workspace.

    For more information, see Using Siebel Tools.

Configuring Lists in Chart Applets

A chart applet typically provides one or more lists that allow the user to determine how Siebel CRM displays or uses data. The following figure shows an example of a chart applet.



As shown in this figure, the following types of lists are available:

1. **Show list.** Allows the user to modify data that Siebel CRM displays on the Y-axis. For more information, see *Configuring the By List of a Chart Applet*.
2. **By list.** Allows the user to modify data that Siebel CRM displays on the X-axis. For information, see *Configuring the By List of a Chart Applet*.

   Second By list. Allows the user to choose the source field that provides data for the Z-axis. For more information, see *Configuring the Second By List of a Chart Applet*.
3. **Type list.** The most common of the four lists. Siebel CRM displays it in most chart applets and allows the user to choose a different type of chart for the same data, such as a pie chart instead of a bar chart, or a two-dimensional line chart instead of a three-dimensional chart. For more information, see *Types of Charts*.

   You can use a comma separated list of chart type names in the Picklist Types property of the chart object definition to define options for the type list. For example:

   ```
   3dBar,3dStackedBar,3dPie,3dHorizBar,2dBar,2dStackedBar,2dPie,2dHorizBar
   ```

   Siebel CRM does not allow spaces between the elements in the comma separated list.

   The default type is the chart type that Siebel CRM displays the first time it displays the chart. This default is defined in the Type property. A chart that does not include a type list uses the Type property to define the chart type. The user cannot modify a chart that does not include a type list.

Required Properties of the Lists

Each list in a chart applet requires a corresponding ComboBox control that is a child object of the chart applet. The following table describes the required properties for each type of list.

Type of List	Control Name Property	MethodInvoked Property
Type	ChartPicktype	PickChartType
Show	ChartPickfunction	PickYAxis

Type of List	Control Name Property	MethodInvoked Property
By	ChartPickby	PickXAxis
Second By	ChartPickby2	PickZAxis

Configuring the Show List of a Chart Applet

The show list allows the user to modify data that Siebel CRM displays on the Y-axis. It displays a list of field and function combinations that determine the values that Siebel CRM plots along the Y-axis. The title of the Y-axis mirrors the label of the show list.

To configure the show list of a chart applet

1. In Siebel Tools, display the Chart object type and all child objects of the Chart object type. A chart is a child of an applet.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. In the Object Explorer, click Applet.
3. In the Applets list, locate the applet you must modify.
4. In the Object Explorer, expand the Applet tree, and then click Chart.
5. In the Charts list, locate the chart you must modify.
6. Define chart properties using values from the following table.

Property	Description
Data Point Field	Enter a comma separated list of source fields: • Enter one field for each entry that Siebel CRM displays in the show list. • The default value is the first entry in the list. • If you only enter one field name, then this entry applies to all functions in the list.
Data Function	Enter a comma separated list that includes the following values: SUM, COUNT, AVERAGE, or PLOT. PLOT configures Siebel CRM to get Y values from the values in the source field. The order of items in this list determines the association with a data point field and title, which is the list function. If the Data Function property contains fewer elements than the list of names that the Picklist Functions property defines, then Siebel CRM substitutes the values in the Data Function property with the following values: `Sum,Count,Average,Plot`
Picklist Functions	Enter a comma separated list of Y-axis titles, which is the text that Siebel CRM displays in the list. The order of values in the Picklist Functions property determines the association with a data point field and data function.

Example of a Show List That Is Defined Explicitly

You can define a show list with an explicit format that displays the following choices:

- Number of Opportunities
- Opportunity Revenue
- Opportunity Expected Revenue

The following table describes the properties and their values that are required to implement this example.

Property	Value
Picklist Functions	Number of Opportunities, Opportunity Revenue, Opportunity Expected Revenue
Data Function	Count,Sum,Sum
Data Point Field	Name,Revenue,Expected Revenue

In this example, the values in each comma separated list creates the following relationships between the properties:

- Number of Opportunities performs a Count function on the Name field.
- Opportunity Revenue performs a Sum function on the Revenue field.
- Opportunity Expected Revenue performs a Sum function on the Expected Revenue field.

Example of a Show List That Is Defined Implicitly

It is recommended that you explicitly define the show list. Siebel CRM retains the ability to implicitly define the Show list for backwards compatibility with earlier versions of Siebel CRM. It is more restrictive.

The Lead Source Analysis chart in the Opportunity New Business Analysis view in Siebel Sales (Oppty Chart Applet - New Business) is an example of a Show list that Siebel CRM defines with an implicit format and function list. This list displays the following choices:

- Number of Opportunities
- Opportunity Revenue
- Average Opportunity Revenue.

The following table describes the properties and their values that are required to implement this example.

Property	Value
Picklist Functions	Number of Opportunities,Opportunity Revenue, Avg Opportunity Revenue
Data Function	Count
Data Point Field	Revenue

The value of Revenue in the Data Point Field property applies to all entries in the list.

In this example, the values in each comma separated list creates the following relationships between the properties:

- Number of Opportunities performs a Count function on the Revenue field.
- Opportunity Revenue performs a Sum function on the Revenue field.
- Avg Opportunity Revenue performs an Average function.

The value of the Count in the Data Function property is not required. It can be empty.

If the number of entries in the Data Function property is not the same as the number of entries in the Picklist Functions property, then Siebel CRM supplies the following predefined list in the Data Function property:

- Count,Sum,Average,Plot

Configuring the By List of a Chart Applet

The by list allows the user to modify data on the X-axis:

- In a period chart, Siebel CRM enters data into the by list with different date periods. The user can choose from a list of possible X-axis date periods for calendar data. This calendar data includes day, week, month, quarter, and year. You can define these options in the Picklist Periods property of the chart object.
- If you define a list of source fields rather than a single source field, then the list allows the user to choose the source field that provides data for the X-axis.
- The user can invert the X-axis and the Z-axis. The user can view the data from a source field in a business component that displays along the X-axis or Z-axis according to the selection the user makes in the list.

To configure the by list of a chart applet

- Define the Category Field property of the chart object.

Calendar Increments in the List and the X-Axis

If the Category Field property contains the name of a single field that is a DTYPE_DATE data type, then the X-axis displays calendar increments and the chart is a period chart. In this situation, Siebel CRM enters data into the list with calendar increment options, including user defined periods, such as Day, Week, Month, Quarter, and Year. You can administer these increments in the Periods view of the Administration - Data screen.

For example, in the New Business Analysis chart the category field is Created, which is the date that Siebel CRM created the opportunity record. The increment that the user chooses in the by list determines the date increments that the category axis contains.

Text labels in the X-Axis and Category and Series Field Names in the List

If the Category Field property contains the name of a single text field from the business component, and if a series field is defined in the Series Field property, then the by list includes the names of the category field and the series field. The user can choose either field to update the X-axis with labels from the contents of that field. The unchosen field provides labels for the legend box. The legend box is the Z-axis. The default value is the category field and Siebel CRM initially displays it on the X-axis.

For example, the chart in the Service Request Product Analysis view in the Siebel Service application includes a Product category field and a Severity series field. When Siebel CRM initially displays the chart, the X-axis labels are product names and the legend labels are severity levels. Siebel CRM displays the Product and Severity field names in the by list. The severity allows the user to display severity levels in the X-axis and product names in the legend.

Text Labels in the X-Axis and Multiple Field Names in the List

If the Category Field property contains a comma separated list of field names, then Siebel CRM displays this list in the by list. The user chooses the field that provides data for the X-axis. The default value is the first value in the list. You must not include an empty space before or after a field name in the list.

Numeric Values in the X-Axis and No List

If the Category Field property contains the name of a single numeric field, then the X-axis includes numeric increments, similar to the process of creating increments for the Y-axis. In this situation, Siebel CRM does not display the by list.

For example, the Probability Cluster Analysis chart in the Opportunity Probability Cluster Analysis view includes the Rep % category field, which is the probability of a sale. In this chart, Siebel CRM plots probability against the X-axis, the X-axis increments are percentages from 0% to 100%, and Siebel CRM displays no by list.

Configuring the Second By List of a Chart Applet

You can configure the second by list.

To configure the second by list of a chart applet

- Define the Series Field property of the chart object.

The following values in the Series Field property in the chart object determines the behavior of the second by list:

- If the Series Field property is empty, then Siebel CRM maps all records into a single series.
- If the Series Field property contains the name of a field from a business component, then Siebel CRM includes labels on the Z-axis that it gets from the contents of that field.
- If the Series Field property contains a comma separated list of field names, then Siebel CRM displays this list of fields in the second by list. The user chooses the field that provides data for the Z-axis. The default value is the first value in the comma separated list.

Configuring a Chart That Includes Multiple Lines Against One Y-Axis

This topic includes an example that defines different combinations of the source field and function to determine how Siebel CRM plots a chart with multiple lines against the same Y-axis. Siebel CRM displays the name for each line in the legend. For example, it can display revenue, expected revenue, and net profit as superimposed lines on the same line graph.

To configure a chart that includes multiple lines against one Y-axis

1. Complete Step 1 through Step 5 in the procedure *Configuring the Show List of a Chart Applet*.
2. Define properties of the chart object, using values from the following table.

Property	Description
Data Point Field	Create a comma separated list of source fields, one for each line that Siebel CRMdisplays in the graph.

Property	Description
Data Function	Create a comma separated list that includes some of the following function names: SUM, COUNT, AVERAGE, or PLOT. PLOT indicates that Siebel CRM gets the Y values directly from the values in the source field. The list of function names must include the same number of entries that the Data Point Field list includes. The order in the list in the Data Function property determines the association with the data point field and title.
Picklist Functions	Create a comma separated list of Y-axis titles. Items in this list define the individual lines in the Legend. The list of titles must include the same number of entries that Siebel Tools displays in the list in the Data Point Field property. The order in the list determines the association with the data point field and data function.
Series Field	This property must be empty. Remove any existing values from the Series Field property. If the Series Field property contains a value, then Siebel CRM converts the multiple lines to a Z-axis.
Multi Data Point	Set to TRUE to plot multiple lines.

3. In the Applet Web Editor, remove the Show combo box and the label for the Show combo box.

Configuring a Chart That Includes Two Y Axes

You can define a chart that includes two lines that Siebel CRM plots against different Y axes. Siebel CRM displays one line at start of the graph and the other line at end of the graph. You can use any field or function combination for the first Y-axis or for the second Y-axis.

To configure a chart that includes two Y axes

1. Complete Step 1 through Step 5 in *Configuring the Show List of a Chart Applet*.
2. Define properties of the chart object, using values from the following table.

Property	Description
Data Point Field	Define two fields that are separated by a comma: • The first field defines the first Y-axis. • The second field defines the second Y-axis.
Data Function	Define two functions that are separated by a comma: • The first field defines the first Y-axis.

Property	Description
	• The second field defines the second Y-axis.
Type	Set to Combo.

Limiting and Sorting Axis Points

You can limit the number of X-axis or Z-axis labels to a predefined number. The X-axis defines the category and the Z-axis defines the series. You can use this feature to display only the *N* highest or *N* lowest values for a field or calculated Y value. For example, to display the 10 highest revenue accounts, you can chart the Revenue field in descending order and limit the X-axis to 10 data points.

To limit and sort axis points

1. Complete Step 1 through Step 5 in procedure *Configuring the Show List of a Chart Applet*.
2. In the Object Explorer, expand the Chart tree, and then click Chart Element.
3. In the Chart Elements list, define properties of the axis label chart element, using values from the following table.

Property	Description
Divisions	Defines the X-axis or the Z-axis. Enter an integer to limit the number of X-axis or Z-axis labels. Siebel CRM limits the number of labels it displays to the number you enter. Note the following: • Make sure the AxisId property is equal to XAxis or ZAxis. • Make sure the Type property is equal to AxisLabel.
Sort Specification	Defines the Y-axis. Enter **`Ascending`** or **`Descending`**. Note the following: • Make sure the AxisId property is equal to YAxis. • Make sure the Type property is equal to AxisLabel.

Sorting the Y-Axis

You can sort the Y-axis.

To sort the Y-axis

- Create a sort specification on the Y-axis.

  This sort specification is independent of limiting the number of X-axis or Z-axis divisions. A sort specification on Y orders the data points regardless of if you limit or do not limit the display to the first *N* points. You cannot set a number of X-axis or Z-axis divisions without setting a sort specification on Y.

Sorting on the X-Axis or Z-Axis

You can sort the X-axis or Z-axis labels.

To sort on the X-axis or Z-axis

1. Complete Step 1 through Step 5 in *Configuring the Show List of a Chart Applet*.
2. In the Object Explorer, expand the Chart tree, and then click Chart Element.
3. Set the Sort Specification of the chart element in the X-axis or Z-axis label.

   For example, if the X-axis displays country names, then sort the names so that they are in alphabetical order. This is different from sorting on Y-axis values from a business component field or function according to the field where these values are numeric.

Defining the Physical Appearance of a Chart

You can define the physical appearance of a chart.

To define the physical appearance of a chart

1. Complete Step 1 through Step 5 in *Configuring the Show List of a Chart Applet*.
2. In the Object Explorer, expand the Chart tree, and then click Chart Element.
3. Define the Type property using values from the following table.

Property	Description
AxisLabel	Displays an axis label along each axis with one label for each division of the axis. You cannot define more than 49 labels on the X-axis. If you define more than 49 labels, then Siebel CRM does not display any of these additional labels.
AxisTitle	Displays a title along each axis with one title for each axis.
Title	Displays a large text string. typically displays this text at the start of the chart.

4. (Optional) Change the chart color by using a custom theme.

   Siebel Open UI does not read the color or property of the Charts control. Charts are rendered using SVG, hence you can specify or change the chart color by modifying the custom theme. To configure and control the color and font information for charts using CSS, the chart control provides the CSS class name at every level. For more information on how to add a custom theme, see the topic about customizing themes in *Configuring Siebel Open UI* .

   For example, to modify the default color provided by Siebel Open UI to color code AFAFAE for the first series in the Bar Chart Applet, then modify the CSS by adding the following to the custom theme:

```
.siebui-charts-barchart .siebui-barchart-bar1 {
 fill: #AFAFAE;
}
```

   Note that you can configure series color up to 20 bar. Siebel Open UI inserts the CSS class again from siebui-barchart-bar1 for anything greater than 20 bar and so on.

Using Properties of the Chart Element That Apply To the X-Axis Label

If you define a list of X-axis source fields, then do not use the following properties of the Chart Element that apply to the X-axis label. These properties are relevant only for one X-axis field:

- Coordinates
- Display Format
- Divisions
- List Of Values
- Sort Specification
- Text

Defining the Text of the X-Axis or Z-Axis Title

If the by combo box includes a list of source fields, then Siebel CRM determines the text of the X-axis or Z-axis title dynamically from the combo box selection. It overrides the value in the Text property in the AxisTitle chart element for the X-axis or Z-axis when it displays the chart in the client.

Making an X-Axis Label Vertical

You can make an X-axis label vertical so that one label does not overlap another label.

To make an X-axis label vertical

1. Complete Step 1 through Step 5 in the procedure in *Configuring the Show List of a Chart Applet*.
2. In the Object Explorer, expand the Chart tree, and then click Chart Element.
3. In the Chart Elements list, locate the chart element that contains properties described in the following table.

Property	Value
Axis Id	XAxis
Type	AxisLabel More than one XAxis element might exist. The Vertical property only applies to an element whose Type property is set to AxisLabel.

4. Set the Vertical property to TRUE.

Defining the Size of a Chart Control

You can define the size of a chart control.

To define the size of a chart control

- Define properties for the chart control using values described in the following table.

Property	Value
HTML Width	Set the value in pixels. The default value is 1012.
HTML Height	Set the value in pixels. The default value is 560.

Configuring a Tree Applet

This topic describes how to configure a tree applet. It includes the following information:

- *Overview of Configuring a Tree Applet*
- *Using the Tree Applet Wizard to Create a Tree Applet*
- *Configuring a Tree Node*
- *Using the Applet Web Template Editor to Add a Tree Control*
- *Configuring a Recursive Tree Applet*

For more information, see *About Tree Applet Templates* and *Configuring Icons in a Tree Applet*.

Overview of Configuring a Tree Applet

A *tree applet* is a type of applet that you can use to create an explorer view that allows the user to navigate hierarchically through a structured list of records of related business components. It displays hierarchically structured information in an expandable tree control. Siebel CRM displays the tree control in a frame at start of the applet. It displays detailed information for a chosen tree node in the details applet. Separate vertical frames allow the user to scroll through the contents of the tree applet independently from the detail applet. This is important because the tree structure can grow very large in length and width.

A *tree item* includes any of the following objects. Siebel CRM displays these objects in a tree:

- Root
- Branch
- Leaf

A *tree node* is a repository tree node. The tag specifies the placeholder for a tree item. For more information, see *About Siebel Tags*.

A tree control can include repository tree nodes and field values as elements in the tree. Siebel CRM displays the following:

- Name for a tree node
- Field values for tree items

Example of a Tree Applet

To view an example of a tree applet, do the following:

1. Open Siebel Call Center.
2. Click the Service screen tab, and then the Explorer link.

   Siebel CRM displays the SR Tree Applet in a frame before the interface and the Service Request List Applet in a frame after the interface.

A tree applet in an explorer view operates in a way that is similar to how the Object Explorer and Object List Editor operates in Siebel Tools. The user can expand and collapse folders in the tree applet and view the records in the folder in the list applet. The hierarchy in the tree applet represents a parent-child relationship between records of different business components.

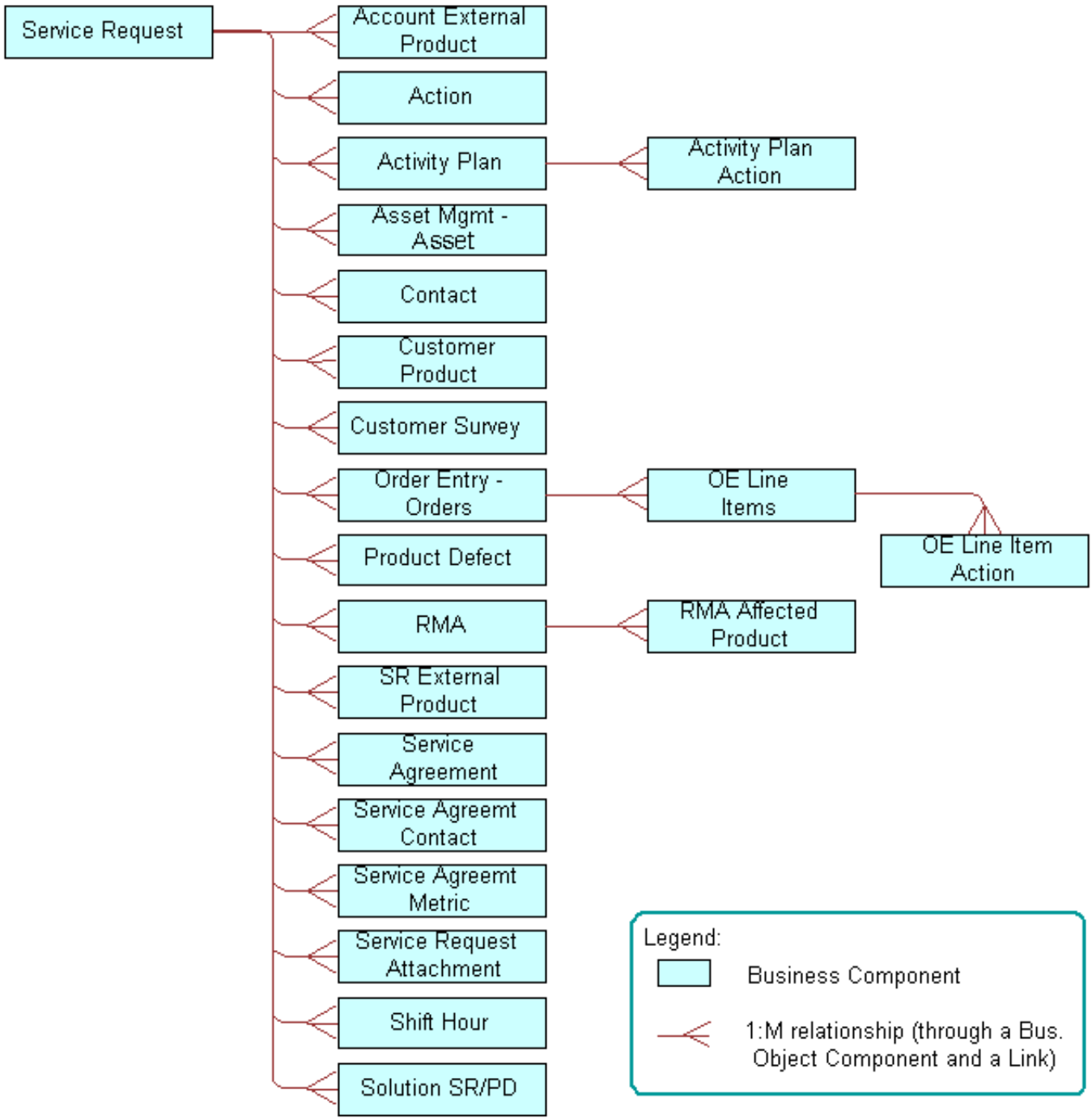
For example, if the user expands a document tree in the Service Requests tree, such as the 1-49119-Claim-New Claim document, then Siebel CRM displays a set of folders that it positions hierarchically beneath the service request. Note the following:

- These folders include Activities, Attachments, Change Requests, Solutions, and so on.
- If the user expands one of these child folders, then Siebel CRM displays a list of records that represent the corresponding business component.
- If the user expands the folder for a service request, and then expands the Activities folder beneath it, then Siebel CRM displays a list of records that constitute the set of activities for that service request. In the parent-child relationship between service requests and activities, these activity records are child records of the parent service request record that is expanded.
- The user can add or associate child records of various kinds to a parent record. For example, to associate a solution record from an association applet, the user can navigate down through the hierarchy to the Solutions folder, click the list applet, and then choose New Record from the applet menu. The product solution record becomes a detail record of the service request.

Relationships Between Business Components, Business Objects, and Tree Applets

A tree applet in an explorer view uses the set of parent-child relationships defined in the business object that Siebel CRM assigns to the view. A business object represents a business model or entity-relationship diagram and specifies the set of parent-child relationships with the business components that the business object references. This configuration makes it possible to arrange the records of these business components hierarchically. For more information, see *Business Objects and Business Components, Views, and Screens*.

The following figure shows the relationships and objects that the Service Request business object contains.

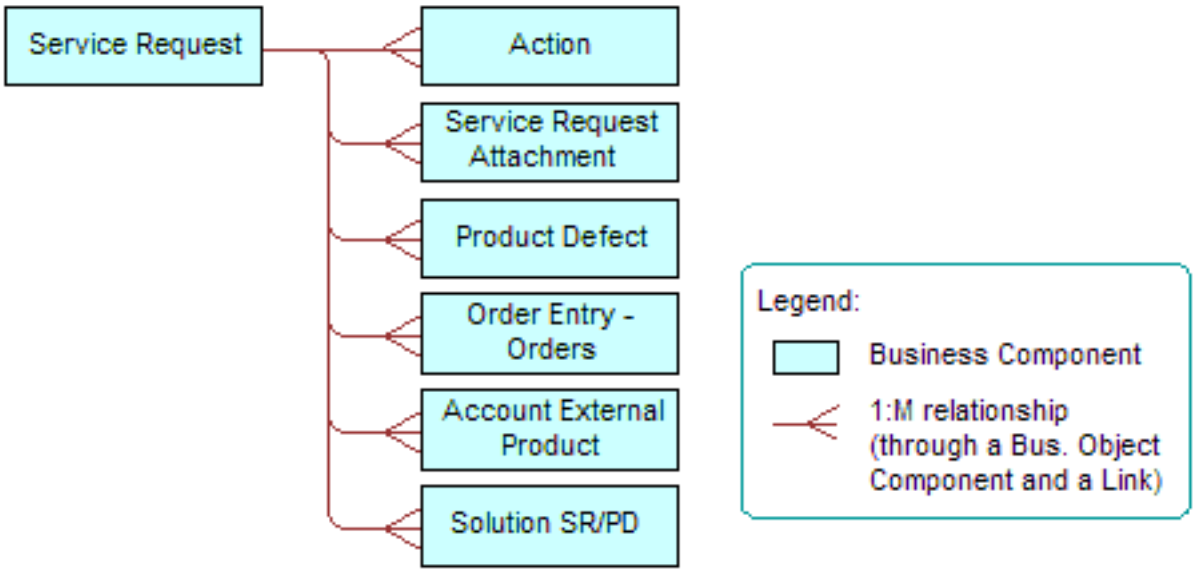


As shown in this figure:

- There is one-to-many relationship between Service Request and the following objects: Account External Product, Action, Activity Plan, Asset Mgmt - Asset, Contact, Customer Product, Customer Survey, Order Entry - Orders, Product Defect, RMA, SR External Product, Service Agreement, Service Agreement Contact, Service Agreement Metric, Service Request Attachment, Shift Hour, and Solution SR/PD.
- There is one-to-many relationship between Activity Plan and Activity Plan Action.
- There is one-to-many relationship between Order Entry - Orders and OE Line Items.

- There is one-to-many relationship between OE Line Items and OE Line Item Action.
- There is one-to-many relationship between RMA and RMA Affected Product.

The following figure shows the relationships and objects in the Service Request business object that Siebel CRM uses in the Service Request Explorer View.



As shown in this figure, there is a one-to-many relationship between Service Request and the following objects: Action, Service Request Attachment, Product Defect, Order Entry - Orders, Account External Product, and Solution SR/PD.

The following table describes the relationship between business components in the Service Request business object and folder names in the tree applet.

Business Component	Folder Name in Tree Applet
Account External Product	Service Profile
Action	Activities
Order Entry - Orders	Service Orders
Product Defect	Change Request
Service Request	Service Requests
Service Request Attachment	Attachments
Solution SR/PD	Solutions

You can configure the tree applet and explorer view for service requests to include more business components. For example, you can add the Contacts, Customer Surveys, and Service Agreements folders as child folders of Service Requests. You can add a Line Items folder as a child of RMAs and Service Orders. You can only add business

components from the business object in an explorer view that references the business object. In this example, that business object is Service Request. You can only add a business component as the immediate child folder of the business component that is the parent of this business component in the business object. For example, you can add Order Entry Line Items as a child of RMAs and Service Orders. You cannot add Order Entry Line Items as a child of Activities.

Objects of a Tree Applet

A *tree* is a child of an applet. It includes the child tree node. Each tree node defines one folder symbol. The tree object includes the following:

- Only provides a named reference point. A tree is similar to the List object type that Siebel CRM uses in a list applet because the tree works only as a reference for child objects.
- Always includes the text Tree in the Name property.

The following table describes properties of an applet that implement a tree applet.

Property	Description
Class	Must be set to CSSFrameTree to support a tree applet.
Business Component	Must reference the same business component as the highest level tree node.

Siebel CRM does not support a search specification on a tree applet. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.

A tree applet references an explorer view as a view web template item. A list applet does not reference an explorer view. If the user chooses a folder, then Siebel CRM determines the list applet dynamically. The folder in a tree applet represents a tree node. The Business Object property of the view determines the business component data that Siebel CRM displays.

Using the Tree Applet Wizard to Create a Tree Applet

It is recommended that you use the Tree Applet Wizard to create a new tree applet.

To use the Tree Applet Wizard to create a tree applet

1. In Siebel Tools, click the File menu, and then click New Object.
2. In the New Object Wizards dialog box, click the Applets tab, and then click the Tree Applet icon.
3. In the General dialog box, define the following properties, and then click Next:
    - Project
    - Business Component
    - Name
    - Display Name
4. In the Web Layout - General dialog box, choose the web template to use for the tree applet, and then click Next.

    The following are some templates that you can use for a tree applet:

- Applet Tree
- Applet Tree 2
- Applet Tree Marketing

5. In the Finish dialog box, review the information, and then click Finish.

   The Tree Applet Wizard creates the tree object and sets the required properties according to the information you entered.
6. Add a tree node for each applet that Siebel CRM must display in the Explorer section of the view, including the top (highest) level node.

   The Tree Applet Wizard does not create child objects for the tree node. You must add a tree node for each applet that Siebel CRM must display in the Explorer section of the view, including the top level node, such as Service Requests. For more information, see *Configuring a Tree Node*.

Configuring a Tree Node

One tree node defines one folder icon. This includes the top (highest) level node, such as Service Requests. Each tree node is a child of the tree. No hierarchy of child and grandchild tree nodes exist under the tree. The hierarchy of these object definitions does not reflect the hierarchy in the tree applet. Instead, the Position property of the tree node defines the hierarchical position of each tree node in the tree applet.

To configure a tree node

1. In the Object Explorer, click Applet.
2. In the Applets list, locate the applet you must modify.
3. In the Object Explorer, expand the Applet tree, and then click Tree.
4. In the Trees list, locate the tree you must modify.
5. In the Object Explorer, expand the Tree tree, and then click Tree Node.
6. In the Tree Nodes list, add a new tree node using values from the following table.

Property	Description
Display Name	Define the name of the tree node.Siebel CRM displays this name in the tree applet after the folder icon.
Applet	Specify the applet that Siebel CRM displays in the second portion of the view if the user opens the corresponding folder. Typically, you specify a list applet. Make sure the applet references a business component that is in the appropriate hierarchical position in the business object.
Position	Do the following: • Define the hierarchical position of the tree node relative to other tree nodes. • Define the sequence of the tree node for the level where the tree node resides. For more information, see *Defining the Position Property of a Tree Node*.
Business Component	Specify the same business component that is defined for the applet that Siebel CRM displays in the second portion of the view.

Property	Description
	Make sure each tree node in the hierarchy references a unique business component. You cannot use one business component for multiple tree nodes because Siebel CRM will not properly refresh the business component.
Label Field	Specify the name of the field that provides the names in the list that Siebel CRM displays if the user expands the node. For example: • The Order Number field provides values for the RMAs and Service Orders node. • The Description field provides values for the Activities node.
Selected Bitmap Index	Specify the number 5. This number identifies the folder symbol.

Defining the Position Property of a Tree Node

The value in the Position property of a tree node includes an integer or a set of integers that are separated by periods, such as 1.1.2. Use the following format:

- Define the top (highest) level node with a position of 1. For example, *x*.1.2, where *x* specifies the top level node.
- Define immediate child nodes of the top level node with a value of 1.*x*, where *x* specifies the order of the node relative to other nodes that reside on the same level.

For example, to display the Activities folder after the Attachments folder rather than before the Attachments folder:

- Set the Position value for the Activities folder to 1.2.
- Set the Position value for the Attachments folder to 1.1.

To attach a child node at the third level, you can define the Position property for the new node so that the first two integers match the position of the parent node. For example, assume you define the RMAs and Service Orders node at position 1.4. To attach a node to the RMAs and Service Orders node, you define the new node with a position of 1.4.1. The farthest digit in a position typically specifies the order relative to other nodes that exist on the same level.

Using the Applet Web Template Editor to Add a Tree Control

You can use the Applet Web Template Editor to add a tree control to a tree applet.

To use the Applet Web Template Editor to add a tree control

1. In the Object Explorer, click Applet.
2. In the Applets list, locate the applet you must modify.
3. In the Object Explorer, expand the Applet tree, and then click Applet Web Template.
4. In the Applet Web Templates list, right-click the template you must modify, and then click Edit Web Layout.
5. Move a TreeControl control from the palette to the applet layout.

   Siebel Tools creates the required controls and the object definition for the tree.
6. To modify your configuration, right-click the tree control, and then click a menu item from the pop-up menu using values from the following table.

Menu Item	Description
Select Tree	Allows you to copy and paste the tree control to another applet.
Create New Tree Node	Adds a new tree node to the tree. Siebel Tools creates the tree node at the highest level. You can then use the Move Selected Tree Node menu item to move the new node.
Move Selected Tree Node	Allows you to modify the position of the tree node in the tree. Press and hold down the SHIFT key, and then use the following keys on your keyboard: • Use the (up and down) arrow keys to move the tree node up or down a level. • Use the (left/back and right/forward) arrow keys to modify the position of the node in the current level.

Siebel Tools updates the Position property of each node according to each operation you perform in the Applet Web Template Editor.

If you press the DELETE key when Siebel Tools displays the tree in the Applet Web Template Editor window, then Siebel Tools deletes the currently chosen tree node. You can use the Undo and Redo menu items in the Edit menu of the Applet Web Editor to modify your modifications.

Configuring a Recursive Tree Applet

A *recursive tree applet* is a type of tree applet where all levels in the hierarchy are of the same object type. For example, the Account Explorer Applet includes a tree applet where the only node is for the Account business component. Siebel CRM displays subaccounts beneath accounts. A recursive tree can contain almost any number of levels of subrecords. Predefined recursive trees exist in Siebel CRM for the following objects:

- Accounts
- Activities
- Campaigns
- Opportunities
- Positions
- Various other business components where records contain subrecords

A recursive tree applet is defined with a tree object where only one tree node is attached. The business component in a recursive tree must reference the record of the same type at the next level up in the hierarchy. In the accounts tree example, the Account business component includes a Parent Account Id field that references the parent account. A link object must exist that references this field in the Destination Field property of the link. In the accounts example, this link is Account/Account.

To configure a recursive tree applet

- Set properties for the tree node using values from the following table.

Property	Description
Recursive	Set to TRUE to indicate that this is a recursive tree.

Property	Description
Recursive Link	Specify the link that references the one-to-many relationship that exists between the parent business component and the child business component. For example, Account/Account. The Account business component is the parent business component and the child business component defines the recursion.
Root Search Spec	Create a search specification that configures Siebel CRM how to get the list of top level records. The top level records typically contain nothing in the parent Id field, so you can use the following format: `[Parent xxx Id] is NULL` where: • *xxx* completes the name of the field. For example: `[Parent Account Id] is NULL` For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.

Configuring a Hierarchical List Applet

This topic describes how to configure a hierarchical list applet. It includes the following information:

- *Viewing an Example of a Hierarchical List Applet*
- *Configuring Indentation and Order of a Hierarchical List Applet*
- *Limiting the Number of Records That Siebel CRM Returns in a Hierarchical List Applet*
- *Configuring a Hierarchical List Applet to Use External Data*

A *hierarchical list applet* is a type of applet that displays records that include a hierarchical relationship. It is similar to a list applet, but you can display a hierarchical list applet in a way that is similar in appearance to a tree control. For example, the Categories list that the user accesses to create and manage a catalog category in Siebel Web Sales.

The Hierarchy Parent Field property of the business component creates the hierarchy.

The HTML Hierarchy Bitmap object that is defined in the HTML Hierarchy Bitmap property of the list defines the icons that Siebel CRM uses to display the list applet. You must define the following bitmaps for the HTML Hierarchy Bitmap:

- Expand Bitmap
- Collapse Bitmap
- Space

Siebel CRM can display a hierarchical list applet in Base or Edit List mode.

It is recommended that the number of columns that Siebel CRM displays in a hierarchical list applet be small because the width of the column expands as the user navigates down the hierarchy. It is recommended that it only display fields that contain small values in a column that includes an expand control and a collapse control.

Running a Query on a Hierarchical List Applet

If you run a query on a hierarchical list applet, then Siebel CRM only returns the root layer of records. It returns no child records. This situation does not cause a problem in the Siebel client because the user can expand the root level record to view child records. If you run a query in a script, then it only returns the top level records.

Viewing an Example of a Hierarchical List Applet

You can view an example of a hierarchical list applet in the Siebel client.

To view an example of a hierarchical list applet

1. In the Siebel client, click the Quotes screen tab, and then click the List link.
2. Click a link in the Quote # column.
3. Click the Line Items tab.

   Siebel CRM displays the Quote Item List Applet. This applet is an example of a hierarchical list applet.
4. Expand the hierarchy in the Line # column.

   Siebel CRM displays an expanded hierarchy that includes indented document icons and sequence numbers.

Configuring Indentation and Order of a Hierarchical List Applet

If you call an Indent or Outdent applet method menu item on a record, then Siebel CRM demotes or promotes the child records. It does not modify the relationship to the child records of the called record.

Modifications that the MoveUp and MoveDown applet method menu items make are temporary. Siebel CRM does not save these modifications to the Siebel database.

You can define other bitmaps to create an applet that resembles the tree applet. The tree applet is not defined for a hierarchical list applet in the Siebel client. Siebel CRM uses the Arrow Down and Arrow Up bitmap only in a tree control. For more information, see *Configuring a Tree Applet*.

To configure indentation and order of a hierarchical list applet

1. In the Object Explorer, click Applet.
2. In the Applets list, locate the applet you must modify.
3. In the Object Explorer, expand the Applet tree, and then click Applet Method Menu Item.
4. In the Applet Method Menu Items list, create a new record and set properties using values described in the following table.

The following table describes the applet method menu items you can call from a control. These methods allow you to edit the indentation and order that Siebel CRM uses to display objects in the hierarchical list applet.

Item	Description
Indent	Moves the current record to a position that is indented from the peer record.

Item	Description
Outdent	Moves the current record to a position that is at the same level as a peer record of the parent.
MoveUp	Moves the current record to a position that is prior to the position of the peer record.
MoveDown	Moves the current record to a position that is after the position of the peer record.

Limiting the Number of Records That Siebel CRM Returns in a Hierarchical List Applet

If you define a hierarchical list applet, then the business component returns all the records that make up the hierarchy. It does this to create the hierarchy of records. Siebel CRM cannot typically return more than ten thousand records. If a query returns more than ten thousand records, then it does not display the applet and the user might encounter an error that is similar to the following:

```
There were more rows than could be returned. Please refine your query to bring back fewer rows.
```

To limit the number of records that Siebel CRM returns in a hierarchical list applet

- Use one of the following configurations to make sure that the applet does not return more than ten thousand rows:
  - Create a search specification on the business component or on the applet. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.
  - Use a predefined query for the view. For more information, see *Guidelines for Modifying a Predefined Query*.

Configuring a Hierarchical List Applet to Use External Data

This topic describes how to configure a hierarchical list applet to use external data. To develop this example, do the following tasks:

1. *Creating the Virtual Business Component*
2. *Creating the Business Service for the Hierarchical List Applet*
3. *Implementing the Customization*

You can configure a hierarchical list applet to get external data from a virtual business component. A hierarchical list applet does not require special configuration on a business component other than a properly set Hierarchy Parent Field property. More configuration is required for a virtual business component.

Creating the Virtual Business Component

You start by creating the virtual business component.

To create the virtual business component

1. In the Object Explorer, click Business Component.
2. In the Business Components list, create a new business component using values from the following table.

Property	Value
Hierarchy Parent Field	Id

3. In Business Components list, right-click the record you created in Step 2, and then click Edit Server Scripts.
4. In the Scripting Language dialog box, choose eScript, and then click OK.
5. In the BusComp Script window, expand the BusComp tree, and then click BusComp_PreInvokeMethod.
6. In the script editing window, remove the existing script, and then enter the following script:

```
function BusComp_PreInvokeMethod (MethodName)
{
TheApplication().Trace(this.Name() + ".PreInvoke." + MethodName + "()");
 return (ContinueOperation);
}
```

7. In the BusComp tree, click BusComp_InvokeMethod, remove the existing script, and then enter the following script:

```
function BusComp_InvokeMethod (MethodName)
{
TheApplication().Trace(this.Name() + ".Invoke." + MethodName + "()");
}
```

8. In the Object Explorer, expand the Business Components tree, and then click Field.
9. In the Fields list, add fields to your virtual business component using values from the following table.

Name	Type
Has Children	DTYPE_BOOL
Is Expanded	DTYPE_BOOL
Last Child Info	DTYPE_TEXT
Outline Number	DTYPE_TEXT

Creating the Business Service for the Hierarchical List Applet

In this topic, you create the business service for the hierarchical list applet.

To create the business service for the hierarchical list applet

1. Display the Business Service Server Script object type.

The business service server script is a child of the business service. For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.

2. In the Object Explorer, click Business Service.
3. In the Business Services list, add a new business service using values from the following table.

Property	Value
Name	Hierarchical List Service
Server Enabled	Check mark

4. In the Object Explorer, expand the Business Service tree, and then click Business Service Server Script.
5. In the Business Service Server Scripts list, add a new record using values from the following table.

Property	Value
Name	Init
Program Language	JS
Sequence	1
Script	Enter the following script: `function Init (Outputs)` `{` ` with( Outputs )` ` {` ` SetProperty ("Parent Row Id", "");` ` SetProperty ("Amount", "");` ` SetProperty ("Description", "");` `// SetProperty ("Has Children", "N");` `// SetProperty ("Is Expanded", "N");` `// SetProperty ("Outline Number", "0");` `// SetProperty ("Last Child Info", "");` ` }` ` return( CancelOperation );` `}`

You can copy the text from this book, and then paste it into the Script property. To view the correctly formatted script, right-click Hierarchical List Service in the Business Services list, choose Edit Service Scripts, expand the general tree, and then click Init.

6. In the Business Service Server Scripts list, add a new record using values from the following table.

Property	Value
Name	Query
Program Language	JS
Sequence	2
Script	For more information, see *Extensive Code Examples That This Book Uses*.

7. In the Business Service Server Scripts list, add a new record using values from the following table.

Property	Value
Name	Service_PreInvokeMethod
Program Language	JS
Sequence	3
Script	Enter the following script: `function Service_PreInvokeMethod (MethodName, Inputs, Outputs)` `{` `TheApplication().Trace( this.Name() +".PreInvokeMethod( " + MethodName + " )");` `switch( MethodName )` `{` `case "Init":` `return( Init (Outputs) );` `case "Query":` `return( Query (Inputs, Outputs) );` `}` `return (ContinueOperation);` `}`

8. Add code to the Query method of the business service so that the method provides output that is meaningful for these fields:
   - **Has Children.** Y or N, depending on if the record references children or does not reference children.
   - **Is Expanded.** Y or N, depending on if Siebel CRM displays the record as expanded or not expanded in the applet.
   - **Outline Number.** A string that describes the position of the record in the hierarchy. For example. 1.2 or 2.1.1.

- **Last Child Info.** A string that represents a binary sequence that indicates if the record and the parent of the record is the last record in the list of children. For more information, see the following section.

The code can be similar to the code that you added in the previous step.

9. To maintain the values appropriately, add code to the Update method.

   The Update method is specific to your custom implementation and requires a mechanism to update the records of the virtual business component. The exception is if all records are read-only, then no Update method is required.

About Last Child Info

The output for Last Child Info in this example is a string of three bits that Siebel CRM displays for each level in the hierarchy if more records exist. Consider the test values in the Query method for this example. The following situations apply for a tree that includes three levels:

- If an item exists in the tree that is at position 1.3.2, and if item 1.3.3 does not exist, then the third bit is 1, which you can think of as xx1. Otherwise the third bit is 0, which you can think of as xx0.
- If the parent record at position 1.3 is the last child, then the second bit is 1, which you can think of as x1x. If item 1.4 does not exist in the tree, then Siebel CRM considers the record as the last child.
- If the grandparent record at position 1 is the last child, and if item 2 does not exist in the tree, then the first bit is 1, which you can think of as 1xx.

Implementing the Customization

In this topic, you implement the customization.

To implement the customization

1. Apply any modifications that you made in the base table to the Siebel runtime repository.
2. If you defined a new screen, then add the screen to the application screen object.
3. Test and then deliver your Workspace.
4. In the Siebel client, add the view to the list of views, and then add an appropriate responsibility so that the user can access this view.
5. Test your modifications.

Configuring a File Attachment Applet

This topic describes how to configure a file attachment applet. It includes the following information:

- *Configuring an Attachment Business Component*
- *Configuring an Attachment Table*

A *file attachment applet* is a type of applet that provides access to an external document, such as a spreadsheet, word processing document, or slide presentation in Siebel CRM. It provides the following capabilities:

- Allows the user to click the name of a file from a list to open a document.
- Allows the user to add a document file to a list, edit it, or remove it.
- Provides synchronization and shared access support for attached documents.

You can use any file type that Windows supports.

To view an example of a file attachment applet in the Siebel client, navigate to the Account screen, drill down on an account, and then click the Attachments tab. The client displays the Account Attachment view:

- The form applet is the predefined Account Form Applet.
- The list applet is the Account Attachment Applet. This attachment applet displays attachments for the account.

A parent-child relationship exists between the account and the list of account attachments. A row in the attachments list represents each document. Siebel CRM displays the following information in this applet:

- File name for the document. Siebel CRM underlines and uses colored font to display each file name. This style indicates that the user can click the name to open the file in a Windows application.
- Local and server status.
- File size.
- File name extension that identifies the file type.
- Date of last update.

To add a document to the attachment list, the user clicks New File, and then clicks the select button in the Attachment Name field. Siebel CRM searches for files that it must attach in the directory that it last used to attach a file. If the user chooses a different folder while attaching a file, then it searches for the file in the different folder the next time the user attaches a file.

A file attachment applet uses specialized objects and methods in the Siebel File System. For more information, see *Caution About Using Specialized Classes*.

To configure a file attachment applet

1. If necessary, configure an attachment business component.

   For more information, see *Configuring an Attachment Business Component*.
2. Create a file attachment applet using values from the following table.

Property	Description
Business Component	Specify the required business component. For more information, see *Configuring an Attachment Business Component*.
Class	Set to one of the following values: o CSSFrameListFile for an attachment list applet o CSSFrameFile for an attachment form applet

3. Add a new child list column or control to the applet for each row in the following table.

Display Name	Field	Type
Name	*prefix* FileName	TextBox
Local	Dock Status	CheckBox

Display Name	Field	Type
Request	*prefix* FileDockReqFlg	CheckBox
Size	*prefix* FileSize	TextBox
Type	*prefix* FileExt	TextBox
Modified	*prefix* FileDate	TextBox
Auto Update	*prefix* FileAutoUpdFlg	CheckBox

For the prefix, enter the required prefix for the business component. For more information, see *Prefix for the Field Name*.

These list columns or controls reference fields in the attachment business component. For more information, see *Configuring an Attachment Business Component*.

4. Make sure the value in the Detail Applet property of each list column or text box control you added in the previous step is File Popup Applet.

   This value references the dialog box that Siebel CRM displays if the user clicks the ellipsis in the list column or text box.

Prefix for the Field Name

Siebel CRM displays a consistent prefix in the field name for each field in the attachment business component. These fields reference the base table for the business component. Fields that reference a joined table use a different prefix. For example, the prefix for account attachments is Accnt. The field names are AccntFileName, AccntFileDockReqFlg, and so on.

Configuring an Attachment Business Component

The Business Component property of the attachment list applet identifies the business component that the Siebel File System uses to store the attachment list data. For example, this business component for the Account Attachment Applet is named Account Attachment.

To configure an attachment business component

1. In Siebel Tools, in the Object Explorer, click Business Component.
2. In the Business Components list, locate the attachment business component you must modify.
3. Make sure the value in the Class property is CSSBCFile or a subclass of CSSBCFile, such as CSSBCSalesTool or CSSBCEventFile.
4. Make sure the Table property references an attachment table.

For example, the attachment table is S_ACCNT_ATT in the Account Attachment Applet. For more information, see *Configuring an Attachment Table*.

5. In Siebel Tools, in the Object Explorer, expand the Business Component tree, and then click Business Component User Prop.
6. In the Business Component User Props list, create a new business component user property using values from the following table.

Name	Value
DefaultPrefix	Specify the prefix. For more information, see *Prefix for the Field Name*.

7. In the Business Component User Props list, create a new business component user property using values from the following table.

Name	Value
FileMustExist	Use one of the following values: o **TRUE.** If the file does not exist, then the user cannot enter the name of the file. TRUE is the typical value. o **FALSE.** If the file does not exist, then the user can enter the name of the file.

8. Make sure the Predefault property of the FileDockReqFlg business component field is N.

   The FileDockReqFlg references the required FILE_DOCK_REQ_FLG column in the attachment table.
9. Add the required business component fields.

   For more information, see *Fields in an Attachment Business Component*.

Fields in an Attachment Business Component

The following table describes each field name that the file engine supplies. These names must use a special format and reference a column name in the attachment table. The name includes the prefix followed by a required suffix. The DefaultPrefix user property defines the prefix.

Name	Column	Type	Text Length
prefix FileAutoUpdFlg	FILE_AUTO_UPD_FLG	DTYPE_BOOL	1
prefix FileDate	FILE_DATE	DTYPE_DATETIME	7
prefix FileDeferFlg	FILE_DEFER_FLG	DTYPE_TEXT	1
prefix FileDockReqFlg	FILE_DOCK_REQ_FLG	DTYPE_TEXT	1
prefix FileDockStatFlg	FILE_DOCK_STAT_FLG	DTYPE_TEXT	1

Name	Column	Type	Text Length
prefix FileExt	FILE_EXT	DTYPE_TEXT	10
prefix FileName	FILE_NAME	DTYPE_TEXT	200
prefix FileRev	FILE_REV_NUM	DTYPE_ID	15
prefix FileSize	FILE_SIZE	DTYPE_NUMBER	22
prefix FileSrcPath	FILE_SRC_PATH	DTYPE_TEXT	220
prefix FileSrcType	FILE_SRC_TYPE	DTYPE_TEXT	30

The following table describes a field that the file engine does not supply. This field is typically present but is not required.

Name	Column	Type	Calculation
Dock Status	(calculated)	DTYPE_BOOL	IIf ([AccntFileDockStatFlg] = "N" OR [AccntFileDockStatFlg] IS NULL, "N","Y")

You can include more fields. For a special use of an attachment, such as an image control, the file engine fields can be present in addition to the fields from a predefined business component. Siebel CRM typically gets the fields from the predefined business component through a join. For example, a Product or Literature business component can contain file engine fields to support how Siebel CRM displays a product picture or a brochure picture from a bitmap image.

You can include multiple sets of file engine fields from different tables in the same business component. For example, a literature attachment can include subattachments where Siebel CRM gets the subattachments from an intersection table or an extension table. Make sure the prefix for the field name is different for each table.

Configuring an Attachment Table

An *attachment table* is a type of table that provides the underlying data storage for the attachment business component. The attachment business component can support uses in addition to file engine functionality. The attachment table stores only file engine data.

The user does not directly update the attachment table. Siebel CRM provides the user an empty attachment table. The user then uses the relocate functionality, using the mouse, or the browser dialog box in the corresponding file attachment applet to bring data into the empty table one file at a time.

The following table describes required file columns in an attachment table.

Name	Default	User Name	Type	Physical Type	Length
FILE_AUTO_UPD_FLG	N	File Auto Upd Flg	Data (Public)	Character	1
FILE_DATE	None	File Date	Data (Public)	Date Time	7
FILE_DEFER_FLG	R	File Defer Flg	Data (Public)	Character	1
FILE_DOCK_REQ_FLG	N	File Dock Req Flg	Data (Public)	Character	1
FILE_DOCK_STAT_FLG	N	File Dock Stat Flg	Data (Public)	Character	1
FILE_EXT	None	File Ext	Data (Public)	Varchar	10
FILE_NAME	None	File Name	Data (Public)	Varchar	200
FILE_REV_NUM	0	File Rev Num	Data (Public)	Varchar	15
FILE_SIZE	None	File Size	Data (Public)	Number	22
FILE_SRC_PATH	None	File Src Path	Data (Public)	Varchar	255
FILE_SRC_TYPE	None	File Src Type	Data (Public)	Varchar	30

You must use the value described in the previous table for the Name property. You are not required to use the value described in the previous table for the User Name property.

Various system columns that are not related to the file engine are present, such as CREATED, LAST_UPD_BY, and ROW_ID.

If the table includes a file engine column, then you must make sure the File property in the corresponding table is TRUE.

Configuring an Organization Analysis Applet

This topic describes an example of how to configure an Organization Analysis Applet. The contacts that Siebel CRM associates with each opportunity determines how Siebel Sales creates organization charts. Siebel CRM reflects these modifications in the organization chart when it updates contact information. To view an organization chart, you can open the Siebel client, navigate to the Accounts screen, and then the Organization Analysis view. Siebel CRM displays the organization chart in the Account Organization Analysis Applet.

Siebel CRM uses the CSSFrameContactOrgChart class for predefined applets in an organization chart. It displays the following fields:

- Full Name
- Job Title

- Work Phone#

To require one or more fields to display in the box of an organization chart, you can add a new list column for each field. For more information, see *Siebel Applications Administration Guide* . See also *Using Siebel Tools* .

To configure the organization analysis applet

1. In Siebel Tools, in the Object Explorer, click Applet.
2. In the Applets list, locate the Account Organization Analysis Applet.
3. In the Object Explorer, expand the Applets tree, expand the List tree, and then click List Column.
4. In the List Columns list, add a new record using values from the following table.

Property	Description
Name	Enter any text that describes the content of the field.
Field	Choose the field that contains the information for the contact that you must display in the organization chart.
Display Name - String Reference	Choose an appropriate value.
HTML Sequence	Enter a value to define the sequence that Siebel CRM uses to display the field. A lower sequence causes the field to display higher in the box in the organization chart. If you set the sequence, then make sure the ClientConfigurationMode parameter is not set to All. For more information, see *Setting Up the Configuration File for Siebel Tools*.

5. Repeat the previous step for each additional field that Siebel CRM must display in the applet.
6. Test and then deliver your Workspace.

18 Configuring Lists and Pick Applets

Configuring Lists and Pick Applets

This chapter describes how to configure lists and pick applets. It includes the following topics:

- *About Lists and Pick Applets*
- *List and Pick Applet Configuration*
- *Creating a List of Values*
- *Associating an Organization with a List of Values*

About Lists and Pick Applets

This topic describes lists and pick applets. It includes the following information:

- *About Static Lists*
- *About Pick Applets*
- *About Dynamic Lists*
- *About Hierarchical Lists*

A *list* is a type of user interface element that allows the user to choose values from a list to update a field instead of entering values into a field. You can define the following types of lists:

- **Static list.** Gets data from the Siebel list of values table that an administrator maintains. The data in the list of values table is static. For more information, see *Creating a List of Values*.
- **Dynamic list.** Gets data from tables that the user maintains, such as S_CONTACT or S_ORG_EXT. The data in these tables are dynamic.

About Static Lists

A *static list* is a type of list that displays a list of predefined values. If the user clicks the arrow that Siebel CRM displays after a field in an applet, then it displays a list that contains a single column. The user chooses a value from the list, and then clicks Save to enter the value for the field. You can define the values in the list to store them in the list of values table.

A list can be bounded or unbounded:

- A *bounded list* is a type of list that allows the user to choose a value only from the list.
- An *unbounded list* is a type of list that allows the user to choose a value from the list or to type a value directly into the field.

You cannot delete the lookup value. You can set the field that the user chooses back to empty, unless it is required. Lead Quality is an example of a chosen field.

For more information, see *Using the Pick List Wizard to Create a Static List* and *Creating a List of Values*.

Viewing an Example of a Static List

You can view an example of a static list.

To view an example of a static list

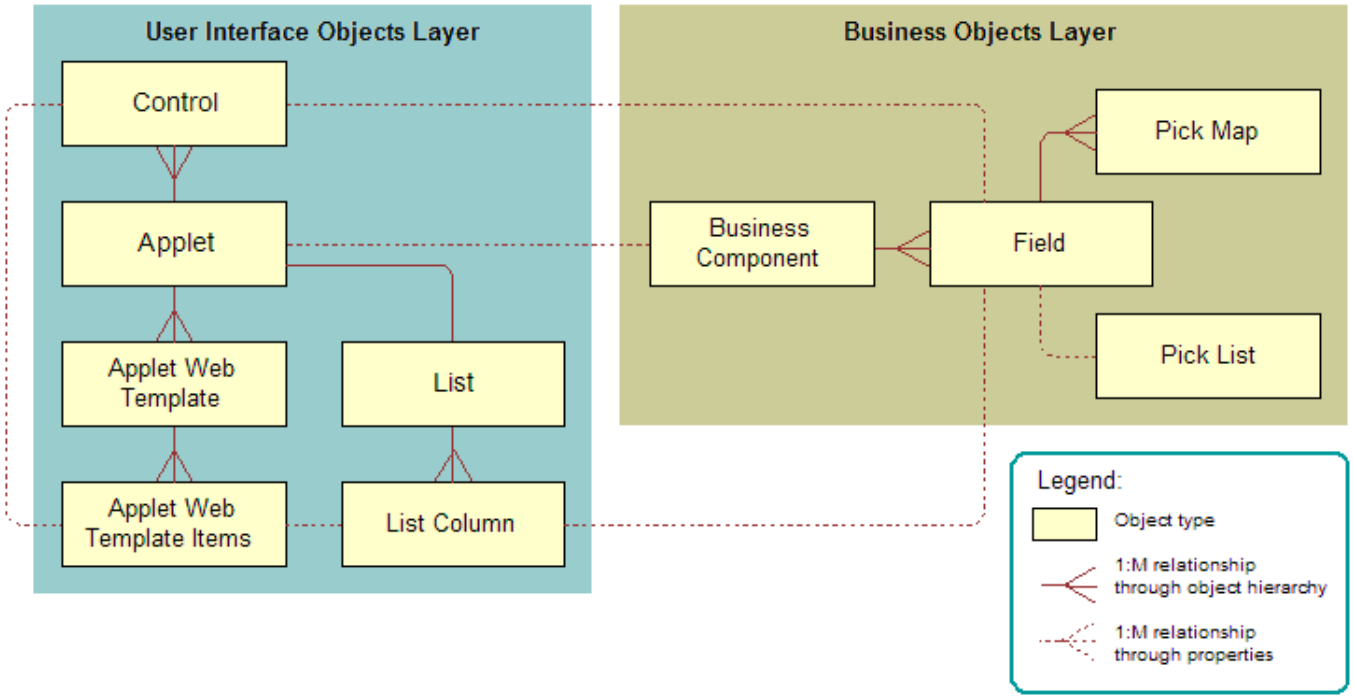
1. In the Siebel client, choose the Contacts screen.
2. Choose the Contacts List link.
3. Choose an existing record in the Contacts list.
4. Choose the Mr/Ms field.
5. Click the down arrow.

   The list that Siebel CRM displays is an example of a static list. It includes static values, such as Miss, Mr., Ms., Mrs. and Dr.

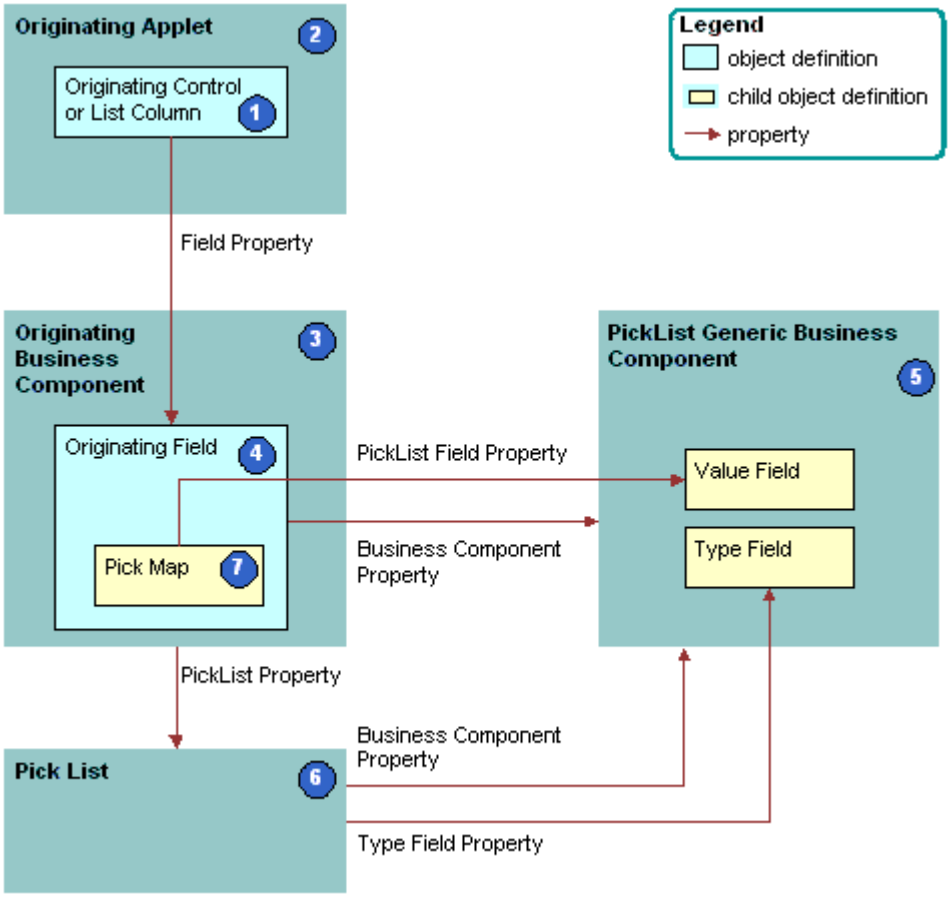
How Siebel CRM Creates a Static List

The following figure shows the relationships and objects that Siebel CRM uses to create a static list, which are as follows:

- **User Interface Objects Layer.** Contains the following objects: Control, Applet, Applet Web Template, Applet Web Template Items, List and List Columns. There is a 1:M relationship between Applet and Control, Applet and Applet Web Template, Applet Web Template and Applet Web Template Items (through properties), List and List Column. There is a 1:1 relationship between Applet and List.
- **Business Objects Layer.** Contains the following objects: Business Component, Field, Pick Map, Pick List. There is a 1:M relationship between Business Component and Field, Field and Pick Map. There is a 1:1 relationship between Field and Pick List (through properties).



The following figure shows an example of how Siebel CRM creates a static list. This example implements the Quality list that is described in the preceding figure.



As shown in this figure, Siebel CRM uses the following objects to create a static list:

1. **Originating control or list column.** The control or list column that the user clicks to call the list. In this example, Quality is the originating control.
2. **Originating applet.** The applet that contains the originating control or list column. After the user chooses a value from the list, the originating control displays a revised value. In this example, the Opportunity form applet is the originating control. The Business Component property of the originating applet identifies the originating business component.
3. **Originating business component.** Business component that the originating applet references. This business component supplies the data that Siebel CRM displays in the originating applet. It updates one field in the current record in this business component after the user chooses a value in the list. In this example, the Opportunity business component is the originating business component.
4. **Originating field.** Field in the originating business component that the originating control represents. This field typically contains one pick map child that defines how the field from the PickList Generic business component maps to the originating business component. In this example, Quality is the originating field.
5. **PickList Generic business component.** Business component that provides the lists in a static list. You use the List of Values view in the Administration - Data screen in the Siebel client to administer the PickList Generic business component. For more information, see *About the Picklist Generic Business Component*.
6. **Pick list.** Identifies the business component that the pick applet references and the field that provides data for the pick applet. This business component is always PickList Generic. In this example, the Pick List is named Picklist Quality. The field of the originating control references the list.
7. **Pick map.** Defines a relationship between the Value field in the PickList Generic business component and the originating field. If the user chooses a value from the list, then this relationship provides the information that Siebel CRM requires to update the record in the current originating business component with information from the PickList Generic business component. The pick map is a child of the originating field.

8. **Sequence property.** Defines the sequence that Siebel CRM uses to update fields in the current record of the originating business component. It updates these fields with information from the pick business component. If you do not define sequence numbers on the pick map, then it updates fields in the order that it uses to create these fields.

About the Picklist Generic Business Component

The PickList Generic business component is a specialized business component that Siebel CRM reserves for lists of values for static lists. The following fields in the Picklist Generic business component define and group the lists of values:

- **Type.** Groups together all records that Siebel CRM includes in one list of values. For example, the LEAD_QUALITY type identifies a record as a member of the Lead Quality list of values. Each list of values includes a type. For more information, see *Creating a List of Values*.
- **Value.** The value that Siebel CRM displays in the static list. For example, values for Lead Quality include Excellent, Very Good, High, Fair, and Poor.

For more information, see *Caution About Using Specialized Classes*.

The following table lists example data in the Picklist Generic business component.

Type Field	Value Field
LEAD_QUALITY	Excellent
LEAD_QUALITY	Very Good
LEAD_QUALITY	High
LEAD_QUALITY	Fair
LEAD_QUALITY	Poor
PERSON_TITLE	Mr.
PERSON_TITLE	Ms.
PERSON_TITLE	Dr.
ACCOUNT_TYPE	Commercial
ACCOUNT_TYPE	Competitor
ACCOUNT_TYPE	Customer

Comparison of a Static List to a Dynamic List

A static list differs from a dynamic list in the following ways:

- A static list is a static list of values. It does not get values dynamically from a pick business component. An administrator defines these values in the List of Values view in the Administration - Data screen.
- A static list typically does not call a dialog box with multiple list columns and buttons. Instead, it uses a simple pop-up list that contains one column and no buttons. It is possible to use a pick applet rather than a simple list to display a static list of values, but Siebel CRM does not frequently use this configuration.
- A static list does not provide data for multiple controls in the originating applet. It provides data for a single control in the applet and the corresponding field in the business component that the applet references.

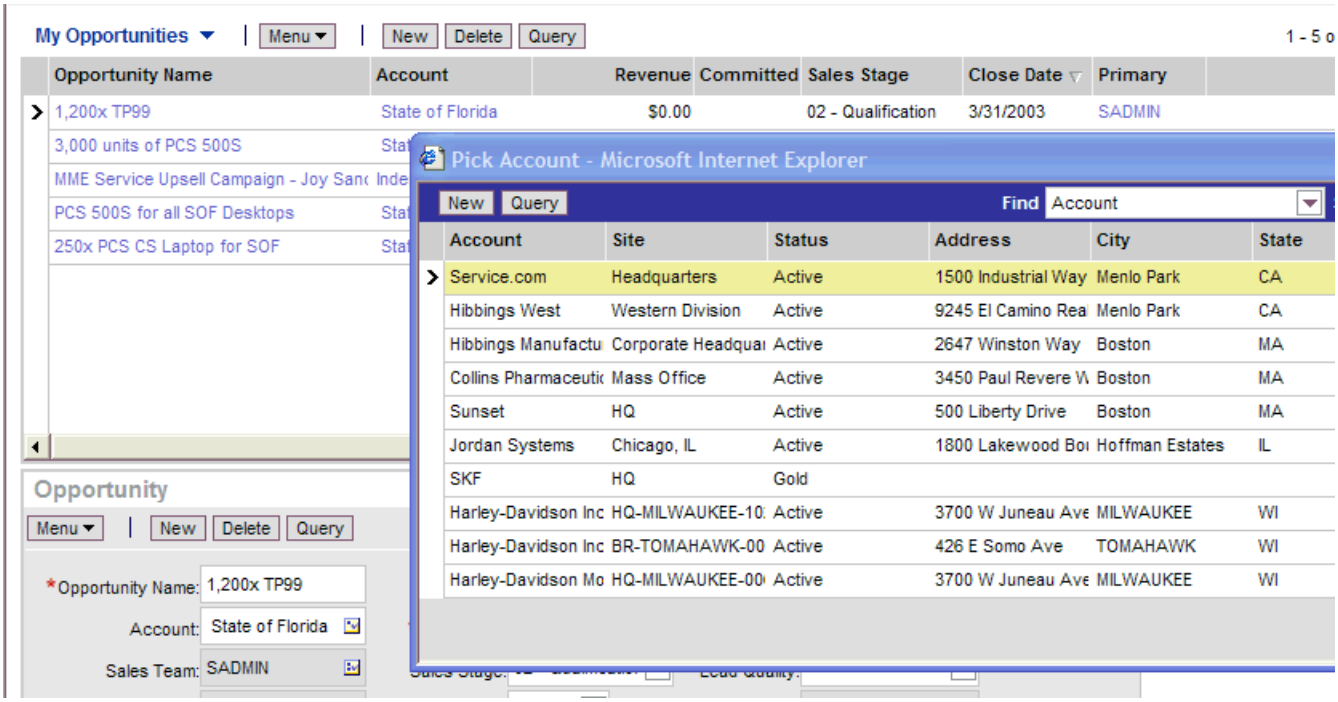
For more information, see *About Dynamic Lists* and *Creating a List of Values*.

About Pick Applets

A *pick applet* is a type of applet that Siebel CRM calls if the user clicks the Select button that Siebel CRM displays next to some fields. A pick applet contains a scrolling table that lists choices in one list column and more information in adjacent columns. Each row corresponds to a business component record in the pick business component. If the user chooses a row in the scrolling table, and then clicks OK, then Siebel CRM hides the pick applet, and then enters data into the column cell and other controls and cells in the originating applet. This data includes information according to the choice the user makes in the pick applet. For more information, see *Using the Pick Applet Wizard to Create a Pick Applet*.

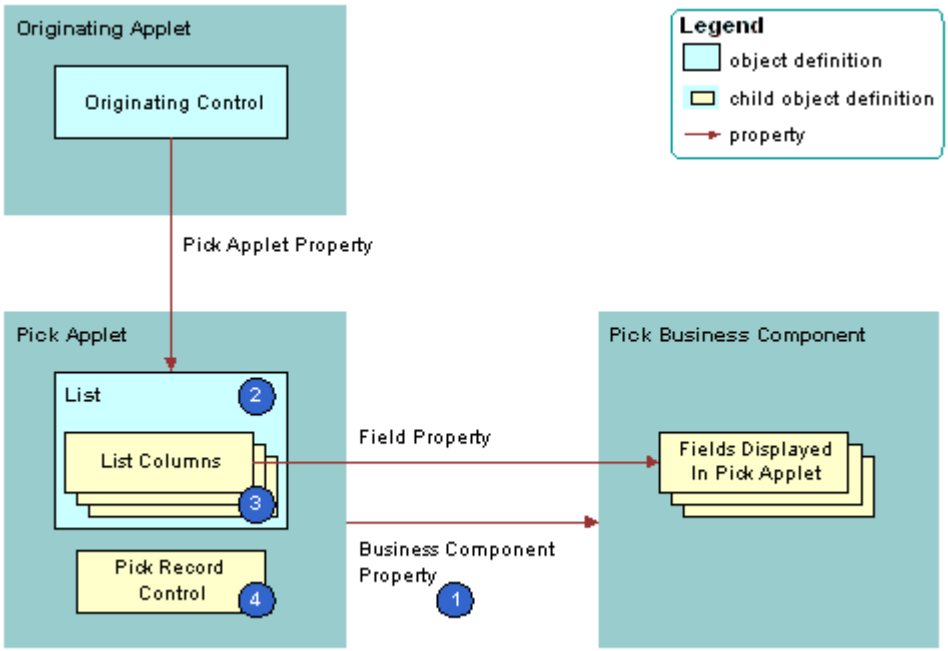
Example of a Pick Applet

Assume the user navigates to the Opportunities screen, and then clicks Select in the Account field in the Opportunity Form applet. Siebel CRM displays the Pick Account dialog box (shown in the following image). This dialog box is a pick applet. The user then chooses an account and clicks OK. It hides the Pick Account dialog box, and then enters data into the Account field in the Opportunity Form applet. This data includes the account that the user chose in the Pick Account dialog box. Siebel CRM might update other fields and controls, such as the Site field. It enters data into the Account and Site fields only if a pick map is defined.



How Siebel CRM Creates a Pick Applet

The following figure shows how Siebel CRM creates a pick applet.



As shown in this figure:

1. A pick applet is a child of an applet. It includes the following properties:
    - **Business component.** Identifies the business component.
    - **Class.** Set to CSSFrameList, which indicates that this is a list applet.
    - **Type.** Set to Pick List, which indicates that this is a pick applet. This property determines the behavior of the dialog box and button controls.
    - **Title.** Set to the name of the pick applet that Siebel CRM displays in the title bar.
2. The pick applet includes the following child objects:
    - **List.** List columns that Siebel CRM attaches to the list.
    - **List columns.** Each list column displays the contents of one field in the business component.
    - **Pick Record control.** Calls the PickRecord method if clicked. The PickRecord method locates the pick map child objects of the originating field. The PickRecord method uses these child objects to identify the fields that Siebel CRM updates in the originating business component. The record that the user chooses from the pick business component determines how it updates these fields.
    - **Web templates.** Defines the layout for each of the defined modes. Example layout includes the position of the list columns and controls.
    - **Web template items.** Maps list columns and controls to placeholders in the web template. Web template items exist for each list column and control that is defined for the applet.

Pick Applet Usage in Query Mode

If a pick map operates in query mode, then the fields that Siebel CRM copies includes the source field that the following items use:

- The list
- Any other list field that is part of the primary key

The source field is the business component field that the Business Component property of the list defines. Siebel CRM uses multiple fields for the query, so it is not possible to only copy back a single field.

For example, assume the user does a query through the list that Siebel CRM displays for the Parent Account Name field of the account list in the Account screen. Siebel CRM uses the name and location for the query because these fields are part of the U1 index of the underlying S_ORG_EXT table. The Location field is a primary key field for the Account business component and Siebel CRM includes it in the query.

You can define a pick applet so that Siebel CRM correctly enters data into the Subarea field according to the choice that the user makes, such as the Area field of the Service Request Detail Applet. You can use this configuration in edit mode but not in query mode. For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data* and *Creating a Hierarchical List*.

About Dynamic Lists

This topic describes the dynamic list. It includes the following information:

- *Example of How Data Flows in a Pick Applet*
- *How Siebel CRM Creates a Dynamic List*
- *Originating Applet of a Dynamic List*

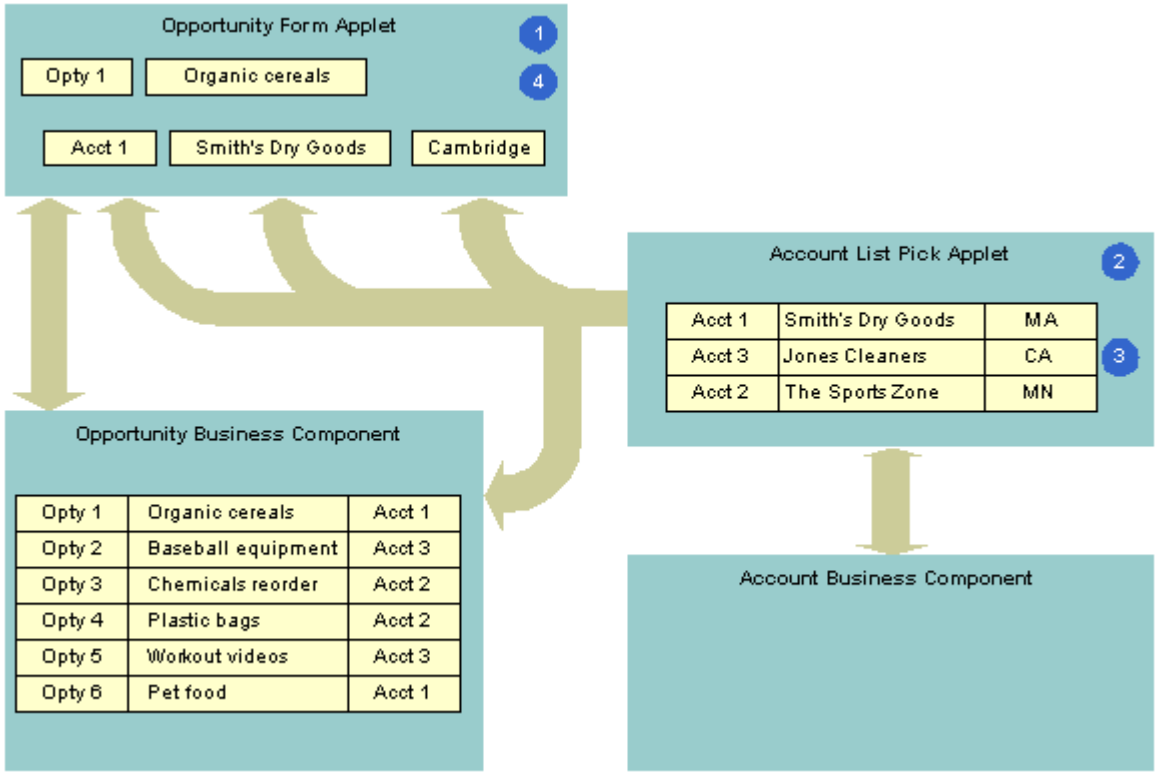
- *Originating Business Component of a Dynamic List*
- *How Siebel CRM Constrains a Dynamic List*

Similar to a static list, a *dynamic list* is a type of list that allows the user to choose a value from a list, and then Siebel CRM uses the value to update a field. Rather than getting the values from the list of values table, a dynamic list gets values from another business component that the user maintains. A field that uses a dynamic list is typically a joined field that displays data from a table other than the base table of the business component. The dynamic list allows the user to update the joined field.

You use a pick applet to display a dynamic list in the Siebel client. The pick applet allows the user to choose a value from a list, and then enter the value into a control or the cell of a list column. For more information, see *Using the Pick List Wizard to Create a Dynamic List*.

Example of How Data Flows in a Pick Applet

The following figure shows how data in the pick applet typically originates from a different business component than the business component that supplies data to the originating applet.



The steps shown in this image are as follows:

1. The user enters information for the Organic Cereals opportunity in the Opportunity Form applet, and then clicks Select.
2. Siebel CRM displays the Account List pick applet and displays rows from the Account business component.
3. The user chooses Account 1, Smith's Dry Goods, and then clicks OK.
4. Siebel CRM enters Account data for Smith's Dry Goods into the Opportunity Form applet.

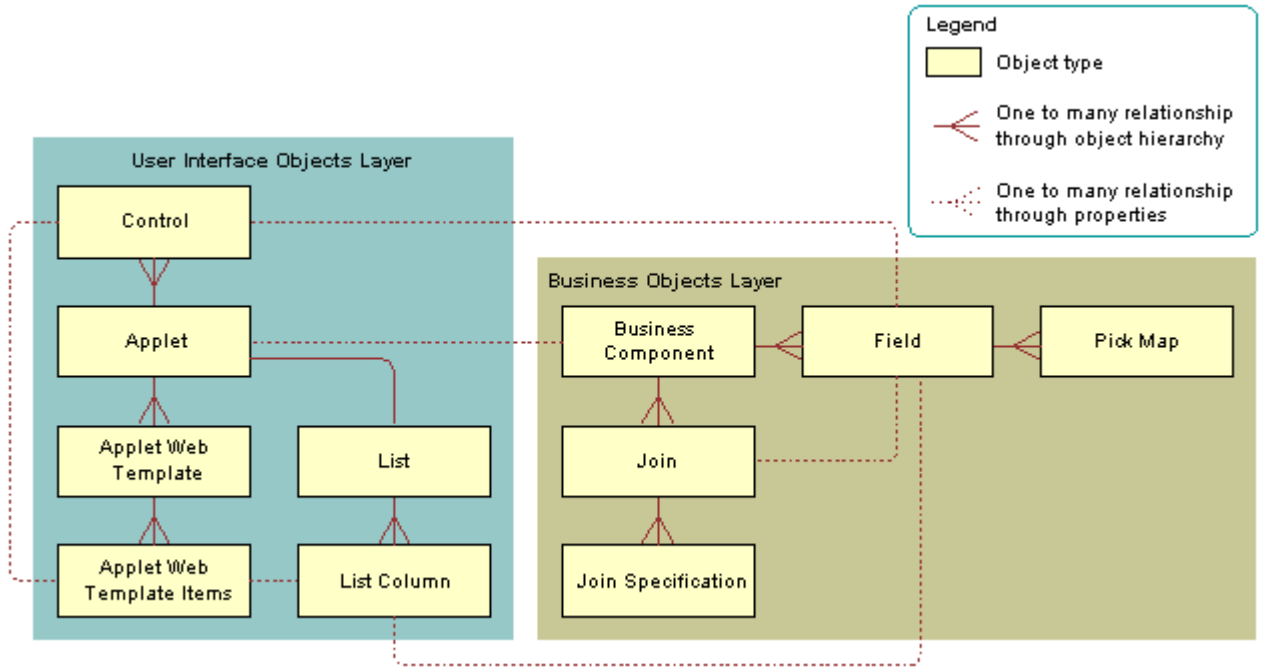
A dynamic list maintains the foreign keys that facilitate a join relationship. In the opportunity and account example, a foreign key in the Opportunity business component identifies the account for each opportunity. If the user chooses an account in the pick applet, then Siebel CRM enters data into this foreign key field. This choice associates the account with this opportunity for future use by the join that uses the foreign key. For example, if the user chooses a record in

the pick applet, then it copies values in some list columns in the chosen record to corresponding list columns in the originating applet. In this example, the user chooses a parent account for an account record.

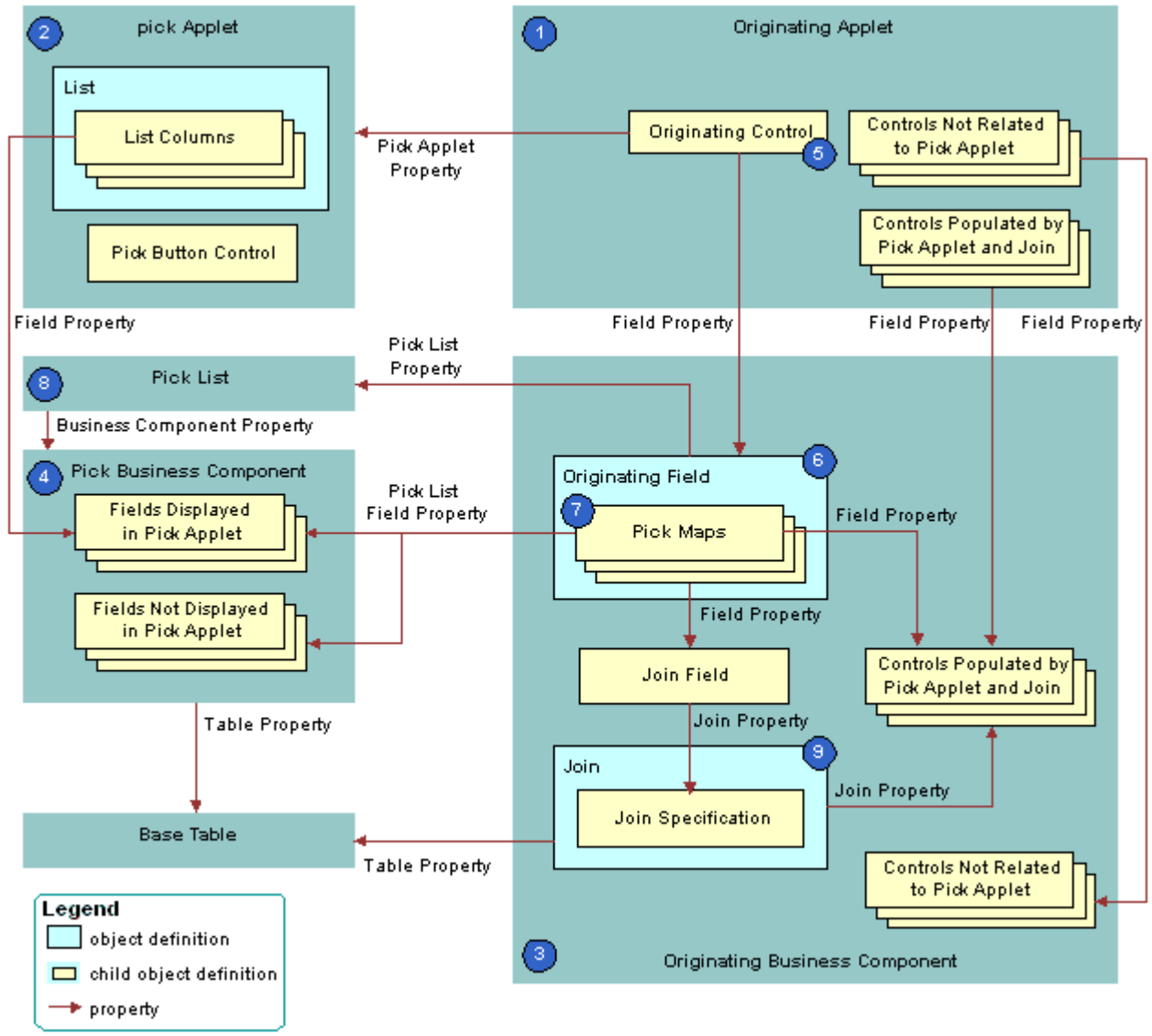
How Siebel CRM Creates a Dynamic List

The following figure shows the relationships between object types that Siebel CRM uses to create a dynamic list, which are as follows:

- **User Interface Objects Layer.** Contains the following objects: Control, Applet, Applet Web Template, Applet Web Template Items, List and List Columns. There is a 1:M relationship between Applet and Control, Applet and Applet Web Template, Applet Web Template and Applet Web Template Items, List and List Column. There is a 1:1 relationship between Applet and List.
- **Business Objects Layer.** Contains the following objects: Business Component, Field, Pick Map, Join, Join Specification. There is a 1:M relationship between Business Component and Field, Field and Pick Map, Business Component and Join, Join and Join Specification. There is a 1:1 relationship between Join and Field.



The following figure shows the objects that Siebel CRM uses to create a dynamic list.



As shown in this figure, Siebel CRM uses the following objects to create a dynamic list:

1. **Originating applet.** Contains the control or list column that calls the pick applet. After Siebel CRM calls the pick applet and chooses a value, it displays revised values in specific controls in the originating applet. In the example, the Opportunity Form Applet is the originating applet. For more information, see *Originating Applet of a Dynamic List*.
2. **Pick applet.** Dialog box that Siebel CRM calls to choose a value. The dialog box is a list applet that contains a scrolling list table of rows. Each row corresponds to a business component record. In the example, the Account applet is the pick applet.
3. **Originating business component.** Business component of the originating applet. This business component supplies the data that Siebel CRM displays in the originating applet. The selection process in the pick applet causes Siebel CRM to update the current record in this business component. In the example, the Opportunity Form Applet is the originating applet and it references the Opportunity business component.
4. **Pick business component.** Business component of the pick applet. Data from fields in this business component display in the list columns of the pick applet. In the example, the Account business component is the pick business component.

5. **Originating control or originating list column.** If the user clicks the originating control or list column, then the originating control or list column calls the pick applet. In the example, the Account control is the originating control.
6. **Originating field.** Field in the originating business component that the originating control references. It includes child pick maps that define how Siebel CRM maps fields from the pick business component to the originating business component. In the example, the Account field is the originating field.
7. **Pick maps.** Each pick map defines a relationship between a field in the pick business component and a field in the originating business component. If the user chooses a record, then these relationships provide the information that Siebel CRM requires to update the current, originating business component record with information from the pick business component record.

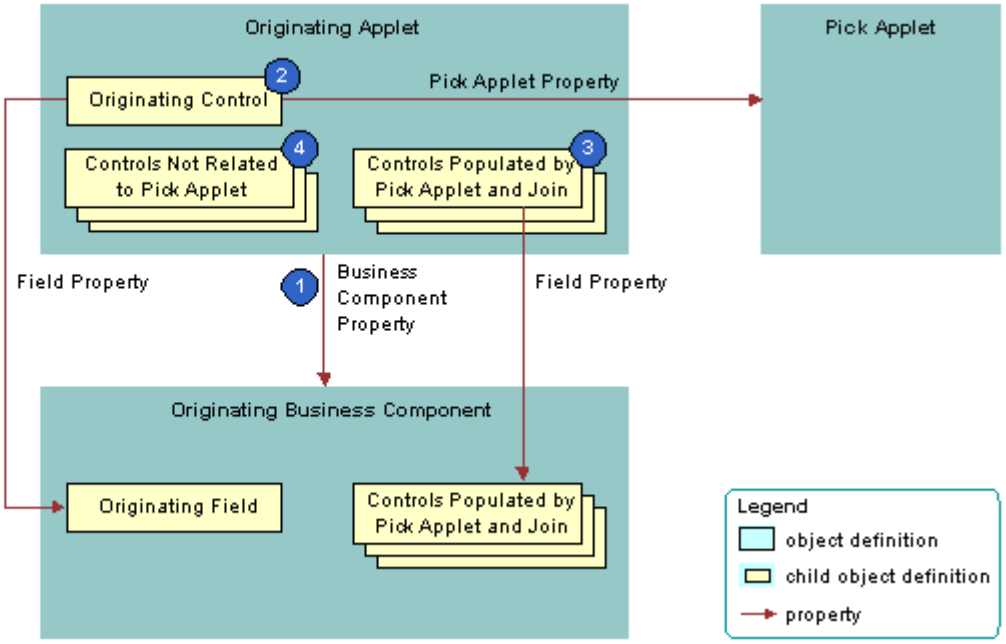
   If the user chooses a value from an unbounded list, then Siebel CRM uses the corresponding pick map that references the same field that it uses to copy the value to the field that it associates with the list. If the list is bounded, then Siebel CRM only enters data into fields that it associates with other child pick maps.

   Entering a new value in an unbounded list does not cause Siebel CRM to display the value in the list of values that the user can choose. It does not update fields in a pick map if the user chooses a value from an unbounded list. An applet that references the CSSBuscomp or the CSSBCBase class with an unbounded list does not map all the values in the pick map. To map all the values in a pick map, the list must be bounded.
8. **Pick list.** References the business component of the pick applet. In the example, the PickList Opportunity Account pick list is the list.
9. **Join and join specification.** The join is a child of the originating business component. The join specification is a child of the join. The join field references this child object. One of the pick maps updates the join field. If Siebel CRM modifies the value in the join field, then it updates all fields whose values it gets from the join. This update is not as immediate as the update that it does through the pick map. If the other pick maps are absent, then Siebel CRM does not update the data until the user navigates away from the view, and then returns to the view. In the example, S_ORG_EXT is the join and Account Id is the join specification.

Originating Applet of a Dynamic List

The originating applet contains the control or list column that calls the pick applet. It can contain other controls or list columns into which Siebel CRM enters data if the user chooses a value from the list applet. The originating applet does not require special configuration.

The following figure describes the details of the originating applet that *How Siebel CRM Creates a Dynamic List* includes.



As shown in this figure, Siebel CRM uses the following objects in the originating applet of a dynamic list:

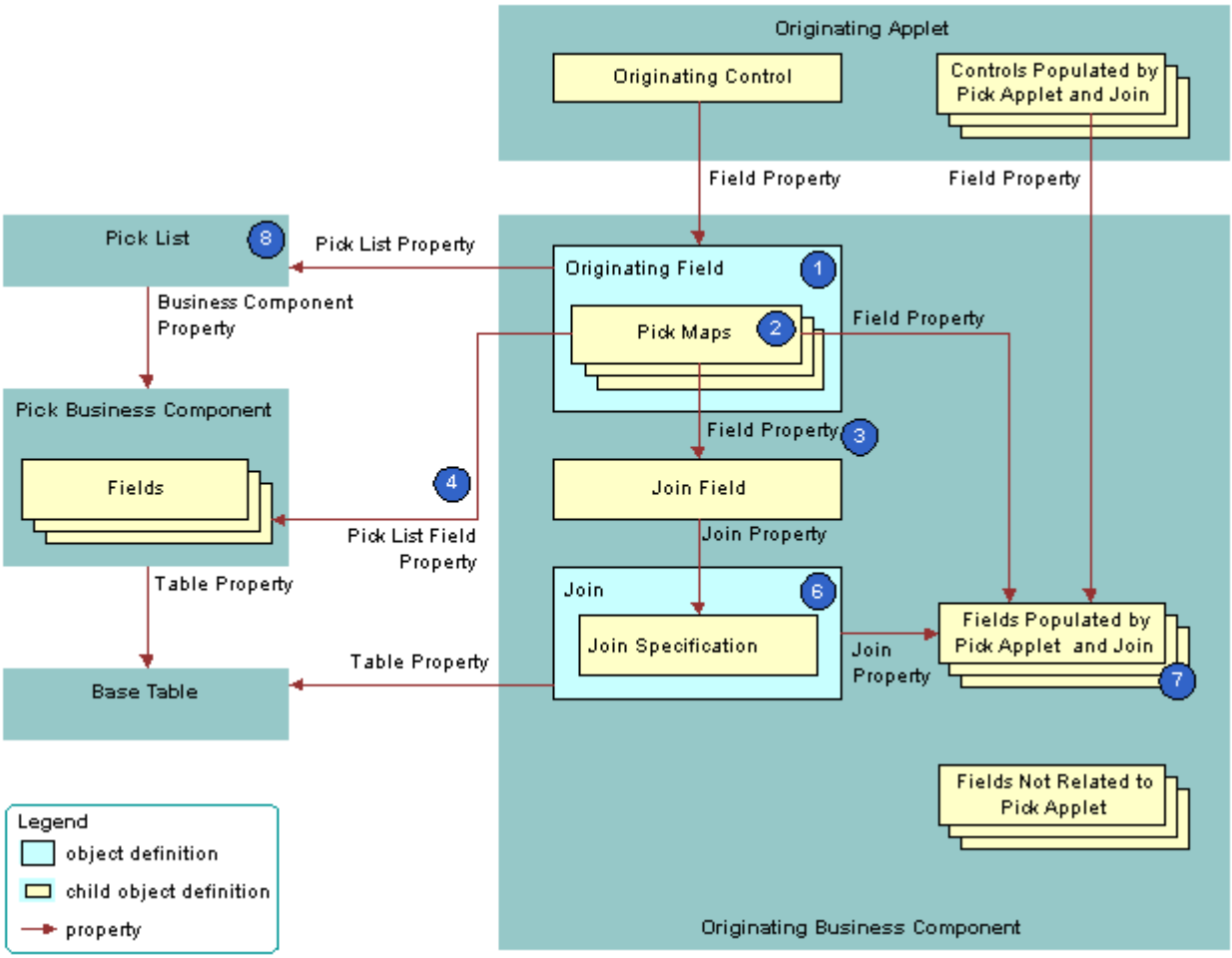
1. **Business component property.** Creates the association between the originating applet and the originating business component.
2. **Originating control.** Calls the pick applet if the user clicks the arrow. The Pick Applet property of the originating control defines the name of the pick applet. The Field property of the originating control defines the originating field. It includes definitions for the pick map child object. For more information, see *Originating Business Component of a Dynamic List*.

   The Runtime property of the control or list column must be TRUE.
3. **Controls that contain data from a pick applet and join.** If the user chooses a value from the pick applet, then Siebel CRM updates each control that contains data from a pick applet and join.
4. **Controls not related to pick applet.** Other controls in the applet.

Originating Business Component of a Dynamic List

The originating business component of a dynamic list is the business component that the Business Component property of the originating applet references. This business component supplies the data that Siebel CRM displays in the originating applet.

The following figure describes how Siebel CRM defines the originating business component of a dynamic list. This list is described in *Originating Applet of a Dynamic List*.



As shown in this figure, Siebel CRM uses the following objects in the originating business component of a dynamic list:

1. **Originating field.** Provides data to the originating control. The originating field is the parent of the pick map. The Pick List property of the field specifies the pick list. A pick map is a child of an originating field. It supports pick applets on more than one field in the business component.

   The originating field must reference a database column. You cannot associate a pick applet or list with a read-only field, including a calculated field.
2. **Pick maps.** Creates a relationship between a field in the pick business component and a field in the originating business component. This relationship provides the information that Siebel CRM requires to update the active record of the originating business component with information from the pick business component. One of the pick maps updates the join field. The join updates the business component fields that depend on the join.

   It is recommended that you test your pick map after you create it. If the value in the originating field remains the same after you choose a value from the pick applet, then you must check the pick map definition for that field.
3. **Field property.** Identifies a field in the originating business component that contains data from a field in the pick business component when Siebel CRM calls the PickRecord method.

4. **Pick List Field property.** Identifies a field in the pick business component that provides data for the field in the Field property of the pick map. Siebel CRM does the following:

   - If the user picks a value from an unbounded list, then it updates the fields in the pick map.
   - If the user types in a new value, then it does not update fields in the pick map.
   - If the user types a new value into a field that references an unbounded list, then it does not add the value to the list of values that the user can choose.

   You must not define more than one multi-value field in an originating business component that references the same destination field that the pick applet references in the Pick List Field Property. If you do this, then Siebel CRM does not display the arrow for the list and the user cannot use the list. For more information, see *About the Multi-Value Field*.
5. **Join field.** Works as a foreign key in the join that the pick applet references. Typically, the name of the join field includes the Id, such as Account Id or Key Contact Id. Siebel CRM defines it in the Source Field property of the join specification. The join field is one of the fields defined in a pick map. If the user chooses a record from the pick applet, then Siebel CRM updates the join field and all fields that reference the join.

   The pick maps initially update fields in the originating business component and the controls or list columns that reference these fields. The join and join specification do not update the contents of the applet until the user navigates away from the view, and then returns to the view.
6. **Join and join specification.** Sets up the join between the base table of the originating business component and the base table of the pick business component. Siebel CRM uses this join to update fields in the originating business component that include the name of the join in the Join property of the field.
7. **Fields that get data from the pick applet and join.** If Siebel CRM modifies the value in the join field, then it updates fields that include the name of the join in the Join property. If the user chooses a value from the pick applet, then it updates the fields that it defines in the Field property of the pick maps.

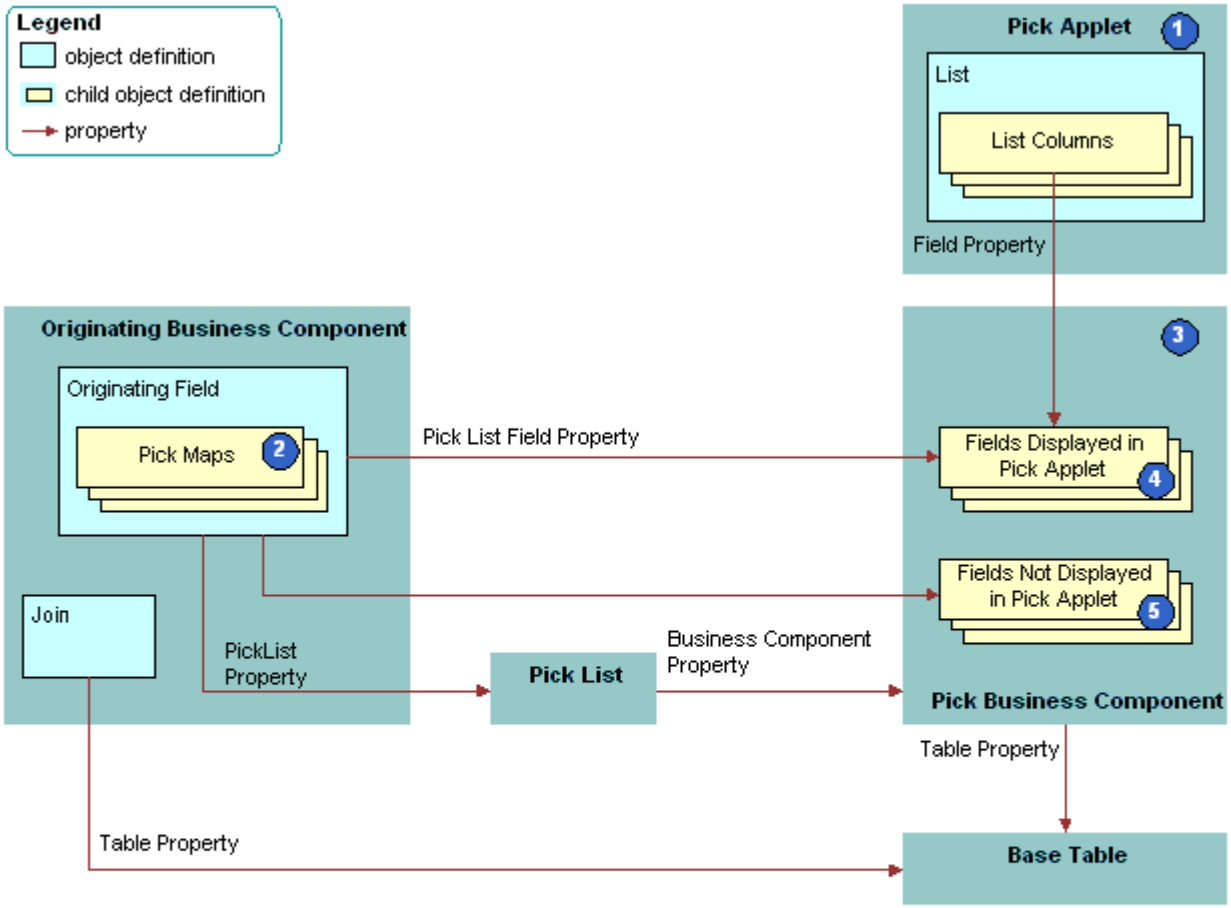
   A pick map and a join update the same fields, but an update that involves a pick map is immediate. An update that involves a join is somewhat delayed. if the user picks a record, then a pick map can update the display value of a joined field. For example, with the Account Name joined field. A pick map does not physically copy a value to the joined fields. It only writes to the foreign key field. For example, Account Id.
8. **Pick list.** The field of the originating control references the pick list. The Business Component property of the pick list references the pick business component.

   If you define a pick applet that a multi-value group applet calls, then define the list and the pick maps on the originating field in the originating business component. Do not define the list and the pick maps on fields in the child business component that the multi-value group applet references. For more information, see *How Siebel CRM Creates a Multi-Value Group*.

How Siebel CRM Constrains a Dynamic List

You can configure Siebel CRM to dynamically filter a pick applet to display only records that include a field value that matches a corresponding field in a record of the originating business component. This configuration is known as *constraining a list*. For example, you can define the pick applet for a contact that an applet that displays quotes calls so that the pick applet only displays contacts that Siebel CRM associates with the account for the current quote. For more information, see *Constraining a Dynamic List*.

The following figure describes how Siebel CRM constrains a dynamic list. It includes details of the pick business component that the second figure in the section *About Dynamic Lists* describes.



As shown in this figure, Siebel CRM uses the following objects to constrain a dynamic list:

1. **Pick applet.** Displays only contacts that contain the same account, account Id, and account location as the quote. To do this, you define a constraint pick map as a child of the Contact Last Name field. This is in addition to the predefined copy pick map object definitions that define pick behavior.
2. **Pick map.** The following types of pick maps are available:
    - **Copy pick map.** Updates the current record in the originating business component with information from the pick business component. For more information, see *Originating Business Component of a Dynamic List*.
    - **Constraint pick map.** Displays only the records that contain a matching value in a corresponding field in the originating and the pick business component. A constraint pick map causes Siebel CRM to filter

the records that it displays in the pick applet. For more information, see *How Siebel CRM Constrains a Dynamic List*.

If the Constrain property of the pick map is:

- **TRUE.** The pick map is a constraint pick map.
- **FALSE.** The pick map is a copy pick map.

3. **Pick business component.** The business component that the pick applet references. Siebel CRM displays data from fields in this business component in the list columns of the pick applet.
4. **Fields that Siebel CRM displays in the pick applet.** Enters data into the list columns in the pick applet. The Field property of the corresponding list columns in the pick applet reference these list columns. Siebel CRM includes some of the same fields in the Pick List Field property of Pick Map object definitions. These fields have a role in updating corresponding fields in the originating business component.
5. **Fields that Siebel CRM does not display in the pick applet.** Siebel CRM does not display these fields in list columns that reside in the pick applet, but it does include some of these fields in the Pick List Field property of the object definitions for a Pick Map. Such fields have a role in updating corresponding fields in the originating business component.

Constraint Pick Map Acts as a Predefault Value

A constraint pick map uses the new record that Siebel CRM adds from a pick applet as a predefault value. For example, assume the user chooses a record in the Quotes list that is associated with an account, and then clicks the Opportunities control in the form for the quote. The user then adds a new opportunity in the Pick Opportunity dialog box with no account included. Siebel CRM assigns the new opportunity to the constrained account. This situation occurs if the Constrain property of the Account Id pick map of the Opportunity field in the Quote business component is TRUE.

Foreign Key Field Must Be Constrained

If the constrained field references a joined table in the pick business component, then the foreign key field must be constrained. If the foreign key field is not constrained in this situation, and if the user creates a new record in the pick applet, then Siebel CRM displays an error that is similar to the following:

```
This operation is not available for read-only field
```

About Hierarchical Lists

A hierarchical list constrains values according to the value that the user chooses in another list. For example, in the Service Request Detail Applet, the Area and Subarea fields are lists that get their values from the S_LST_OF_VAL list of values table. The value that the user chooses in the Area list determines the values that Siebel CRM displays in the Subarea list. For more information, see *Creating a Hierarchical List*.

The list of values table creates the hierarchical relationship between these values. Siebel CRM uses the same LOV Type to determine the values for the lists in the hierarchy. For example, for Area and Subarea, the LOV Type named SR_AREA determines the values. For more information, see *Creating a List of Values*.

How Siebel CRM Creates a Hierarchy

Siebel CRM uses the Parent LIC (language-independent code) field to define a parent value. For example, consider the example list of values described in the following table.

Type	Display Value	Language Independent Code	Parent LIC
SAMPLE_LOV	1	1	None
SAMPLE_LOV	A	A	1
SAMPLE_LOV	B	B	1
SAMPLE_LOV	2	2	None
SAMPLE_LOV	C	C	2
SAMPLE_LOV	D	D	2

Assume Siebel CRM displays the values listed in the previous table in the following lists in a hierarchical relationship:

- **Parent list.** Displays the values 1 and 2.
- **Child list.** Displays values depending on the following value that the user chooses in the parent list:
- **1.** Siebel CRM displays the values A and B in the child list.
- **2.** Siebel CRM displays the values C and D in the child list.

For more information, see *Creating a List of Values*.

Creating a Hierarchical List of Values

You can create a hierarchical list of values.

To create a hierarchical list of values

- Configure the following lists:
  - Configure the parent list to reference the PickList Hierarchical business component.
  - Configure the child list to reference the PickList Hierarchical Sub-Area business component.

List and Pick Applet Configuration

This topic describes how to configure lists and pick applets. It includes the following information:

- *Using the Pick List Wizard to Create a Static List*
- *Creating a Static List Manually*
- *Using the Pick List Wizard in Web Tools*
- *Using the Pick Applet Wizard to Create a Pick Applet*
- *Using the Pick List Wizard to Create a Dynamic List*
- *Constraining a Dynamic List*

- *Creating a Hierarchical List*

Using the Pick List Wizard to Create a Static List

You can use the Pick List Wizard to create a static list.

To use the Pick List Wizard to create a static list

1. In Siebel Tools, click the File menu, and then click New Object.
2. Click the Pick List icon, and then click OK.
3. In the Pick List dialog box, enter the following information, and then click Next:
   - **Project.**
   - **Business Component.** Choose the originating business component. This is the parent business component of the field whose values display in the list.
   - **Field.**
4. In the Pick List Type dialog box, choose Static, and then click Next.
5. In the Pick List Definition dialog box, do one of the following:
   - If you must create a new list, then choose Create New Pick List, and then click Next.
   - If you must use a predefined list, then choose the list and the associated list of values you must use, click Next, and then proceed to Step 8.
6. If you must create a new list of values (LOV), then do the following:
   a. Enter a unique name for the list.
   b. Click Create New List of Values, and then click Next.
   c. In the List of Values dialog box, enter a name for the list of values, and then enter the values.

      For more information, see *Creating a New List of Values Locally*.
   d. Click Next.
7. If you must use a predefined list of values, then do the following:
   a. Enter a unique name for the list.
   b. Choose the Use Predetermined List of Values option.
   c. Choose the List of Values Type, and then click Next.
   d. In the third Pick List Definition dialog box, enter a search specification, enter a comment, and then choose to bind or not bind the list. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.
   e. Click Next.
8. In the Finish dialog box, review the specifications for the list, and then click Finish.

Creating a New List of Values Locally

If you use the Picklist Wizard to create a list of values, and if you store this list of values on your local database, then you cannot use Check In to copy this list of values to the server database. To avoid this situation, you can do one of the following:

- Use Siebel Remote to synchronize your modifications to the Siebel Server:
  a. On the computer where the S_LOV_OF_VAL records exist, open the Developer Web Client, and then connect to the local database.

b. Click the File menu, and then click Synchronize Database.
c. Make sure you clear the cache in the List Of Values view.

- Create the list of values on the Siebel Server instead of on your local development computer.

For more information, see *Creating a List of Values* and *Siebel Applications Administration Guide* .

Creating a Static List Manually

It is highly recommended that you use the Pick List Wizard to create a static list, but you can create a static list manually. For more information, see *Using the Pick List Wizard to Create a Static List* and *How Siebel CRM Creates a Static List*.

To create a static list manually

1. In Siebel Tools, click Applet in the Object Explorer.
2. In the Applets list, locate the originating applet.
3. In the Object Explorer, expand the Applet tree, and then click Control.
4. In the Controls list, add an originating control using values from the following table.

Property	Description
Field	Specify the originating field that resides in the originating business component.
Pick Applet	Leave empty.
Runtime	Set to TRUE. This setting indicates that Siebel CRM attaches and activates a static list if the user clicks the control or list column.

5. Click Business Component in the Object Explorer, and then locate the originating business component in the Business Components list.
6. In the Object Explorer, expand the Business Component tree, click Field, locate the originating field in the Fields list, and then modify the field using values from the following table.

Property	Description
PickList	Specify the pick list.

If the originating field is a custom field, then make sure that it can accommodate the LOV table values. If the originating field is shorter than the values that exist in the LOV table, then Siebel CRM truncates the values from the LOV table when it displays these values in the Siebel client or when it stores them in the Siebel database.

7. In the Object Explorer, expand the Field tree, click Pick Map, and then add a pick map in the Pick Maps list using values from the following table.

Property	Description
Field	Choose the originating field.
Pick List Field	Enter **`Value`**. This value references the Value field in the PickList Generic business component.

8. If you use a multiple column selection list, then configure more pick maps, as required.
9. In the Object Explorer, click Pick List, and then create a new pick list in the Pick Lists list using values from the following table.

Property	Description
Business Component	Choose PickList Generic. This value indicates that Siebel CRM gets the list of values from a system table. For more information, see *About the Picklist Generic Business Component*.
Type Field	Choose Type. This value configures Siebel CRM to search the Type field in the PickList Generic business component for types. Each list of values includes a type that uniquely identifies the list and each value in the list.
Type Value	Enter the relevant type for the list of values. For example, the values that display in the Lead Quality list in the table in the section *About the Picklist Generic Business Component About the Picklist Generic Business Component* include a Type Value property whose value is LEAD_QUALITY.
Search Specification	In most situations, you can leave the Search Specification property empty. For more information, see *How Siebel CRM Handles a Hierarchy of Search Specifications*.
Sort Specification	Do one of the following: o To use the sort specification that is defined for the business component, leave the Sort Specification property empty. o To override the sort specification that is defined for the business component, define a value in the Sort Specification property. For more information, see *Creating a Sort Specification for a Static List*.

Property	Description
No Insert	Make sure the No Insert property contains a check mark. If the No Insert property does not contain a check mark, then Siebel CRM creates an error that is similar to the following: `Unable to create list popup applet`

Creating a Sort Specification for a Static List

You can define a Sort Specification on a list to override the sort specification that is defined on the business component. The default value for the Sort Specification property in a list is empty, so Siebel CRM uses the sort that is defined on the business component. It uses the Order By field in a Type to sort the list of values in ascending order, by default. If the Order By values are empty, then it sorts the entries for the Type alphabetically in ascending order according to the Value field.

You can specify a sort specification on a static list to modify this behavior. This modification applies only to the static list. A sort specification on a list object sorts values in the static list that references the list of values in the PickList Generic business component. For more information, see *How a Business Component Sorts Records*.

Using Calculated Fields with Bounded Lists

It is recommended that you do not configure a list of values that references a calculated field if this field references another field that the user can update. For example, Siebel CRM comes predefined with a list named PickList Activity Priority that resides on the Priority field in the Action business component. The following properties of this list include a check mark:

- No Insert
- Static
- Bounded

This list references the ACTIVITY_PRIORITY list of values that includes the following values:

- 1-ASAP
- 2-High
- 3-Medium
- 4-Low

This configuration requires the user to choose a value from the list of values. The user cannot manually add a value. However, if you configure the ACTIVITY_PRIORITY list of values to reference a calculated field, and if this field references another field that the user can update, and if the Predefault property of the business component contains a check mark, then Siebel CRM might allow the user to choose any value that the user enters in this updatable field. In this example, the user could add to the Priority field any value that the updatable field contains.

Using the Pick List Wizard in Web Tools

Web Tools provides a wizard to create a new Pick List. You must have an editable Workspace open in Web Tools to create a new Pick List. There are five combinations of Pick Lists that you can create using the wizard.

- *To create a new Static Pick List using the wizard*
- *To create a new Static Pick List using existing List of Values*
- *To re-use an existing Static Pick List using the wizard*
- *To create a new Dynamic Pick List using the wizard*
- *To re-use an existing Dynamic Pick List using the wizard*

To create a new Static Pick List using the wizard

1. Click the magic wand icon in the Web Tools toolbar.
2. Click Pick List.
3. In the Enter Data view choose a Project.
4. Choose the Business Component that will host the new Pick List.
5. Choose the Field that will host the new Pick List.
6. Click on **Static Type**.
7. Click on Create new Pick List.
8. Click Next.
9. Give the Pick List a name.
10. Select **Create New**.
11. Give the new List of Values a name.
12. Add all the values that should display in the Pick List.
13. Click Next.
14. **Optional**: Add a Search Specification, Sort Specification, or Comment.
15. Click **Next** to get a summary of your choice.
16. Click Finish.

To create a new Static Pick List using existing List of Values

1. Click the magic wand icon in the Web Tools toolbar.
2. Click Pick List.
3. In the Enter Data view choose a Project.
4. Choose the Business Component that will host the new Pick List.
5. Choose the Field that will host the new Pick List.
6. Choose **Static Type**.
7. Choose **Create new Pick List**.
8. Click Next.
9. Give the Pick List a name.
10. Choose **Use Predefined** to use LOVs already in existence.
11. Search for the List of Values Type that you will use in the Pick List. The values for the chosen LOV type will display on the left.
12. Click Next.
13. **Optional**: Add a Search Specification, Sort Specification, or Comment.
14. **Optional**: Check the **Bounded** checkbox if you wish to make this Pick List read only.

15. Click Next to get a summary of your choices.
16. Click Finish.

To re-use an existing Static Pick List using the wizard

1. Click the magic wand icon in the Web Tools toolbar.
2. Click Pick List.
3. In the Enter Data view choose a Project.
4. Choose the Business Component that will host the new Pick List.
5. Choose the Field that will host the new Pick List.
6. Select Static Type.
7. Click on Use existing Pick List.
8. Click Next.
9. Search for the Pick List you wish to re-use.
10. Click **Next** to get a summary of your choices.
11. Click Finish.

To create a new Dynamic Pick List using the wizard

1. Click the magic wand icon in the Web Tools toolbar.
2. Click Pick List.
3. In the Enter Data view choose a Project.
4. Choose the Business Component that will host the new Pick List.
5. Choose the Field that will host the new Pick List.
6. Choose **Dynamic Type**.
7. Select Create new Pick List.
8. Click Next.
9. Choose the Business Component that will supply the records to pick.
10. Choose the Field in the Business Component that will supply the data.
11. Give the new Pick List a name.
12. **Optional**: Choose the update attributes.
13. **Optional**: Add a Search Specification, Sort Specification, or Comment.
14. Click Next.
15. Define a Pick Map. A Pick Map maps the data from the business component from which you pick the records to the fields that will hold the data in the host Business Component.
16. Click **Next** to get a summary of your choices.
17. Click Finish.

To re-use an existing Dynamic Pick List using the wizard

1. Click the magic wand icon in the Web Tools toolbar.
2. Click Pick List.
3. In the Enter Data view choose a Project.
4. Choose the Business Component that will host the new Pick List.
5. Choose the Field that will host the new Pick List.
6. Choose **Dynamic Type**.
7. Choose Use existing Pick List.
8. Click Next.
9. Search for the Pick List you wish to re-use.
10. Click Next.

11. Define a Pick Map. A Pick Map maps the data from the business component from which you pick the records to the fields that will hold the data in the host Business Component.
12. Click **Next** to get a summary of your choices.
13. Click Finish.

Using the Pick Applet Wizard to Create a Pick Applet

You can use a predefined pick applet to display a dynamic list in the Siebel client. You can use the Pick Applet Wizard to create a custom list.

To use the Pick Applet Wizard to define a pick applet

1. Click the File menu, and then click New Object.
2. Click the Applets tab, click the Pick Applet icon, and then click OK.
3. In the General dialog box, define the following properties, and then click Next:
    - Project.
    - Pick business component.
    - Name for the Picklist Applet:
        - Use the following format to name a pick a applet: *business component name* Pick Applet.
        - Use the following format to name an association applet: *business component name* Assoc Applet.
    - Display Name.
4. In the Web Layout General dialog box, choose the templates to use for the Base and Edit List modes, and then click Next.

   For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.
5. In the Web Layout - Fields dialog box, choose the fields that Siebel CRM must display in the pick applet, and then click Next.
6. In the Second Web Layout - Fields dialog box, choose the controls that Siebel CRM must display in the pick applet, and then click Next.

   Siebel Tools displays all controls in the Selected Controls, by default. These controls reference the Model Pick Applet in the Siebel repository. To move a control to the Available Controls window, choose the control, and then click the left/back arrow.
7. In the Finish dialog box, review the information, and then click Finish.

   The Pick Applet Wizard creates the pick applet object, and then opens the Web Layout Editor that you can use to map list columns and controls to placeholders in the web template.

   For more information, see *Configuring Applet Layouts*.

Using the Pick List Wizard to Create a Dynamic List

You can use the Pick List Wizard to create a dynamic list and related objects. These objects include the following:

- **Pick List.** Defines the properties of the list, including the originating business component and the pick business component. For more information, see *How Siebel CRM Creates a Dynamic List*.

- **Pick Map.** Child object of a business component field that maps the source field in the pick business component with the target field in the originating business component.
- **Pick Applet.** Pop-up applet that allows you to display the list of records where the user can choose a value.

The values of the Visibility Type and Visibility Auto All properties of the list object override the pop-up visibility properties on the business component. For more information, see *Siebel Security Guide* .

To use the Pick List Wizard to create a dynamic list

1. In Siebel Tools, click the File menu, and then click New Object.
2. Click the Pick List icon, and then click OK.
3. In the Pick List dialog box, enter the following information, and then click Next:
   - Project.
   - Business Component. Define the originating business component. This is the parent business component of the field that displays the list.
   - Field.
4. In the Pick List Type dialog box, choose Dynamic.

   Note the following:
   - A static list gets values from a predefined list of values.
   - A dynamic list gets values from a business component.

   For more information, see *Using the Pick List Wizard to Create a Static List*.
5. In the Pick List Definition dialog box, choose to use a new list or a predefined list:
   - If you must create a new list, then do the following:
     - Choose Create a New Pick List.
     - Click Next, and then proceed to Step 6.
   - If you must use a predefined list, then do the following:
     - Choose Use Existing Pick List.
     - Choose the predefined list from the Existing Pick Lists window.
     - Click Next, and then proceed to Step 8.
6. In the Pick List Definition dialog box, enter information using values from the following table, and then click Next.

Property	Description
Business component	Choose the pick business component.
Business component field	Choose the field that Siebel CRM must use to sort the list.
Name	Enter a name for the new PickList object. Use the following format: ABC PickList *entity*

Property	Description
	It is not necessary to repeat the entity name if the name already includes a prefix. For example, a PickList object that reference the MS Subsegment business component is ABC PickList Subsegment. It is not MS PickList MS Subsegment.
Search specification	As an option, you can specify a search specification. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.

7. In the Pick List Specifications dialog box, define the actions, such as No Delete, that the user can perform on Siebel data.

   You can leave all options without a check mark.
8. In the Pick Map dialog box, do the following:
   a. Choose the source field in the originating business component.
   b. Choose the target field in the pick business component.
   c. Click Add.
9. Click Next, verify the information in the Finish Dialog box, and then click Finish.

Constraining a Dynamic List

The Constrain property of a pick map defines a constraint for a pick applet. For example, to configure a Country list to display only states that are part of that country, you must indicate the relationship between each state and the country where the state resides. Siebel CRM displays the State list after the user chooses a value from the Country list. The value that the user chooses from the Country list constrains the values in the State list. It uses the value that the user chooses from the Country list to filter the values that it displays in the State list. It only displays values in the State list if the Description field contains the value that the user chooses from the Country list.

To constrain a dynamic list

1. To define the constraint, do one of the following:
   - Use the predefined Description field in the Picklist Generic business component.
   - Extend the table and use a new column.
2. To update the Description field with valid country values, do one of the following:
   - Do the following in the Siebel client:
     - Navigate to the Administration - Data screen, List Of Values view.
     - Enter valid country values in the Description field.
   - Do the following in Siebel Tools:
     - In the Object Explorer, click Business Component, and then locate the Account business component in the Accounts list.
     - In the Object Explorer, expand the Business Component tree, click Field, and then locate the State field in the Fields list.
     - In the Object Explorer, expand the Field tree, and then click Pick Map.

- In the Pick Maps list, add a new record using values from the following table.

Property	Value
Field	Country
Constrain	True
Pick List Field	Description

Creating a Hierarchical List

This topic describes how to create a hierarchical list.

To create a hierarchical list

1. In Siebel Tools, define a parent list using values from the following table.

Property	Value
Business Component	Picklist Hierarchical

2. Create a child list using values from the following table.

Property	Value
Business Component	PickList Hierarchical Sub-Area

3. In the Object Explorer, click Business Component.
4. In the Business Components list, locate the business component that contains the fields that you must associate with the hierarchical list.
5. In the Object Explorer, expand the Business Component tree, and then click Field.
6. In the Fields list, locate the parent field, and then set properties using values from the following table.

Property	Description
Picklist	Set to the parent list.
Immediate Post Changes	Make sure this property contains a check mark.

7. In the Fields list, locate the child field, and then set properties using values from the following table.

Property	Description
Picklist	Set to the child list.

8. In the Object Explorer, expand the Field tree, and then click Pick Map.
9. In the Pick Maps list, create a new pick map using values from the following table.

Property	Description
Field	Choose the name of the parent field.
PickList Field	Choose the name of the parent field.
Constrain	Make sure this property contains a check mark.

10. In the Pick Maps list, create another new pick map using values from the following table.

Property	Description
Field	Choose the name of the child field.
PickList Field	Choose the Value field.
Constrain	Leave empty.

11. Test and then deliver your Workspace.
12. To designate the parent value, add LOV values to the Parent LIC column.

    For more information, see the table in the topic *About Hierarchical Lists* and the topic about constrained lists of values in *Siebel Applications Administration Guide* .
13. Test the hierarchical list.

Creating a List of Values

A *list of values* is a set of values that Siebel CRM uses to enter values in a static list. If the user chooses a static list, then it displays a list of values. The user can choose a value from the list to cause Siebel CRM to enter values into the field.

Siebel CRM stores the values in a list of values as records in the S_LST_OF_VAL table. A list of values includes the following parts:

- **Header Row.** Defines the name of the type grouping. For example, the first row in the following table. This type grouping name is the value of the Display Value property, such as ACCOUNT_STATUS. The row includes a Display Value property, but Siebel CRM does not display the header row in the list of values. The Display Value property for a header row defines only the name for the list of values. It does not display any strings in the Siebel client.
- **Display Value Rows.** Includes the values that Siebel CRM displays in a list that references the list of values. The rows in the following table where Type is ACCOUNT_STATUS are examples of display value rows. These rows contain the display values that Siebel CRM displays in the Siebel client. The Type property for each display value row is ACCOUNT_STATUS, which is the same as the Display Value of the Header Row.

The Type field groups List of Value records. For example, the Type value is ACCOUNT_STATUS for values that the Status field of the Account Entry Applet includes.

A picklist object includes a Type property that identifies the LOV Type that is associated with the list. Siebel CRM reads this information to determine the list of values that it displays for a list in the Siebel client. For more information, see *About Static Lists*.

The following table lists the values that belong to the LOV Type defined as ACCOUNT_STATUS.

Type	Display Value
LOV_TYPE	ACCOUNT_STATUS
ACCOUNT_STATUS	Candidate
ACCOUNT_STATUS	Qualified
ACCOUNT_STATUS	Active
ACCOUNT_STATUS	Inactive

Creating a New List of Values

Siebel CRM comes with many predefined lists of values that support the static lists that it displays in the Siebel client. You can modify a predefined list of values, or you can create a new one:

- You can use the List of Values view in the Administration - Data screen in the Siebel client to modify a list of values in Siebel Tools. In Web Tools, you can do the same from List of Values in the Tools menu. You can add, modify, or deactivate LOV values for LOV types for predefined list of values. For example, you can add another value to the ACCOUNT_STATUS LOV type. For more information, see *Siebel Applications Administration Guide* .
- You can create a new list of values in Siebel Tools and Web Tools. If you must add a new set of values that you define as a new LOV Type, then you must use Siebel Tools or Web Tools.

To create a new list of values

1. In Siebel Tools, click the Screens application menu, click System Administration, and then click the List of Values menu item.

Note: In Web Tools, click the Tools menu and then click the List of Values menu item.

2. In the List of Values list, create a header record for the new LOV Type using values from the following table.

Field	Description
Type	Enter **LOV_TYPE**.
Display Value	Enter the name of the LOV Type. For example, **ACCOUNT_STATUS**. Do not use single quotes in the Display Value property. Single quotes cause search specifications that reference the Display Value field to fail. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.
Translate	Do not modify this property. For more information, see *Modifying the Translate Property*.
Language Independent Code	In most cases, enter the same value that you enter for the display value. For more information, see *Overview of Language-Independent Code*.

3. Enter a new record for the LOV value using values from the following table.

Property	Description
Type	Choose the name of the LOV type that you created in Step 2. For example ACCOUNT_STATUS. The value you define for this property must match the value you define in the Type property of the list that you configure to display these values.
Display Value	Enter the value that Siebel CRM must display in the list.
Language Independent Code	In most cases, enter the same value that you enter for the display value.
Translate	Do not modify this property. For more information, see *Modifying the Translate Property*.
Language Name	Choose the name of the language for the Display Value.

You use some properties only for a multilingual list of values, such as Translate, Multilingual, and Language-Independent Code. For more information, see *Defining Properties of an MLOV*. For a complete description of LOV fields, see *Siebel Applications Administration Guide* .

4. Repeat Step 3 for each LOV value.

5. Create a list to display the LOV Type.
   For more information, see *Configuring Lists and Pick Applets*.
6. Test and then deliver your Workspace.
   Make sure you clear the cache. For more information, see Step 7 in *Associating an Organization with a List of Values* .

Associating an Organization with a List of Values

This topic describes how to associate an organization with a list of values. It includes the following information:

- *Guidelines for Associating an Organization with a List of Values*
- *Guidelines for Using Script to Associate a List of Values with an Organization*
- *Creating a Value to Display for More Than One Organization*
- *Using the Organization Specifier Property to Display Custom Lists of Values*

You can define a list of values to display for some organizations but not for other organizations. For example, assume your company includes several subsidiary companies and each subsidiary is defined as an organization in your Siebel deployment. For a list, you can display a different list of values for each member of each organization. To do this, you associate each list of values to an organization.

For example, the organization associated with the active position of a user might be Org ABC, but the primary organization that is associated with the record that the user is viewing might be Org XYZ. In this situation, Siebel CRM displays the list of values that it associates with Org XYZ.

For more information, see *Guidelines for Associating an Organization with a List of Values*. For more information about organizations and access control, see *Siebel Security Guide* .

To associate an organization with a list of values

1. In the Siebel client, choose the site map, click Administration - Data, and then click LOV Explorer.
   Siebel CRM displays the LOV types in a tree. You can expand each LOV type to view the LOV values it associates with each LOV type.
2. In the List of Values - Type list, query the Type field for the LOV type that requires LOV values that are specific to the organization.
3. Click the Organization field, and then click Select.
4. In the Organizations dialog box, choose the organizations you must add, click Add, and then click OK.
5. In the LOV explorer, expand the Types folder, and then expand the Values folder.
6. In the List of Values list, create a set of LOV values for each organization:
   a. In the List of Values list, click New.
   b. Enter a value in the Display Name field and Code field.
      The code is typically the same as the display name.
   c. Choose an organization to associate with the LOV value.
   d. Repeat for each LOV value that you must associate with an organization.
      You can associate each LOV value with only one organization. If you must associate a value with more than one organization, then you must create a duplicate value for each organization.

If a LOV Value is not associated to an organization, then it is available to all organizations, except the organizations that Siebel CRM associates with the LOV Type in Step 4.

7. Click Clear Cache.

   The list of value modifications take effect after you clear the cache.

Guidelines for Associating an Organization with a List of Values

If you associate an organization with a list of values, then use the following guidelines:

- Identify all LOV types that require lists of values that Siebel CRM associates with an organization. For each of these LOV types, do the following:
  - Identify and use predefined lists of values. These are the values that all organizations use. They do not require custom lists of values.
  - Identify the organizations that require custom lists of values. For each organization, define the custom lists of values for the organization.
- For a large deployment, use Enterprise Integration Manager to load list of values data that is specific to an organization. Make sure to associate the appropriate organizations with the LOV types and a single organization with each LOV value. For more information, see *Siebel Enterprise Integration Manager Administration Guide* .
- Explicitly associate each list of values with each organization. A list of values that is associated with an organization is associated with only one organization. Organization hierarchy does not determine inheritance between lists of values. For example, a list of values that is associated with a parent organization does not mean that all child organizations inherit access to the list of values.
- After an upgrade, review your custom lists of values to make sure that any predefined lists of values that come with the upgrade do not interfere with your custom lists of values.
- If you associate an MLOV with an organization, then make sure one of the following situations is true:
  - The values for the Language Independent Code property and the Display Value property are distinct from all other records.
  - The values for the Language Independent Code property and the Display Value property are the same as another record that belongs to another organization.

Guidelines for Using Script to Associate a List of Values with an Organization

If you use script to associate a list of values with an organization, then use the following guidelines.

- If you use LookupValue or LookupName as an expression in a script, and:
  - **Data does not exist.** Siebel CRM uses the organization that is associated with the current position of the user to determine visibility of the list of values. Creating a new record is an example where data does not exist.
  - **Data does exist.** Siebel CRM uses the primary organization that is associated with the record to determine visibility of the list of values.
  - If you use LookupValue or LookupName as a function in a repository configuration or a script, then Siebel CRM uses the organization that is associated with the primary position of the user to determine visibility of the list of values.

Note: To use scripts, go to Siebel Tools.

Creating a Value to Display for More Than One Organization

If you require the same value to display for more than one organization that is associated with the LOV type, then you must create duplicate values for each organization. Siebel CRM displays a list of values that is associated with an organization to members of that organization only. For example, assume the following:

- *Value 1* is associated with *Org ABC*.
- *Value 2* is associated with *Org XYZ*.
- *Value 3* is not associated with any organization.

In this example, Siebel CRM displays value 3 for all organizations except for *Org ABC* and *Org XYZ*. For *Value 3* to display for *Org ABC* and *Org XYZ*, you must create duplicate values, and then add them to the lists of values that are specific to the organization, one assigned to *Org ABC* and one assigned to *Org XYZ*.

To create a value to display for more than one organization

1. Create duplicate values.
2. Add these values to the lists of values that are specific to each organization.

Using the Organization Specifier Property to Display Custom Lists of Values

If the user chooses an existing record, then Siebel CRM uses the primary organization that is associated with the record to determine the organization context. It does not use the organization that is associated with the position of the user. The Owner Organization Specifier property of the base table that the business component references specifies the

column that contains the organization Id. Siebel CRM defines this property on the primary extension table for business components that reference the S_PARTY table.

The Organization Specifier property in most tables reference the column that contains the primary organization Id. For example, the S_ORG_EXT table references BU_ID. You can define this property in several levels. This configuration allows you to define a child business component so that it uses the organization context from the row in the parent business component. For example, if the user creates a child record, then the value of the column defined as the Owner Organization Specifier determines the lists of values that Siebel CRM displays.

The following is an example of the Organization Specifier property defined with several levels:

[S_TAB_X][S_TAB_COL1][S_TAB1_COL2]

In this example, each element is one of the following:

- A column in the current table
- The name of an extension table
- The name of an FK column
- The name of the column that contains the BU_ID

To use the organization specifier property to display custom lists of values

1. Identify the objects involved in the configuration.

   For example, the columns, extension tables, and FK columns.
2. Define the Organization Specifier property.

19 Configuring Multi-Value Group, Association, and Shuttle Applets

Configuring Multi-Value Group, Association, and Shuttle Applets

This chapter describes how to configure multi-value group applets, association applets, and shuttle applets. It includes the following topics:

- *Creating Multi-Value Groups and Multi-Value Group Applets*
- *About Association Applets*
- *About Shuttle Applets*
- *Creating a Shuttle Applet*

For more information, see *Creating an Applet*.

Creating Multi-Value Groups and Multi-Value Group Applets

This topic describes how to create multi-value groups and multi-value group applets. It includes the following information:

- *About the Multi-Value Group Applet*
- *How Siebel CRM Creates a Multi-Value Group*
- *Guidelines for Creating Multi-value Group Applets and Pick Applets*
- *Creating a Multi-Value Group*
- *Creating a Multi-Value Group Applet*

For more information, see the following topics:

- *How Siebel CRM Sorts a Multi-Value Field*
- *About Links*
- *About Multi-Value Links*

About the Multi-Value Group Applet

A multi-value group applet lists records from the detail business component that the multi-value group references. These records are child records in the parent-child relationship with the record of the master business component. The multi-value group applet does the following:

- Contains list columns that display data from corresponding fields in the detail business component

- Allows the user to add or delete detail records

For more information, see *Multi-Value Group, Multi-Value Link, and Multi-Value Group Applet*.

Viewing an Example of a Multi-Value Group Applet

You can view an example of a multi-value group (MVG) applet.

View an Example of a Multi-Value Group Applet

1. In the Siebel client, select the Account screen, and then the Accounts List.
2. In the Account Entry applet, click the Select button that's located after the Address field.

Siebel CRM displays the Account Addresses MVG applet. This applet lists the detailed Address records that Siebel CRM associates with the master account record. This dialog box lists the address information that's associated with each account, including the street address, city, state, and ZIP code.

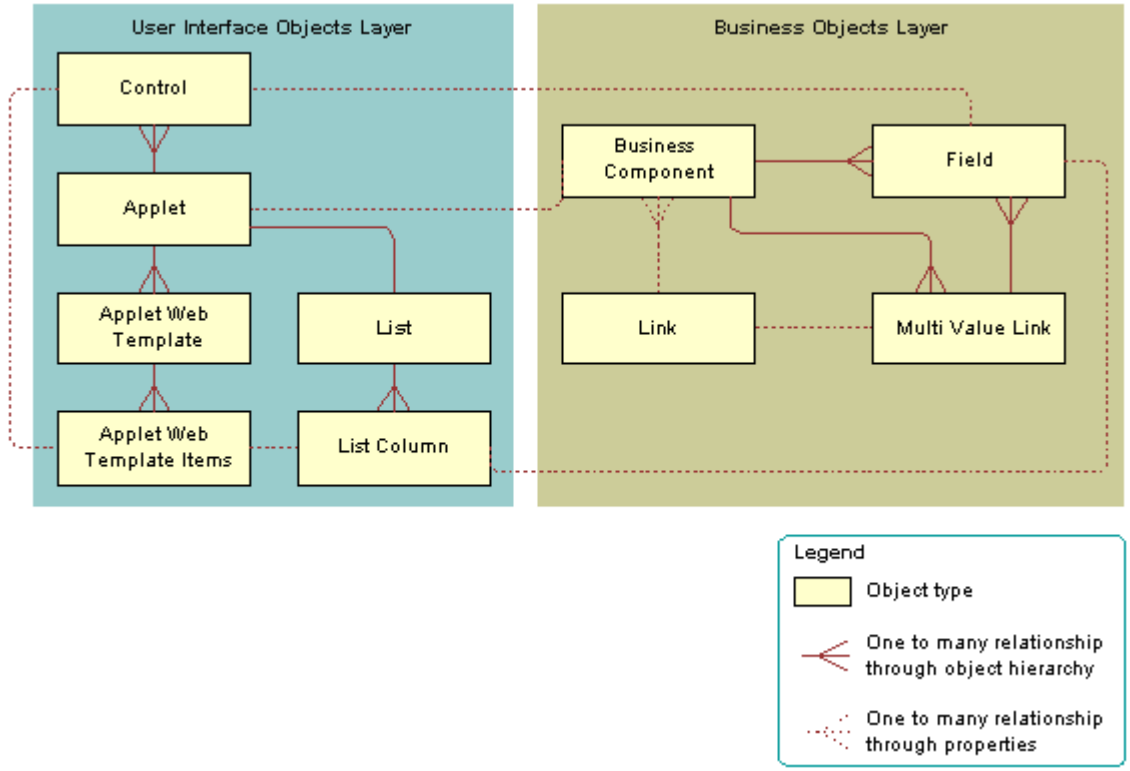
A check mark in the Primary column indicates that Siebel CRM displays data from this record in the Address field of the Account Entry applet. While the Account Addresses MVG applet is open, you can view the list of addresses that Siebel CRM associates with the account. You can add, query, or delete an address.

Relationships and Objects of a Multi-Value Group Applet

The list column is a child of the list. It includes a Field property that identifies the field in the detail business component that the multi-value group references. Siebel CRM displays this data in the list column. For more information, see *How a Business Component Field Identifies the Type of Data*.

The following figure shows the relationships and objects that Siebel CRM uses in a multi-value group applet, which are as follows:

- **User Interface Objects Layer.** Contains the following objects: Control, Applet, Applet Web Template, Applet Web Template Items, List, List Columns. There is a 1:M relationship between Applet and Control, Applet and Applet Web Template, Applet Web Template and Applet Web Template Items (through properties), List and List Column. There is a 1:1 relationship between Applet and List.
- **Business Objects Layer.** Contains the following objects: Business Component, Field, Link, Multi-Value Link. There is a 1:M relationship between Business Component and Field, Business Component and Multi-Value Link, Link and Business Component (through properties), Multi-Value Link and Field. There is a 1:1 relationship between Link and Multi-Value Link (through properties).



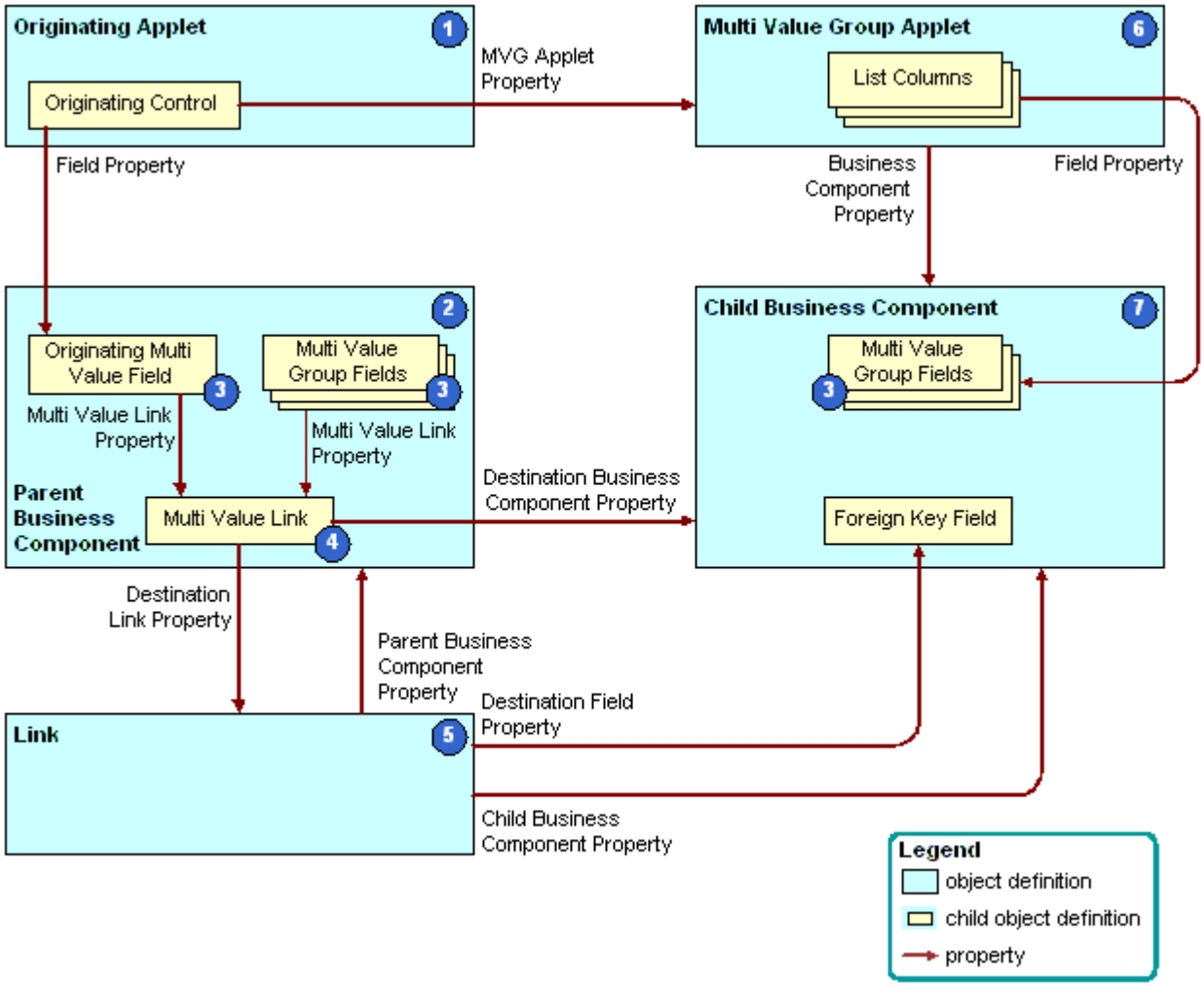
Properties of a Multi-Value Group Applet

The following table describes some of the properties of the multi-value group applet.

Property	Description
Business Component	Identifies the detail business component that the multi-value group references.
Class	CSSFrameList, which indicates that this is a predefined list applet.
Type	MVG, which indicates that this is a multi-value group applet. This property defines the behavior of the dialog box and button controls.
Title	Identifies the name of the multi-value group applet that Siebel CRM displays in the title bar.

How Siebel CRM Creates a Multi-Value Group

This figure describes the objects and properties that Siebel CRM uses to create a multi-value group (MVG) applet:



As shown in the figure, Siebel CRM uses the following objects to create a MVG applet:

1. **Originating applet.** Contains the control or list column that calls the MVG applet. For more information, see *Originating Applet of a Multi-Value Group*.
2. **Parent business component.** Supplies data to the originating applet. For more information, see *Parent Business Component of a Multi-Value Group*.
3. **Multi-value fields.** Includes the fields that constitute a MVG. For more information, see *About the Multi-Value Field*.

   If the field is a multi-value field, then Siebel CRM ignores the Required field. In this situation, you can do one of the following:

   - Use a script in Siebel Visual Basic or Siebel eScript.
   - Create a calculated field that references the multi-value field, and then make the calculated field required.
4. **Multi-value link.** Identifies the link that provides the field values from the child business component that the MVG references.
5. **Link.** Specifies the parent-child relationship between the parent business component and the child business component that the MVG applet references. To allow the fields in the parent business component to get their values, the multi-value link references the link.

6. **Multi-value group applet.** A dialog box that Siebel CRM displays if the user clicks the ellipsis button in the originating applet. It lists the records of the child business component that the MVG references. It allows the user to add, edit, or delete a child record.
7. **Child business component.** Stores the child records. The records that Siebel CRM displays in the MVG applet are the records of the child business component that the MVG references. For more information, see *Child Business Component of a Multi-Value Group*.

Example of Objects Siebel CRM Uses to Create a Multi-Value Group Applet

The following table describes some of the objects that Siebel CRM uses to create a multi-value group applet for the Account Address Mvg Applet.

Object	Name of Object Definition
Originating applet	Account Entry Applet
Parent business component	Account
Multi-value fields	This example includes the following multi-value fields: • Street Address • Address Id • City • Country • Fax Number • Postal Code • State
Multi-value link	Business Address
Link	Account/Business Address
Multi-value group applet	Account Address Mvg Applet
Child business component	Business Address

Originating Applet of a Multi-Value Group

The originating applet contains the control or list column that calls the multi-value group (MVG) applet. The Business Component property of the originating applet identifies the parent business component. The originating control or list column is a child of the originating applet.

This table describes the some properties of the originating control or list column:

Property	Description
Field	Identifies the originating field in the originating business component.

Property	Description
MVG applet	Name of the MVG applet to call.
Runtime	Must be set to TRUE.

Parent Business Component of a Multi-Value Group

The parent business component is the business component of the originating applet. Siebel CRM gets the data values that the originating field includes and other multi-value fields from corresponding fields in a record in the child business component that the multi-value group (MVG) references.

The primary is the record where Siebel CRM gets these values.

The parent business component doesn't include any properties that Siebel CRM requires to define a MVG. The field and multi-value link (MVL) child objects are significant.

The originating field is the field defined in the Field property of the originating control or list column. Other than the relationship with the originating control, the role of the originating field is identical to that of the other multi-value fields that share the MVL. For more information, see *About the Multi-Value Field*.

About the Multi-Value Field Pick Map

You can use a pick map for a multi-value field similarly to how you use it for a single-value field. The *MVF pick map* is an object that's a child of a multi-value field. Each pick map defines a relationship between a field in the child business component that the MVG references and one in the originating business component. If the user selects a record, then these relationships provide the information that Siebel CRM requires to update the record in the parent business component with information from the MVG business component.

This table describes some properties of the multi-value field pick map.

Property	Description
Field	Identifies a field in the parent business component where Siebel CRM enters data. Siebel CRM uses data from a field in the MVG business component when it calls the PickRecord method.
Pick List Field	Identifies a field in the MVG business component that's the source of data for the field in the Field property of the pick map.

The State multi-value field of the Account business component is an example of how Siebel CRM uses the multi-value field pick map. The Account business component includes a MVL to the Business Address business component, where it gets address information.

For more information, see *About Multi-Value Links* and *About Links*.

Child Business Component of a Multi-Value Group

The child business component of a multi-value group stores the child records of the parent-child relationship with the parent business component. Siebel CRM gets the records that it displays in the multi-value group applet from the child

business component. The child business component includes no important properties with respect to defining a multi-value group. It includes child field objects that Siebel CRM uses in the following ways:

- **To store data for a field in the multi-value group.** A list column in the multi-value group applet represents each field that fulfills this role. To supply data to a corresponding field in the parent business component, it might participate in the multi-value link.
- **To identify the primary record in the multi-value group.** The primary field that is defined in the Primary Field Id property of the multi-value link identifies the primary records.

The primary field is relevant to the parent business component, the multi-value link, and the multi-value group applet. The primary field has nothing to do with the child business component that the multi-value group references.

- **As the destination field of the link.** The field with this role is a foreign key to the parent business component.

For more information, see *Activating a Multi-Value Field*.

Guidelines for Creating Multi-value Group Applets and Pick Applets

If you configure an applet web template for a multi-value group applet or pick applet with a control or list column, then use the following guidelines:

- Use Base mode to display the primary value in the multi-value group applet, and to suppress the display of a link that the user can click to pop-up the multi-value group applet.
- To display the primary record from the multi-value group as read-only text, and to display a link after the text that the user can click to pop-up the multi-value group applet., use Edit, New, or Edit List mode. If the user clicks the link, then Siebel CRM displays the multi-value group applet in a separate pop-up window. You must make sure the control or list column is editable.
- Use the EditFieldCaption and EditFieldType parameters in the configuration file to set the style of the link.
- You must make sure an Edit List or Base template is defined for the multi-value group applet:
  - If an Edit List template is defined, then Siebel CRM uses this template to display the applet.
  - If an Edit List template is not defined, then Siebel CRM uses the Base template.
  - If an Edit List template is not defined, and if a Base template is not defined, then Siebel CRM creates an error.
- You can call methods, such as EditRecord, AddRecord, or CreateRecord. The multi-value group applet behaves like any other list applet in the pop-up window. Siebel CRM displays the appropriate template in the current pop-up window when it calls a method. Siebel CRM displays the multi-value group applet in this window in Base mode or Edit List mode after the user saves or chooses the record.

For more information, see *About Applet Web Templates*.

Creating a Multi-Value Group

You use the Multi-Value Group Wizard to define a multi-value group. This wizard helps you define the objects that Siebel CRM requires for a multi-value group. For more information see *Multi-Value Group, Multi-Value Link, and Multi-Value Group Applet* and *How Siebel CRM Creates a Multi-Value Group*.

To create a multi-value group

1. In Siebel Tools, click the File menu, and then click New Object.
2. In the New Object Wizards dialog box, in the General Tab, click MVG, and then click OK.

3. In the Multi Value Group dialog box, choose the following:
    a. The project that the multi-value group references. Only locked projects are available.
    b. The master business component. The master business component must belong to the project you chose.
    c. Click Next.
4. In the Multi Value Group dialog box, do the following:
    a. Choose the detail business component.
    b. Enter a name for the multi-value link.
    c. Click Next.
5. Do one of the following:
    - In the Direct Links dialog box, choose the appropriate link, and then click Next.

      For more information, see *How Siebel CRM Creates a Direct Multi-Value Link*.
    - In the Indirect Links dialog box, choose the link and the source field in the master business component, and then click Next.

      For more information, see *How Siebel CRM Creates an Indirect Multi-Value Link*.

    The Multi-Value Group Wizard displays the Direct Links or Indirect Links dialog box depending on the choices you make in the Multi Value Group dialog box. The available links are the links that already exist between the master business component and the detail business component.
6. In the Primary ID Field dialog box, do the following:
    a. Choose the Primary ID Field in the master business component.

       For more information, see *Configuring the Auto Primary Property of a Multi-Value Link*.
    b. Set the value for the Auto Primary property.

       For more information, see *Configuring the Auto Primary Property of a Multi-Value Link*.
    c. Set the Use Primary Join property.

       For more information, see *Configuring the Use Primary Join Property of a Multi-Value Link*.
    d. Set the Check No Match property.

       For more information, see *Configuring the Check No Match Property of a Multi-Value Link*.
    e. Click Next.
7. In the Multi Value Link dialog box, choose the appropriate properties, and then click Next.
8. In the multi-value fields dialog box, enter information to create multi-value fields on the parent business component:
    a. Choose a field on the destination business component.
    b. Enter a name for the multi-value field.

       For more information, see *About the Multi-Value Field*.
    c. Click Add.
    d. Repeat the previous steps for each field you must add.
    e. Click Next.
9. In the Finish dialog box, review the information you entered for the multi-value group, and then click Finish.

Creating a Multi-Value Group Applet

You use the MVG Applet Wizard to create a multi-value group applet. This wizard helps you create the objects that Siebel CRM requires for a multi-value group applet. For more information see *Multi-Value Group, Multi-Value Link, and Multi-Value Group Applet* and *How Siebel CRM Creates a Multi-Value Group*.

To create a multi-value group applet

1. In Siebel Tools, click the File menu, and then click New Object.
2. In the New Object Wizards dialog box, click the Applets tab, click MVG Applet, and then click OK.
3. In the General dialog box, enter values using information from the following table, and then click Next.

Property	Description
Project	Choose the project to associate with this applet. Siebel Tools only includes locked projects in the list.
Applet Name	Apply the format for naming a multi-value group applet. For more information, see *Guidelines for Naming an Applet*.
Business Component	Choose the business component that this applet references.
Display Title	Enter the name that Siebel CRM displays in the Siebel client. For more information, see *Guidelines for Creating an Applet Title*.
Upgrade Behavior	Choose Admin.

4. In the Web Layout - General dialog box, enter the web templates to use for the applet, and then click Next.

   For more information, see *Including a New Button in a Multi-Value Group Applet*.
5. In the Web Layout - Fields dialog box, choose the fields that Siebel CRM must display in the applet, and then click Next.

   Siebel Tools displays the fields that are defined for the business component that you chose in Step 3. It displays these fields in the Available Fields window.
6. In the Web Layout - Fields dialog box, choose the controls in the Available Controls window that Siebel CRM must display in the applet, and then click Next.

   The wizard adds the controls that the Selected Controls window includes, by default. If you must exclude a control, then move it to the Available Controls window. For more information, see *Configuring How Siebel Tools Enters Data Into the Selected Controls Window*.
7. Review the information displayed in the Finish dialog box, and then click Finish.

   The MVG Applet Wizard creates the applet and the supporting object definitions. For more information about shuttle applets, see *About Shuttle Applets*.

Including a New Button in a Multi-Value Group Applet

You can include a New button in a multi-value group applet. For more information about how the applet mode affects a multi-value group applet, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*. For more information about templates, see *Siebel Developer's Reference* .

To include a New button in a multi-value group applet

- Do one of the following:
  - Manually define an Edit mode that uses the Popup Query template.
  - Set the Type property of the applet web template to New.

Using the Multi Value Group Wizard in Web Tools

Web Tools provides a wizard to create a new MVG. You must have an editable Workspace open in Web Tools to create a new MVG.

To create a new MVG using the wizard

1. Click the magic wand icon in the Web Tools toolbar.
2. Click **MVG**.
3. Provide a unique name for the MVG.
4. Choose a Project.
5. Choose the parent Business Component where the MVG will reside.
6. Choose the detail Business Component from which the records will originate.
7. Click Next.
8. Choose the Link between the parent and detail Business Components.

   > **Note:** Although there may be more than one Link shown, you may only choose one.

9. Click Next.
10. Choose the field in the parent Business Component that will hold the pointer to the primary detail record as the `Primary ID Field`.
11. Determine how the primary record should be handled by selecting either the `Default`, `Selected` or `None` radio button.
    a. **Default**: The first record becomes the primary.
    b. **Selected**: If the user views the multi-value group applet, and then exits, then the highlighted record becomes the primary.
    c. **None**: The user must use the pick map to include the Primary Owner Id manually.
12. **Optional**: Select the **Use Primary Join** check box. It is a recommended to use a primary join for improved performance. The `Use Primary Join` property of a multi-value link enables the primary join feature.

    If you set **Use Primary Join** to `TRUE`, then Siebel CRM gets the primary child record for each parent record through a join with the Primary ID Field. If you set **Use Primary Join** to `FALSE`, then Siebel CRM queries the child table each time it modifies a parent.
13. **Optional**: Select the **Check No Match** check box. When `Check No Match` is set to `TRUE`, a separate SQL query is executed for each parent record in which the primary join did not find a primary child record.

    Doing this ensures that the multi-value field does not appear blank unless there are no child records. But executing these extra SQL queries decreases performance.

Note: The default is False.

14. Click Next.
15. Choose the options for the Multi-value Link.
16. Choose which Fields you wish to create in the parent Business Component. If the Fields you choose already exist in the parent Business Component you cannot add them again.
17. Click **Next** to get a summary of your choices.
18. Click Finish.

About Association Applets

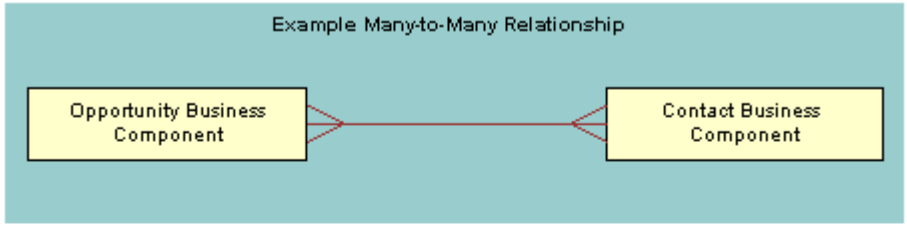
This topic describes association applets. It includes the following information:

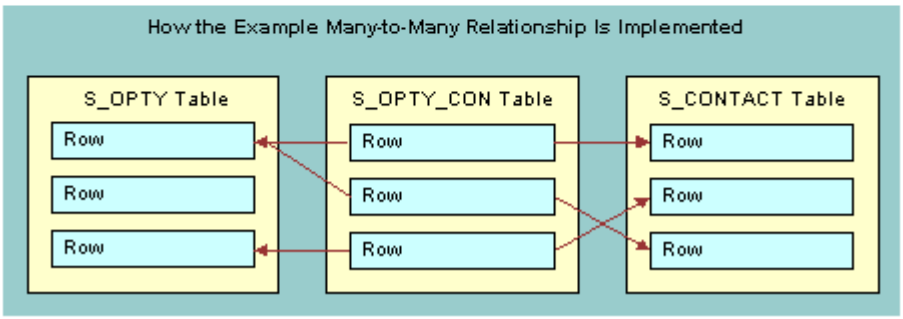
- *Overview of Association Applets*
- *How Siebel CRM Creates an Association Applet*
- *How Siebel CRM Calls an Association Applet from a Master-Detail View*
- *Constraining an Association Applet*

Overview of Association Applets

An *association applet* is a type of applet that allows the user to associate a parent record with one or more children. It uses two business components that possess a many-to-many relationship with one another. The user cannot modify records in an association applet. You can call an association applet from a master-detail view or from a multi-value group applet.

The following figure includes an example of how Siebel CRM implements a many-to-many relationship between two business components (the Opportunity and Contact business components) in the Siebel schema.





If a user adds a record to the child business component in a many-to-many relationship, then Siebel CRM associates the predefined detail record with a parent record rather than creating a new detail record. This is because parent and detail are relative terms in a many-to-many relationship. For example, Siebel CRM can display one opportunity to many contacts or one contact to many opportunities, depending on the view that is active.

In this situation, the association applet displays a list of available child records where the user can choose a detail record. The user can create a new detail record. In the context of this many-to-many relationship, Siebel CRM does the following:

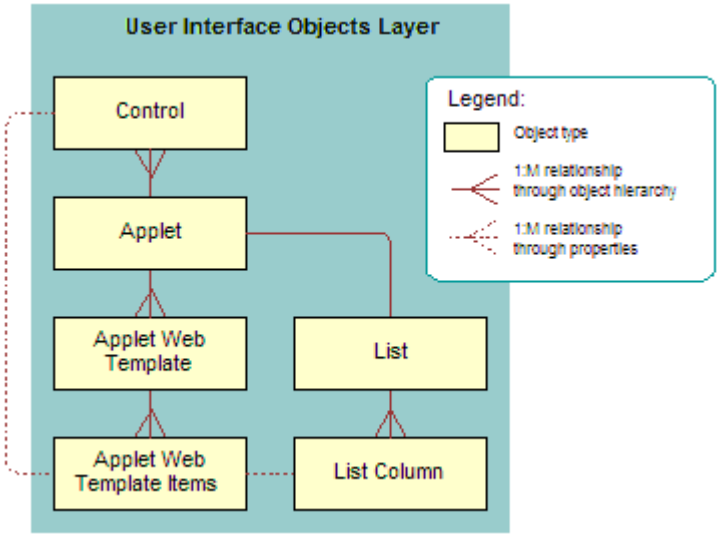
- If the user creates a new association for a predefined detail record, then it creates an association.
- If the user creates a new detail for an association, then it creates an addition.

Siebel CRM creates a new row in the intersection table for an association or an addition. It creates a new row in the detail table for an addition.

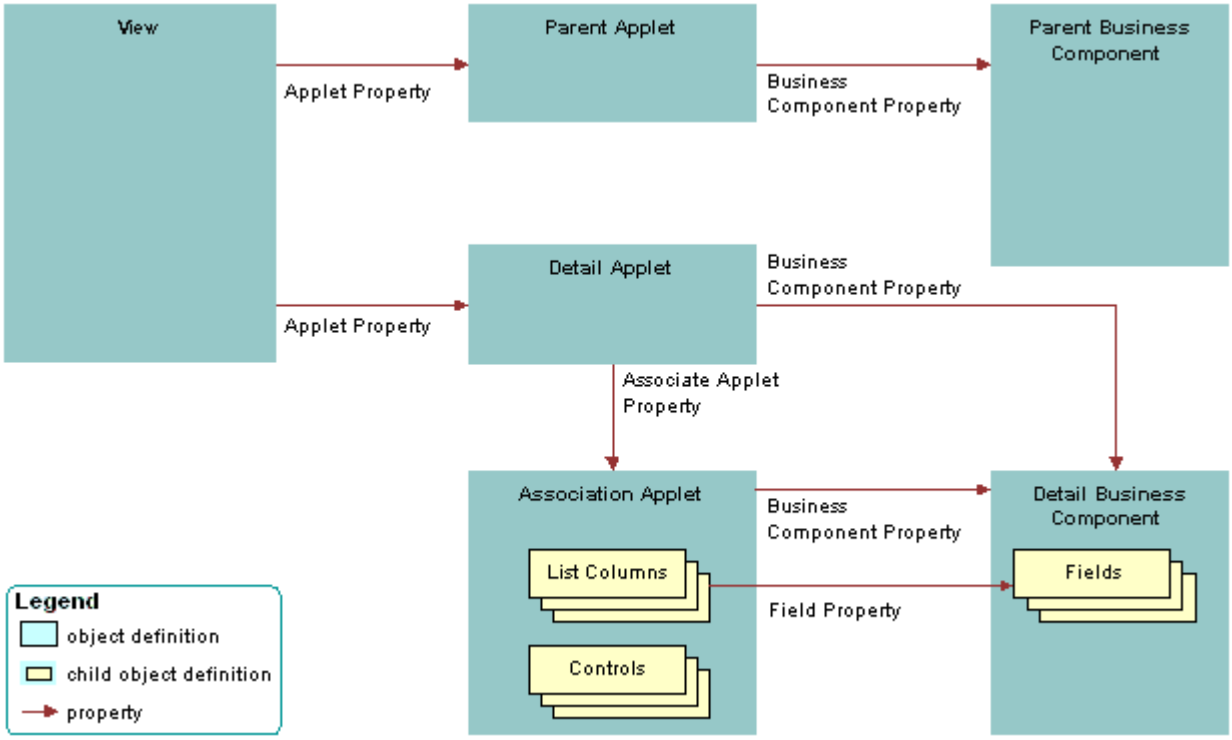
How Siebel CRM Creates an Association Applet

The following figure shows the relationships and objects that Siebel CRM uses to create an association applet, which are as follows.

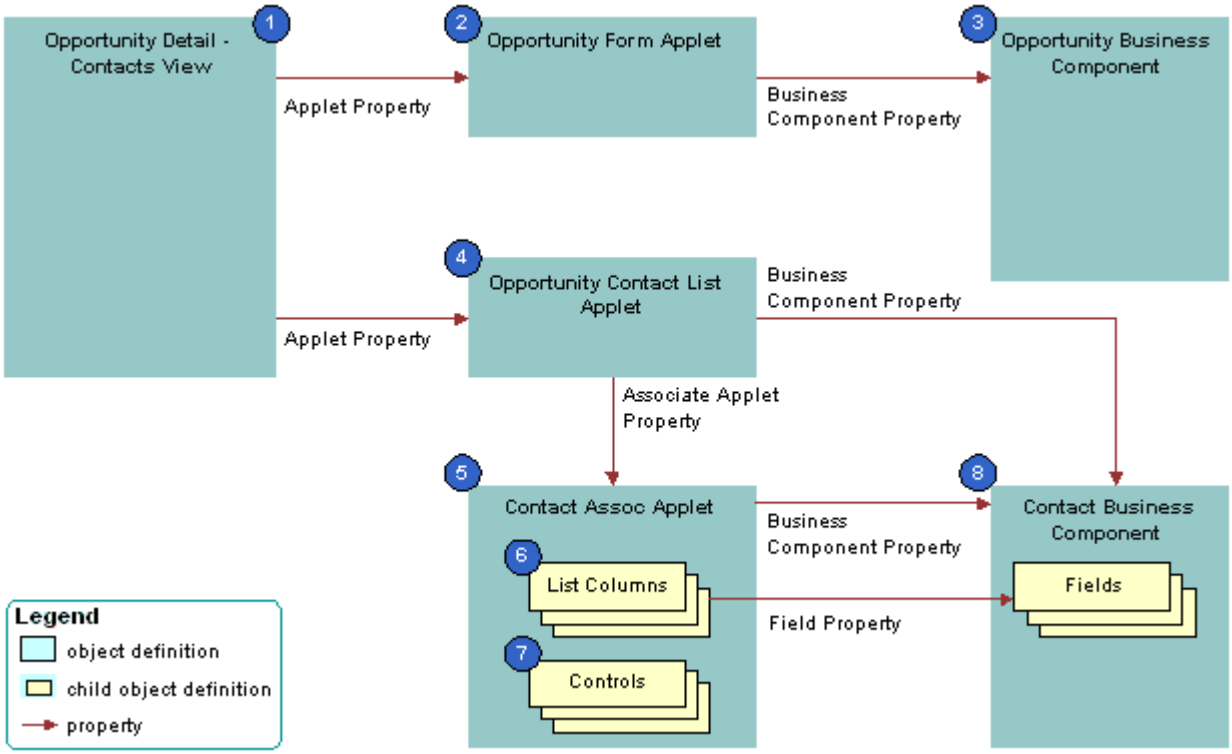
- **User Interface Objects Layer.** Contains the following objects: Control, Applet, Applet Web Template, Applet Web Template Items, List, List Columns. There is a 1:M relationship between Applet and Control, Applet and Applet Web Template, Applet Web Template and Applet Web Template Items, List and List Column. There is a 1:1 relationship between Applet and List.



The following figure describes a generic picture of how Siebel CRM creates an association applet.



The following figure includes an example of how Siebel CRM creates an association applet.



As shown in the previous two figures, Siebel CRM uses the following objects to create an association applet:

1. **Opportunity Detail - Contacts List View.** View that provides the context where Siebel CRM calls the association applet, but no properties of the view directly identify the association applet. The Business Object

property of the view creates the parent-child relationship between the business components whose data Siebel CRM displays.

2. **Opportunity form applet.** Parent applet that displays one record from the parent business component.
3. **Opportunity business component.** Parent business component that provides data for the parent applet.
4. **Opportunity Contact List Applet.** Detail applet that lists records from the child business component that are child records for the current parent record in the parent business component. Siebel CRM defines the name of the association applet in the Associate Applet property.
5. **Contact Assoc Applet.** Association applet that defines the dialog box that Siebel CRM displays if the user attempts to add or insert a record in the detail applet. It includes the following properties:

    - **Type property set to Association List.** Indicates that it is an association applet.
    - **Class property set to CSSFrameList.** Indicates that it is a list applet.

    Siebel CRM configures the association applet as a predefined list applet. This list applet includes a child List object. This child object includes List Object objects.
6. **List columns.** Defines the fields that Siebel CRM displays in the association applet, and in what order. They duplicate some or all of the list columns in the detail applet in the view.
7. **Controls.** For more information, see *Specialized Controls That Siebel CRM Can Display in an Association Applet*.
8. **Contact business component.** Detail business component that provides data for the detail applet and the association applet.

    Siebel CRM displays records from the child business component in the association applet. It only displays records in the detail applet that Siebel CRM already associates to the current parent record.

Specialized Controls That Siebel CRM Can Display in an Association Applet

The following table describes specialized controls Siebel CRM can display in an association applet. For more information, see *Caution About Using Specialized Classes*.

Control	Description
Cancel	Button that dismisses the dialog box.
Check	Button that associates chosen records to the current parent. Siebel CRM creates an intersection table row between the row identified in the parent applet and the row identified in the association applet. The control is named PopupQueryAdd and includes an AddRecord method that Siebel CRM calls.
Find	Combo box that allows the user to search for a record in the association applet.
Go	Button that the user clicks to start the search specified in the Find combo box and Starting With text box.
New	Button that creates a new row in the detail applet. Siebel CRM creates a new row in the detail table and an intersection table row between the row identified in the parent applet and the row created in the association applet. The control is named ButtonNew and it includes a NewRecord method that Siebel CRM calls.
Starting With	Text box where the user enters the search criteria. A wild card completes the criteria that the user enters in this control.

How Siebel CRM Calls an Association Applet from a Master-Detail View

Siebel CRM can call an association applet from a master-detail view where the underlying business components possess a many-to-many relationship. The association applet lists the records from the business component. The user can use the Find or Starting With control to choose one or more records, and then click OK to associate the chosen records with the parent record.

Viewing an Example of an Association Applet That Siebel CRM Calls from a Master-Detail View

You can view an example of an association applet that Siebel CRM calls in a master-detail view.

To view an example of an association applet that Siebel CRM calls from a master-detail view

1. In the Siebel client, click the Opportunities screen tab, and then click the Opportunities List link.
2. In the My Opportunities list, click a link in the Opportunity Name column.
3. In the Contacts list, click Menu, and then click New Record.

   Siebel CRM displays the Add Contacts dialog box. This dialog box is defined as the Contact Assoc Applet association applet.
4. Click the application menu, click Help, and then click About View.

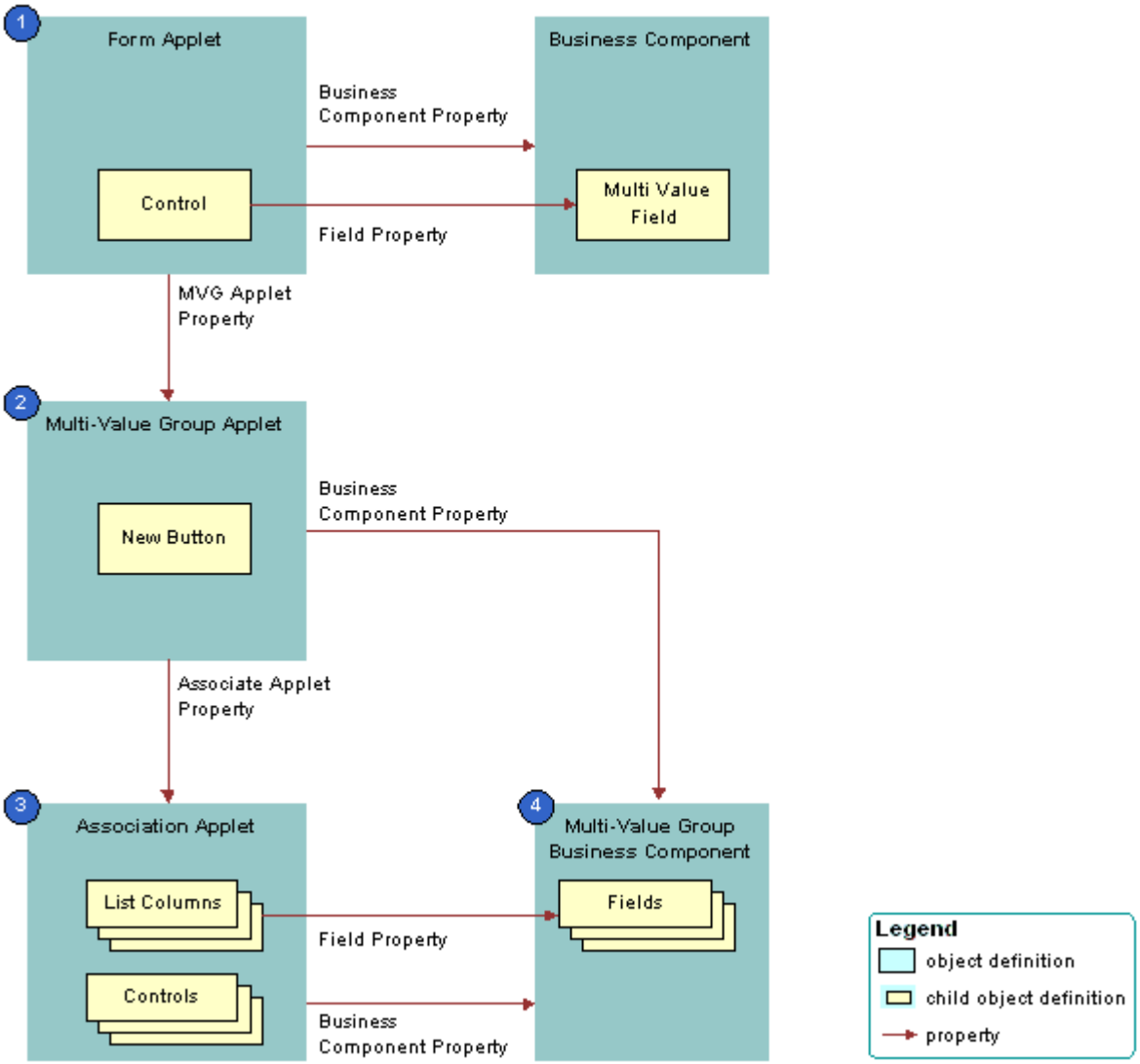
   Note that the Opportunity Detail - Contacts View is the master-detail view.
5. Click OK.
6. Click the Contacts screen tab, and then click the Contacts List link.
7. In the My Contacts list, click a link in the Last Name column.
8. Click the More Views down arrow, and then choose Opportunities.
9. Click the application menu, click Help, and then click About View.

   Note that the Contacts Detail - Opportunities View is a master-detail view that displays the inverse of the parent-child relationship you viewed in Step 4.
10. Click OK.
11. In the Opportunities list, click Menu, and then click New Record.

    If you click New Record, then Siebel CRM displays the Add Opportunities dialog box that allows you to choose an existing opportunity record and insert it, or to create a new opportunity record. If you click New, then Siebel CRM creates a new opportunity and allows you to enter data for the new record in the Opportunities list.

How Siebel CRM Creates an Association Applet It Calls from a Multi-Value Group Applet

The following figure shows how Siebel CRM creates an association applet that it calls from a multi-value group applet.



As shown in this figure, Siebel CRM uses the following objects to create an association applet that it calls from a multi-value group applet:

1. **Form applet.** Contains one or more text box controls that display a multi-value field. If the user clicks the MVG button, then the MVG Applet property of each of these text box controls identifies the multi-value group applet that Siebel CRM calls.
2. **Multi-value group applet.** Includes the list of records that Siebel CRM assigns to the multi-value field in the form applet. The Associate Applet property in the multi-value group applet identifies the association applet that Siebel CRM calls.
3. **Association applet.** Includes the list of records that are available to associate to the parent record. The association applet includes the following properties:
    - **Type property set to Association List.** Indicates the applet is an association applet.
    - **Class property set to CSSFrameList.** Indicates the applet is a list applet. The association applet is configured as a predefined list applet, with a List child object that includes List Object child objects.
4. **Multi-value group business component.** Stores the detail multi-value group records for each parent business component record. The multi-value group business component supplies records to the multi-value group applet and the association applet.

Constraining an Association Applet

You can use the Constrain property of a list to constrain a pick applet, but you cannot use the Constrain property to constrain or filter an association applet.

To constrain an association applet

- Use Siebel Visual Basic or Siebel eScript to create a query with the Exists clause in the WebApplet_Load event on the association applet.

About Shuttle Applets

A *shuttle* applet is a type of applet that allows the user to associate child records with a parent record and to create new records. Siebel CRM displays a shuttle applet in the following situation:

1. The user clicks the MVG button.
2. The business component of the underlying multi-value group applet includes a many-to-many relationship with the parent business component.

A shuttle applet uses the same underlying object architecture as an association applet. For more information, see *How Siebel CRM Creates an Association Applet It Calls from a Multi-Value Group Applet*.

A shuttle applet gets the following items from the association applet:

- Applet header. For example, New, Query, Find, and Starting With.
- *Available* label.
- List body that Siebel CRM displays before the shuttle applet.

A shuttle applet gets the following items from the multi-value group applet:

- Selected label.
- List body that Siebel CRM displays after the shuttle applet.
- OK button.
- Add, Add All, Remove, and Remove All buttons.

You cannot call a popup applet from a shuttle applet.

How the Shuttle Applet Uses Web Templates

Siebel CRM uses the following specialized web templates to display a shuttle applet:

- CCPopupListAssocShuttleButtonsTop
- CCPopupListMvgShuttleButtonsTop

For more information, see *Caution About Using Specialized Classes*.

The Mode property of the applet web template item identifies the applets where Siebel CRM displays the controls:

- If Mode is not defined, then Siebel CRM displays the control in shuttle and nonshuttle applets.

- If Mode is DefaultOnly, then Siebel CRM displays the control only in an applet that is not a shuttle applet. For example, it might display the OK and the Cancel button on the association applet.
- If Mode is More, then Siebel CRM displays the control only in the shuttle applet. For example, it might display Add, Add All, Remove, or Remove All.

For more information, see *Properties of the Applet Web Template Item*.

Viewing an Example of a Shuttle Applet

You can view an example of a shuttle applet.

To view an example of a shuttle applet

1. In the Siebel client, click the Contacts screen tab, and then the Contacts List link.
2. Click a link in the Last Name column.
3. In the form, click the MVG button for the Account field.

   Siebel CRM displays the Accounts shuttle applet.

Creating a Shuttle Applet

To create a shuttle applet, you use a multi-value group applet and an association applet in a view. This example adds employees to a sales team.

To create a shuttle applet, perform the following tasks:

1. *Creating an Association Applet*
2. *Creating the Multi-Value Group Applet*
3. *Creating the View*

Creating an Association Applet

This task is a step in *Creating a Shuttle Applet*.

Siebel CRM displays the association applet before the view in a shuttle applet. It contains the list of records that are available.

To create an association applet

1. In Siebel Tools, click the File menu, and then click New Object.
2. Click the Applets tab, click MVG Applet, and then click OK.
3. In the General dialog box, define properties table.

Property	Description
Project	Choose the locked project where you must create the association applet.

Property	Description
Applet Name	Enter `Create Contact Access List Assoc`. For more information, see *Guidelines for Naming an Applet*.
Display Title	Enter `All Employees`. For more information, see *Guidelines for Creating an Applet Title*.
Business Component	Choose Employee.
Upgrade Behavior	Choose Admin.

4. Click Next.
5. For the Edit List mode, choose Popup List Assoc, and then click Next.

   For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.
6. In the first Web Layout - Fields dialog box, choose the following fields, and then click Next:
   - First Name
   - Last Name
7. In the second Web Layout - Fields dialog box, remove Query Assistant from the list of controls that Siebel Tools displays in the Selected Controls window.
8. Click Next, and then click Finish to create the applet.
9. In the Object Explorer, click Applet.
10. In the Applets list, locate the Create Contact Access List Assoc applet, and then modify properties using values in the following table.

Property	Value
Class	CSSSWEFrameShuttleBaseAssoc
Type	Association List

11. In the Object Explorer, expand the Applet tree, and then click Applet User Prop.
12. In the Applet User Props list, add new records using values from the following table.

Name	Value
CanInvokeMethod: AddRecords	[Active]
EnableStandardMethods	Y

Name	Value

13. Save your modifications.

Creating the Multi-Value Group Applet

This task is a step in *Creating a Shuttle Applet*.

In a shuttle applet, Siebel CRM displays the multi-value group applet after the view. It contains the list of *chosen* records.

To create the multi-value group applet

1. In Siebel Tools, click the File menu, and then click New Object.
2. Click the Applets tab, click MVG Applet, and then click OK.
3. In the General dialog box, define properties using values from the following table, and then click Next.

Property	Description
Project	Choose the locked project where you created the association applet
Applet Name	Enter **`Create Contact Access List MVG`**. For more information, see *Guidelines for Naming an Applet*.
Display Title	Enter **`Team Members`**. For more information, see *Guidelines for Creating an Applet Title*.
Business Component	Choose Contact.
Upgrade Behavior	Choose Admin.

4. In the Web Layout - General dialog box, choose Popup List Mvg for the Edit List mode, and then click Next.
   For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.
5. On the Web Layout - Fields page, choose the following fields, and then click Next:
   - SSA Primary Field
   - First Name
   - Last Name
6. In the second Web Layout - Fields dialog box, remove Query Assistant from the list of chosen controls.
7. Click Next, and then click Finish to create the applet.
8. In the Object Explorer, click Applet.

9. In the Applets list, locate the Create Contact Access List MVG applet, and then modify properties using values in the following table.

Property	Value
Class	CSSSWEFrameShuttleBaseMvg
Associate Applet	Create Contact Access List Assoc

10. In the Object Explorer, expand the Applet tree, and then click Applet User Prop.
11. In the Applet User Props list, add new records using values from the following table.

Name	Value
CanInvokeMethod: AddRecords	[Active]
CanInvokeMethod: DeleteAllRecords	[Active]
CanInvokeMethod: DeleteRecords	[Active]
EnableStandardMethods	Y

12. Relocate controls from the Controls/Columns window to the applet until your layout resembles the layout displayed in the following diagram:



    a. Drop the AddRecord, RemoveRecord, and RemoveAllRecords controls.
    b. Drop the PositionOnRow control before SSA Primary Field.
13. Save your modifications.

Creating the View

This task is a step in *Creating a Shuttle Applet*.

In this optional step, you define the view that contains the multi-value group applet and the association applet.

To create the view

1. In Siebel Tools, click the File menu, and then click New Object.
2. Choose View in the General tab, and then click OK.
3. Define properties in the New View dialog box using values from the following table.

Property	Description
Project	Choose the locked project where you created the association applet
View Name	Enter **`ABC Contact Team View`**.
View Title	Enter **`ABC Contact Team View`**.
Business Object	Choose Contact.
Upgrade Behavior	Choose Admin.

4. Click Next.
5. In the View Web Layout-Select Template dialog box, choose the following value, and then click Next:

   `View 1 Over 2 Over 1`

6. In the Web Layout-Applets dialog box, choose the following applets, and then click Next:
   - Create Contact Access List Assoc
   - Create Contact Access List MVG
7. Click Finish to create the view.

   Siebel Tools creates the new view, and then displays it in the Web Layout Editor.
8. Close the Web Layout Editor.
9. In the Object Explorer, click View.
10. In the Views list, locate ABC Contact Team View.
11. In the Object Explorer, expand the View tree, and then click View User Prop.
12. In the View User Props list, add three new records using values from the following table.

Name	Value
ShuttleViewMvgAppletName	Create Contact Access List MVG
ShuttleViewMvgField	Sales Rep
ShuttleViewParentBuscomp	Contact

13. Open the view in the Web Layout Editor, and then modify the layout until it is similar to the layout displayed in the following diagram:



To position an applet in the view, click the applet, and, keeping the mouse button depressed, move it to one of the empty side-by-side placeholders and then release the mouse button.

14. Test and then deliver your Workspace.

For more information, see *Using Siebel Tools* .

20 Configuring Menus, Toolbars, and Icons

Configuring Menus, Toolbars, and Icons

This chapter describes how to configure menus, toolbars, and icons. It includes the following topics:

- *About Menus and Toolbars*
- *Configuring Menus and Toolbars*
- *Configuring Icons*

For more information, see *Localizing an Application Menu*.

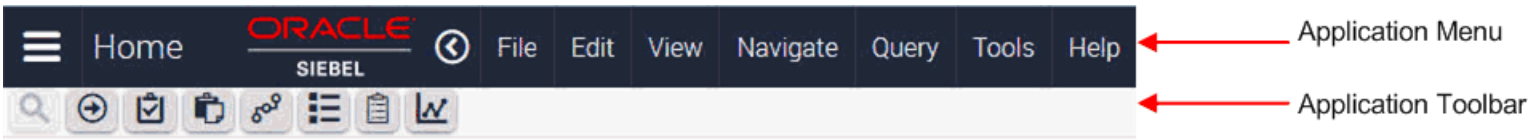
About Menus and Toolbars

This topic describes menus and toolbars. It includes the following information:

- *Objects That Siebel CRM Uses in a Menu or Toolbar*
- *About the Method, Business Service, and Target Properties of the Command Object*

Menus and *toolbars* are user interface elements that allow the user to do something. For example:

- The *application menu*, highlighted in the following image, is a menu that allows the user to perform a task consistently across a Siebel application. Siebel CRM displays it in a frame near the title bar of the Siebel client in the browser window. This menu includes submenus, such as File, Edit, View, Navigate, Query, Tools, and Help.
- The *application toolbar*, highlighted in the following image, is a toolbar that allows the user to access commonly performed tasks. Siebel CRM displays it just beneath the primary tab bar, as illustrated in the following figure. Some icons on the application toolbar are redundant with menu items that the application menu contains.



Applet Menu

An applet can contain a menu. An *applet menu* is a contextual menu that includes a number of menu items. Each menu item that an applet menu contains allows the user to perform a task in the context of the applet. Siebel CRM locates the applet menu in the applet. To view an example applet menu, you can open the Siebel client, click the Accounts screen tab, and then the Accounts List link. In the Accounts List applet, click Menu. The menu that Siebel CRM displays is an example of an applet menu. If you click it, then Siebel CRM displays a pop-up contextual menu.

How Siebel CRM Handles a Menu or Toolbar Action

If the user clicks a menu item or toolbar icon, then Siebel CRM calls a method. This method can exist in one of the following ways:

- In a service that resides on the browser or server
- In a class that resides in the browser application, such as an applet or business component class
- In a class that resides in the server infrastructure, such as a Siebel Web Engine frame manager

Siebel CRM defines the menu item or toolbar icon to target the following items:

- A method name
- A method handler
- A service (optional)

A web template involves toolbar tags. For more information, see *Using Web Templates to Configure Toolbars* and *Siebel Developer's Reference* .

Objects That Siebel CRM Uses in a Menu or Toolbar

This topic describes the objects that Siebel CRM uses in a menu or toolbar.

Command Object Type

A *command* is an object that specifies the method that Siebel CRM calls when the user chooses an application menu item or an applet menu item, or clicks a toolbar icon. It specifies the bitmap that Siebel CRM displays for a toolbar icon. A menu item or toolbar item references a command. An applet menu does not reference a command.

For more information, see *Creating a Command Object* and *Properties of a Command*.

Toolbar Object Type

A *toolbar* is an object that provides a named toolbar that the user can activate or deactivate. You can associate or remove a toolbar item object definition. A toolbar object must exist for each toolbar that Siebel CRM displays.

For more information, see *Properties of a Toolbar*.

HTML and Java Usage

An *HTML toolbar* is a type of toolbar that typically defines toolbar functionality for Siebel CRM. Each button Siebel CRM displays in an HTML toolbar is a static image that you can dim to indicate that the button is not available. Program logic on the browser does not manipulate an HTML toolbar.

A *communication toolbar* is a type of toolbar that Siebel CRM displays that you can modify in reply to an event. For example, Siebel Call Center displays a blinking icon on a communication toolbar to indicate an incoming telephone call. A communication toolbar uses Java. You must enter a class name in the Class property for a toolbar that uses Java. For more information, see *Siebel CTI Administration Guide* .

Menu and Menu Item Object Types

A *menu* is an object that defines a named menu that Siebel CRM displays in the Siebel client. You can add or remove menu items for each menu.

A *menu item* is an object that associates a command object definition with a menu item object definition. This association places a menu item in a position. The command object definition that the menu contains defines the method for this menu item. For more information, see *Properties of a Toolbar Item*.

Toolbar Item Object Type

A *toolbar item* is an object that associates a command with a toolbar. This association places a toolbar icon on the toolbar in a location relative to the other toolbar icons that Siebel CRM displays on this toolbar. The toolbar object is the parent of the toolbar item. For more information, see *Properties of a Toolbar Item*.

Applet Method Menu Item Object Type

An *applet method menu item* is an object that is a child of an applet. It defines a menu item that resides in the applet menu for the parent applet. For more information, see *Properties of an Applet Method Menu Item*.

Class Method Menu Item Object Type

A *class method menu item* is an object that is a child of a class. It adds or suppresses a menu item that Siebel CRM displays on an applet menu for Siebel Web Engine applets of the defined applet class and subclasses. For more information, see *Properties of an Applet Method Menu Item*.

About the Method, Business Service, and Target Properties of the Command Object

You can use the Method, Business Service, and Target properties of a command for application menus, applet menus, or toolbars. The target property specifies the object or service that processes the method that the command calls.

How Siebel CRM Redirects a Method

In some situations, if the target cannot handle a method, then Siebel CRM redirects the method to an underlying object or service. This object or service can be one of the following:

- A mirror instance of the object. This instance exists on the Siebel Server.
- An inherited class.

In these situations, Siebel CRM redirects the method.

Options for the Business Service Property

If the Business Service property identifies a business service, then this business service handles the method depending on the following situations:

- **Siebel CRM receives the call from an application menu or application toolbar.** Siebel CRM uses the object manager service to handle the method. It does not retarget the call.
- **Siebel CRM receives the call from an applet menu.** The method handler does a SetBC call to set to the business component that the applet reference, and then calls the object manager service. It does not retarget the call.

Options for the Target Property

This topic describes the options that are available for the Target property. If you use the Object List Editor, then the Target property for the Command object displays six values. If you use the Command Wizard to create a new Command object, then Browser and Server are the only values available of these six values.

Target Property Set to Browser

If you set the Target Property to Browser, then the following situation applies:

- Siebel CRM does not run the server PreInvokeMethod.
- The method handler for this target exists on the browser as the JavaScript application, a JavaScript applet, or a JavaScript service.
- You must define a method name in the Method property.
- If the Business Service property specifies a business service, then Siebel CRM targets this business service.
- If the Business Service property does not specify a business service, then Siebel CRM does one on the following:
  - **Siebel CRM receives the call from an application menu or application toolbar.** Siebel CRM targets to the method that the JavaScript application defines. For example, if you configure Siebel CRM to use the ActiveBusObject and RaiseErrorText application methods in a server script, then these methods must include a Browser target.
  - **Siebel CRM receives the call from an applet menu.** Siebel CRM targets to the method that the JavaScript applet defines. If Siebel CRM cannot use this configuration to handle the call, then it retargets the call to the method that it defines in the corresponding JavaScript business component. It does no inheritance or retargeting.

Target Property Set to Server

If you set the Target Property to Server, then Siebel CRM does not run the browser PreInvokeMethod, and Siebel CRM calls a method in a C++ class that resides on the Siebel Server on a service or on the infrastructure. If the Service property is not defined, then Siebel CRM targets the method to the infrastructure. This targeting depends on the following:

- **Siebel CRM receives the call from an application menu or toolbar.** Siebel CRM handles the method in the following order of priority:
  - Uses the Siebel Web Engine UDF loader on the Siebel Server
  - Uses the model
- **Siebel CRM receives the call from applet menu.** Siebel CRM handles the method in the following order of priority:
  - Uses the applet class that the applet references
  - Retargets, if necessary, successively up through the applet class hierarchy to CSSSWEFrame
  - If still not handled, retargets to the business component class of the business component that the applet references, and successively up through the business component class hierarchy to CSSBusComp

Summary of the Target and Business Service Properties

The following table summarizes the Target and Business Service properties.

Menu or Toolbar	Target Property	Business Service Property	Result
Application menu or toolbar	Server	Contains a value	The business service that the Business Service property defines determines the method handler that calls the service on the Siebel Server. It does not retarget.
		Does not contain a value	Siebel CRM uses the method handler as the base functionality that it associates with an application object.
	Browser	Contains a value	The business service that the Business Service property defines determines the targets that Siebel CRM uses for the method. It does not retarget.
		Does not contain a value	Siebel CRM targets to the method that the JavaScript application defines. It does not retarget.
Applet menu	Server	Contains a value	The business service that the Business Service Property defines determines the service that the method handler calls on the Siebel Server. It does not retarget.
		Does not contain a value	The method handler is initially the applet class that the applet references. Siebel CRM retargets it successively up through the applet class hierarchy to the CSSSWEFrame class. If still not handled, then it retargets to the business component class of the business component that the applet references, and successively upwards through the business component class hierarchy to the CSSBusComp class.
	Browser	Contains a value	The business service that the Business Service Property defines determines the service that the method handler calls on the browser. It does not retarget.
		Does not contain a value	Targets to the method that the JavaScript applet defines. If not handled, then retargets to the method that the corresponding JavaScript business component defines. No inheritance or more retargeting occurs.

Configuring Menus and Toolbars

This topic describes how to configure menus and toolbars. It includes the following information:

- *Creating a Command Object*
- *Creating a New Toolbar*
- *Adding a New Toolbar Icon to a Predefined Toolbar*
- *Activating Menu Items and Toolbars*
- *Creating an Applet Menu*
- *Activating or Suppressing an Applet Menu Item*

- *Using JavaScript to Configure a Toolbar*

Creating a Command Object

This topic describes how to create a command object. For more information, see *About the Method, Business Service, and Target Properties of the Command Object*.

To create a command object

1. In Siebel Tools, click the File menu, and then click New Object.
2. Click the Command icon, and then click OK.
3. In the Command dialog box, do the following:
   - Enter the project.
   - Enter a unique name for the command object.
   - Choose the browser or the Siebel Server to handle the method that the command calls.
   - Click Next.
4. In the next dialog box, do the following:
   - Choose the object that handles the command. If a business service handles the command, then choose the business service from the list. You must know if the business service is available for your choice of browser or for the Siebel Server.
   - Enter the method that the command calls. You must choose a method that is available to the business service or Siebel application.
   - Optional. Provide the argument that Siebel CRM sends to the method. The argument must be correct for the chosen method.
   - Click Next.
5. In the Window dimensions dialog box, do the following:
   - Specify to run or not run the command in a new browser window. If Siebel CRM runs the command in a new browser window, then define the height and width for the window.
   - Optional. Define the HTML bitmap and the tooltip text that Siebel CRM displays on the toolbar button that is associated with the command.
   - Click Next.
6. In the Command dialog box, review your entries.

   If you must make any modifications, then click Back.
7. Click Finish.

Creating a New Toolbar

You can create a new toolbar for Siebel CRM.

To create a new toolbar

1. In Siebel Tools, display the Toolbar object type.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.

2. In the Object Explorer, click Toolbar.
3. In the Toolbars list, add a new record.
4. Define the name of the new toolbar in the Name property.
5. Add a tag to the Container Page or to one of the child templates that you are using so that Siebel CRM can display the toolbar in the Siebel client.
   For more information, see *About the Container Page* and *Siebel Developer's Reference* .

Adding a New Toolbar Icon to a Predefined Toolbar

You can add a new toolbar icon to a predefined toolbar.

To add a new toolbar icon to a predefined toolbar

1. In Siebel Tools, display the Toolbar object type and all child object types of the Toolbar object type.
   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. Verify that the bitmap image you must use for the toolbar icon currently exists as a child bitmap of the Command Icons bitmap category.
   If it does not exist, then create a bitmap in this bitmap category. For more information, see *Overview of Configuring Icons That Siebel CRM Displays in Siebel Clients*. If it does exist, then note the name of the bitmap.
3. Verify that the method that this toolbar icon calls currently exists.
4. If the method that this toolbar icon calls does not exist, then do the following:
   a. Add a Siebel Visual Basic or Siebel eScript script to the PreInvokeMethod.
   b. Write an If or Case statement in the script that references MethodName. Write the instructions for that MethodName in the If or Case statement.
   c. Modify the last line of PreInvokeMethod from ContinueOperation to CancelOperation.
5. Create a new command object:
   a. In the Object Explorer, click Command.
   b. In the Commands list, add a new command.
   c. Define the required properties.
      For more information, see *Properties of a Command*.
6. In the Object Explorer, click Toolbar.
7. In the Toolbars list, locate the toolbar where Siebel CRM must add the new toolbar item.
8. In the Object Explorer, expand the Toolbar tree, and then click Toolbar Item.
9. In the Toolbar Items list, add a new toolbar item, and then define the required properties.
   You must use a button. You cannot use other types of elements, such as a combo box or label. For more information, see *Properties of a Toolbar Item*.

Activating Menu Items and Toolbars

Siebel CRM calls CanInvokeMethod for each item before it displays the menu or toolbar. If CanInvokeMethod returns FALSE, then Siebel CRM does the following:

1. Does not display the menu item or toolbar item.

2. Retargets CanInvokeMethod from the browser application to the applet class hierarchy that resides on the Siebel Server, and then to the business component class hierarchy.

For more information, see *About the Method, Business Service, and Target Properties of the Command Object*.

To activate menu items and toolbars

- Use CanInvokeMethod to activate or deactivate the menu items that Siebel CRM displays in an application menu or applet menu, or the toolbar items that it displays in a toolbar in the Siebel client.

Creating an Applet Menu

You can modify an applet menu that comes predefined with Siebel CRM. You can create a custom applet menu. The Applet Method Menu Wizard allows you to modify an applet method menu. To create an applet method menu, Siebel CRM uses the menu items from the class where the applet belongs and the super class of the applet. It explicitly creates menu items for the applet. You can use the wizard to do the following:

- Suppress inherited method menu items.
- Resurrect inherited method menu items.
- Create a new method menu item for an applet.
- Delete a predefined method menu item of an applet.

To use the Applet Method Menu Wizard

1. In Siebel Tools, click the File menu, and then click New Object.
2. In the New Object Wizards dialog box, in the General Tab, click Applet Method Menu, and then click OK.
3. In the Applet Method Menu dialog box, do the following:
   a. In the project window, choose the project that is defined in the Project property of the applet.
   b. In the applet name window, choose the applet you must modify, and then click Next.
4. In the Applet Method Menu dialog box, do one of the following:
   - To display a menu item, move the item to the Selected Menu Items window.
   - To suppress display of a menu item, move it out of the Selected Menu Items Window.
5. Do one of the following:
   - Click Finish.
     If you click Finish, then Siebel Tools saves all the modifications that you made to the Siebel repository, displays the object definition for the applet in the Applets list, and then exits this procedure.
   - Choose Create New Menu Item, and then click Next.
     If you choose Create New Menu Item, then Siebel Tools replaces the Finish button with the Next button.
6. To create a new object definition for a method menu item, choose an entry from the Select the Command to be Executed by This Menu Item window.
7. In the Enter the Text to be Displayed for This Menu Item window, define the text to display for this method menu item, and then click Next.
   Siebel Tools displays the Method Menu Item dialog box. You can examine the properties that you defined. Click Back to return to the appropriate dialog box to make a correction.
8. Click Create Menu Item to create the method menu item.

Siebel Tools creates the item.

9. Click Next.

   Siebel Tools displays the method menu item you just defined in the Selected Menu Items window of the Applet Method Menu dialog box.

10. Click Finish.

    Siebel Tools displays the Applet Layout.

To use the Applet Method Menu in Web Tools

1. Open your Workspace. To use Workspaces, see *Using Siebel Tools* .
2. Navigate to Settings and then to Object Explorer.
3. Select Applet and then click the Applet Method Menu Item.
4. In the Applets applet, query on required project.
5. Select the required applet from the search result.
6. In the Applet Method Menu Items applet, view details of selected applet.
7. To perform other functions on the Applet or its method item, use the applet menu in respective applet.

Activating or Suppressing an Applet Menu Item

You can modify an applet menu item that comes predefined with Siebel CRM. You can define a custom applet menu item. For an example, see the topic about defining a menu item to start a task UI in *Siebel Business Process Framework: Task UI Guide* .

You can activate or suppress individual applet menu items. You can use the configurations that this topic describes only for applet menus. You cannot use these configurations for application menus or for toolbars.

Siebel CRM includes some applet menu items in almost all applets, such as Copy, Edit, and Delete. It includes other applet menu items in almost all list applets, such as Columns Displayed. You can activate an applet menu to make a menu item available globally for applets of a class and subclass. You can then suppress it in applets where Siebel CRM must not display the menu item.

> **CAUTION:** You cannot include a browser script in a business service that Siebel CRM calls from an applet menu item. The business service works only with a server script. If Siebel CRM runs a business service that includes a browser script from an applet menu item that resides on the Siebel Server, then the business service fails.

To activate or suppress an applet menu item

- Do one of the following:
  - Set the Suppress Menu Item property in the class method menu item.
  - Use an applet method menu item.

Adding an Applet Menu Item

You can add a class method menu item for a predefined menu item for an applet class, but Siebel CRM does not include this menu item as an applet method menu item in an applet where the menu item must display. You can create an applet method menu item only in the following situations:

- To add a menu item to the applet that the applet class does not already provide.
- To suppress display of an applet menu item that the applet normally inherits. In this situation, you can do the following:
  - Create an applet method menu item object definition with the same name as the applet menu item you must suppress.
  - Make sure the Suppress Menu Item property contains a check mark.

Using JavaScript to Configure a Toolbar

You can configure a JavaScript toolbar or create a new JavaScript toolbar.

Note: To use scripts, go to Siebel Tools

To use JavaScript to configure a toolbar

1. Create a JavaScript file.

   You use this file to define a custom JavaScript toolbar class that is a subclass of JSSToolbar.
2. Copy the JavaScript file to the following directory on the Siebel Application Interface:

   ```
   SIEBEL_AI_ROOT/applicationcontainer_external/siebelwebroot/scripts
   ```
3. In Siebel Tools, display the DLL object type and the Class object type.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
4. Create a DLL object:
   a. In the Object Explorer, click DLL.
   b. In the DLLs list, add a new record using values from the following table.

Property	Value
Name	Enter a name for the DLL object. For example, **`BarcodeToolbar`**.
Project	Choose a project that is currently locked in the Siebel Repository.
File Name	Enter the file name that references the JavaScript file. For example, **`barcodeToolbar.js`**.

5. Create a Class object:
   a. In the Object Explorer, click Class.
   b. In the Classes list, add a new record using values from the following table.

Property	Value
Name	Enter the name of the class that is defined in the JavaScript file. For example, `JSSBarcodeToolbar`.
Project	Choose the project that you defined in Step 3.
DLL	Choose the name of the DLL object that you defined in Step 3.
High Interactivity Enabled	1

6. If you create a new toolbar, then create a Toolbar object.

   Make sure you set the Class property to the class defined in the JavaScript file. For example, JSSBarcodeToolbar. For more information, see *Creating a New Toolbar*.
7. Add new toolbar items.

   For more information, see *Adding a New Toolbar Icon to a Predefined Toolbar*.
8. If you create a new toolbar, then add a tag to the appropriate web template.

   Make sure you set the name property that you define in the tag to the name of the Toolbar object that you created in Step 6. For more information, see *Using Web Templates to Configure Toolbars*.
9. Add tags to the tag.

   For more information, see *Using Web Templates to Configure Toolbars*.

Configuring Icons

This topic describes how to configure icons that Siebel CRM displays in the Siebel client. It includes the following information:

- *Overview of Configuring Icons That Siebel CRM Displays in Siebel Clients*
- *Configuring a Bitmap Category and a Bitmap*
- *Displaying an Icon on a Button*
- *Displaying an Icon as a Link*
- *Using Icons to Represent Values in a Field*
- *Configuring Icons in a Tree Applet*

Overview of Configuring Icons That Siebel CRM Displays in Siebel Clients

The following table describes the object types that Siebel CRM uses to display images in the Siebel client.

Object Type	Description
Bitmap	Allows you to associate an image file, such as a GIF file or JPEG file, with a Siebel object, such as a button control or field. It fulfills the following roles: • Specifies an image that resides in the Siebel repository. This image can be in any format that the browser supports. • Specifies the location of the image file and other properties, such as width and height. A bitmap includes the following properties that Siebel CRM commonly uses: • **Height and Width.** You can set the height and width of the image that Siebel CRM displays on the web page. If you set these properties, then the Siebel Web Engine uses them for the width and height properties of the `img` tag. This configuration allows you to create bitmap objects that share the same image file but that Siebel CRM displays with different dimensions. • **Alt Text.** You can include text in the alt attribute of the image tag. You do not use the other properties of the bitmap with a web image. Example properties include *Data* and *Transparent Color*.
Bitmap Category	Allows you to group image files together according to function. Includes the following bitmap categories: • **Button Icons.** Contains images that Siebel CRM uses to display buttons in applets in the Siebel client. • **HTML Control Icons.** Contains images that Siebel CRM uses to display HTML controls in the Siebel client.
HTML Hierarchy Bitmap	Allows you to configure Siebel CRM to display an image in a hierarchical applet, such as a tree applet.
Icon Map	Allows you to configure Siebel CRM to display an image for a field value. Includes the child Icon object type.

How Siebel CRM Handles Image Files

Siebel CRM handles image files differently, depending on the file type:

- Imports BMP images into the Siebel repository. Sets the File Name field of the bitmap to read-only.
- Stores GIF and JPG files in the `SIEBEL_AI_ROOT\applicationcontainer_external\siebelwebroot\images` directory of your Siebel Application Interface installation. The bitmap references these files. Does not store GIF and JPG files in the Siebel repository.
- Stores GIF and JPG files in the `public\images` folder of your Siebel Tools or Siebel Web Client installation. The bitmap references these files. Does not store GIF and JPG files in the Siebel repository.

Siebel CRM only defines images that it associates with Siebel objects as bitmap objects in the Siebel repository. Example objects include icon maps, page tabs, and so on. It does not associate some images in web templates, such as static images, with Siebel objects. It does not define these images as bitmap objects in the Siebel repository.

The Siebel Web Engine (SWE) uses the HTML img tag to display a bitmap.

Displaying Object Types You Use to Configure Icons

You must display the object types that you use to configure icons in the Siebel client.

To display object types you use to configure icons

- Display the following object types:
  - Bitmap Category
  - Child objects of the Bitmap Category
  - Icon Map
  - Child objects of the Icon Map
  - HTML Hierarchy Bitmap

For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.

Configuring a Bitmap Category and a Bitmap

You can configure a bitmap category and a bitmap.

To configure a bitmap category and a bitmap

1. Display object types you use to configure icons.

   For more information, see *Displaying Object Types You Use to Configure Icons*.
2. In Siebel Tools, in the Object Explorer, click Bitmap Category.
3. In the Bitmap Categories list, create a new bitmap category or choose a predefined bitmap category.
4. In the Object Explorer, expand the Bitmap Category tree, and then click Bitmap.
5. In the Bitmaps list, create a new bitmap using values from the following table.

Property	Description
Name	Enter the name of the bitmap.
Alt Text	Enter alternative text that Siebel CRM uses in place of the name property for a bitmap.
File Name	Do one of the following: o To create a bitmap for a BMP file, leave the File Name property empty.

Property	Description
	○ To create a bitmap for a GIF file, enter the name of the image file in the File Name property. If the image resides in a subfolder of the image folder, then include this subfolder. For example, for an image named image.gif: - That resides in the `SIEBEL_AI_ROOT/applicationcontainer_external/siebelwebroot/images` directory, set the File Name property to `image.gif`. - That resides in the `SIEBEL_AI_ROOT/applicationcontainer_external/siebelwebroot/images/bttns` directory, set the File Name property to `bttns/image.gif`.
Height	Enter the height of the bitmap, in pixels.
Width	Enter the width of the bitmap, in pixels.

6. If you must create a bitmap for a BMP file, then do the following:
   a. Right-click the record in the Bitmaps list, and then click Import Bitmap.
   b. In the Open dialog box, locate the BMP file that you must import, and then click Open.
   Depending on the image you choose, Siebel Tools sets some properties, such as Height and Width. It imports the BMP file into the Siebel repository the next time you deliver your Workspace.

Displaying an Icon on a Button

To display an icon instead of text on a button, you can associate a bitmap object with a button control, similar to a Toolbar icon. Unlike a Toolbar icon, a bitmap button control is a command button in the applet. For example, the More/Less button uses a bitmap object with a button control. Siebel CRM displays the More/Less button in the second quadrant of many applets. The control uses the BTTNS_MORE bitmap object that is part of the HTML Control Icons bitmap category.

To display an icon on a button

1. Create a bitmap object.
   For more information, see *Configuring a Bitmap Category and a Bitmap*.
2. In the Object Explorer, click Applet.
3. In the Applets list, locate the applet that contains the control you must modify.
4. In the Object Explorer, expand the Applet tree, and then click Control.
5. In the Controls list, locate the control you must modify.
6. Define properties for the control using values from the following table.

Property	Description
HTML Bitmap	Choose the bitmap object Siebel CRM must use if the button is active.
HTML Disable Bitmap	Choose the bitmap object Siebel CRM must use if the button is not active.

For more information, see *About Applet Controls and List Columns*.

Displaying an Icon as a Link

To display an icon as a link, you must make sure the properties are set correctly. Siebel CRM uses the contents of the Caption property of the control as the label for the link in the following situations:

- The HTML Type property of the control is set to Button.
- The HTML Bitmap and HTML Disabled Bitmap properties are not set.

To display an icon as a link

- Perform the procedure described in *Displaying an Icon on a Button*, but make sure you set the HTML Type property of the control to Link.

Using Custom HTML Types with a Link

You can use the HTML Bitmap and HTML Disabled Bitmap properties with custom HTML types. If you use the following tag in the definition of the custom HTML type in the web template definition containing SWF (Siebel Web Format) content, then the Siebel Web Engine uses the bitmaps:

```
<div od-type="Data" type="Link">
```

These bitmaps must exist in the HTML Control Icons bitmap category.

Using Icons to Represent Values in a Field

An *icon map* is an object that allows you to represent the values that Siebel CRM displays in a control or list column as icons. Each icon map includes a collection of child icon objects. Siebel CRM associates these icon objects with the bitmap object that defines the image for the icon, and corresponds to a field value. The Icon Map property of a control or list column allows you to define the icon map that Siebel CRM uses to display the values in a field.

The example in this topic uses the Status list column on the Activity List Applet. Assume that the Status field can include the following values:

- Not Started
- In Progress
- Done

You must configure the Status field to display an icon for each of these values.

If you must configure Siebel CRM to use a custom icon in a list applet, then you must size the icon according to the row font size that the list applet uses. For example, an eight point font is typical for Siebel CRM. If you use an eight point font, then the icon must be 23 pixels wide by 14 pixels high. If you modify the list applet row font size dynamically, or if you place an icon that is larger than 23 pixels by 14 pixels in a row, then Siebel CRM scrambles the list applet rows.

To use icons to represent values in a field

1. Create a bitmap category named Activity Status Icons.

For more information, see *Configuring a Bitmap Category and a Bitmap*.

2. In the Bitmaps list, create three new bitmap objects for each image that you must display using values from the following table.

Name	File Name
Not Started	notstarted.gif
In Progress	inprogress.gif
Done	done.gif

For more information, see *Configuring a Bitmap Category and a Bitmap*.

3. In the Object Explorer, click Icon Map.
4. In the Icon Maps list, create a new icon map named Activity Status.
5. In the Object Explorer, expand the Icon Map tree, and then click Icon.
6. In the Icons list, create three new icon objects for each field value using values from the following table.

Name	Bitmap Category	Bitmap
Not Started	Activity Status Icon	Not Started
In Progress	Activity Status Icon	In Progress
Done	Activity Status Icon	Done

Note how you must set the following properties:

- **Name.** Set to the name of the field value.
- **Bitmap Category.** Set to the bitmap category that you must use for the field value.
- **Bitmap.** Set to the bitmap object that you must use for the field value.

7. In the Object Explorer, click Applet, and then locate the Activity List Applet in the Applets list.
8. In the Object Explorer, expand the Applet tree, expand the List tree, and then click List Column.
9. In the List Columns list, query the Name property for Status.
10. Set the HTML Icon Map property to *Activity Status.*

This step configures Siebel CRM to use the icon map that you created in Step 3. For more information, see *Using a Default Icon in an Icon Map*.

11. Test and then deliver your Workspace.

For more information, see *Using Siebel Tools* .

Using a Default Icon in an Icon Map

If you use icons to represent values in a field, then the Siebel Web Engine displays the image that the bitmap references if the field value matches one of the icons you define. If the field value does not match any of the icons you define, then the Siebel Web Engine uses text to display the actual field value.

You can create an icon named Default in an icon map. If the field value does not match any of the icons, then Siebel CRM uses the Default icon to represent values in the field. This feature is useful to create an icon that Siebel CRM uses if a field might contain different values, such as URLs.

To use a default icon in an icon map

1. Create or locate an icon map that contains only one icon, named Default.
   For more information, see *Using Icons to Represent Values in a Field*.
2. Define the control or list column using values from the following table.

Property	Description
HTML Type	Set to URL.
HTML Icon Map	Set to an icon map that contains only one icon, named Default.

Configuring Icons in a Tree Applet

An *HTML hierarchy bitmap* is an object that defines the icons that Siebel CRM displays in a hierarchical object, such as a tree applet. To view an example, do the following:

- In the Siebel client, click the Accounts screen tab, and then click the Explorer link.
  Siebel CRM displays the tree portion of the Account Tree Applet. The HTML *hierarchy bitmap* defines the icons that Siebel CRM uses to represent the folders, the plus symbol, and the minus symbol in the Account Tree Applet.

To configure icons in a tree applet

1. Open Siebel Tools.
2. If an existing bitmap does not meet your requirements, then you must modify an existing or create a new bitmap.
   The HTML hierarchy bitmap references bitmaps in a bitmap category. For more information, see *Configuring a Bitmap Category and a Bitmap*.
3. If an existing HTML hierarchy bitmap does not meet your requirements, then you must modify an existing or create a new HTML hierarchy bitmap.
   You can specify the icons that an HTML hierarchy bitmap references. For more information, see *Properties of an HTML Hierarchy Bitmap*.
4. In the Object Explorer, click Applet.
5. In the Applets list, locate the applet that contains the tree you must modify.

6. In the Object Explorer, expand the Applet tree, and then click Tree.
7. In the Trees list, locate the tree you must modify, and then set properties for the tree using values from the following table.

Property	Value
HTML Hierarchy Bitmap	Enter the name of any HTML hierarchy bitmap.

   To modify an object in a list, do Step 6 and Step 7 for a list object. For more information, see *How Applet Objects Reference an HTML Hierarchy Bitmap*.
8. Optional. Define the tree node.

   The tree node object is a child of the tree object. It includes the optional HTML Open Bitmap and HTML Close Bitmap properties:

   - If you define these properties, then Siebel CRM uses them for the node where the properties are defined. This is useful if different nodes must display different icons.
   - If you do not define these properties, then Siebel CRM uses the Open Bitmap and Close Bitmap properties of the HTML Hierarchy Bitmap object.

   For more information, see *Configuring a Tree Applet*.
9. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

How Applet Objects Reference an HTML Hierarchy Bitmap

The tree and list objects are child objects of the applet object. They include the HTML Hierarchy Bitmap property. You can set this property to the name of any HTML hierarchy bitmap object. This allows different object definitions of the tree object and list object to share the same bitmaps.

A predefined tree applet references the bitmap objects that the following HTML hierarchy bitmap defines:

HTML Hierarchy Icons

Properties of an HTML Hierarchy Bitmap

The following table describes properties that Siebel CRM commonly uses with an HTML hierarchy bitmap.

Property	Description
Name	The name for the HTML hierarchy bitmap object.
Collapse Bitmap, Collapse Elbow Bitmap, Collapse Tee Bitmap	Icons to collapse a node.
Expand Bitmap, Expand Elbow Bitmap, Expand Tee Bitmap	Icons to expand a node.
Elbow Bitmap, Tee Bitmap	Icons to create an elbow (L) or a Tee (T).

Property	Description
Bar Bitmap	Icon to create a vertical line.
Space Bitmap	Icon to create an indent.
Open Bitmap	Icon for a node that Siebel CRM displays in an expanded state.
Close Bitmap	Icon for a node that Siebel CRM displays in a collapsed state.
Leaf Bitmap	Icon for a leaf node.
Arrow Down Bitmap, Arrow Up Bitmap	Icons to scroll a tree up or down.

21 Configuring Siebel Web Templates

Configuring Siebel Web Templates

This chapter describes how to configure Siebel web templates and related entities. It includes the following topics:

- *Configuring Siebel Web Templates and Web Pages*
- *Configuring Web Templates to Display Menus, Toolbars, and Thread Bars*
- *Configuring an HTML Control Type*

Configuring Siebel Web Templates and Web Pages

This topic describes how to configure Siebel web templates and Siebel web pages. It includes the following information:

- *Editing the Layout of a Web Page*
- *Adding Graphics to a Web Template*
- *Displaying Multiple Views on a Page*
- *Configuring How Siebel CRM Displays an Error That Occurs on the Siebel Server*

For other tasks that use Siebel web templates and tags, see the following topics:

- *Displaying Totals for a List Column in an Applet*
- *Using a Control to Allow the User to Click a Link to Activate a Record*
- *Using JavaScript to Configure a Toolbar*

For more information, see *About Siebel Web Templates and Siebel Tags*.

Editing the Layout of a Web Page

The Web Page Object is the top (highest) level object in the web hierarchy that Siebel CRM uses to create web pages, such as the following:

- Login pages
- Error pages
- Container pages

Similar to an applet or view, a web page is associated with a web template. Siebel CRM maps web page objects to placeholders in the template. The Web Page Editor allows you to view and edit web page objects. For more information, see *Editing the Layout of a View*.

To edit the layout of a web page

1. Make sure the configuration context is set.
2. In Siebel Tools, in the Object Explorer, click Web Page.
3. In the Web Pages list, locate the web page you must modify, right-click, and then click Edit Web Layout.

4. Choose a custom control from the combo box on the toolbar, and then move it to a placeholder.
5. Use the Properties window to set properties for the control, such as Caption, Method Invoked, and so on.

   After you add controls to the web page, you can choose the Web Page Item object type in the Object Explorer, and then use the Web Page Items list to modify the mappings you just created. For example, you can modify the caption for the Queries menu label, which is the FavoritesLabel web page item.

Multiple Image Display in the Web Layout Editor

The layout editor might display multiple images because the template that the web page references contains a conditional tag, such as <div od-if="<>"> or <div od-case="<>">. The template content varies depending on if one of the conditions is met or is not met. The layout editor displays the page as if all the conditions are true. This is useful if you must edit any of the pages. Only one condition is typically true in the Siebel client, so Siebel CRM does not display redundant images in the Siebel client.

Adding Graphics to a Web Template

To improve the appearance or navigation of your Siebel application, you can create a GIF file and include a link to it from an HTML page.

To add graphics to a web template

1. Place your graphic files in the following directory on the Siebel Application Interface:

   ```
   SIEBEL_AI_ROOT\applicationcontainer_external\siebelwebroot\images
   ```

   The Siebel CRM installer creates the `SIEBEL_AI_ROOT\applicationcontainer_external\siebelwebroot` directory and its subdirectories when you install the Siebel Application Interface.

   The Siebel client also includes directories for a Siebel application to use, including graphics files.
2. For Siebel Tools or Siebel Web Client, place your graphic files in the following directory:

   ```
   public\images
   ```

   The Siebel CRM installer creates the `public` directory and its subdirectories when you install Siebel Tools or Siebel Web Client.
3. Create a link to the graphic file from an HTML page.
4. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Displaying Multiple Views on a Page

The Siebel Web Engine can simultaneously display multiple views on a page. These views include the following main view and one or more alternate views:

- **Main view.** Chosen from the link bar. Only one main view exists.

- **Alternate views.** Other views that Siebel CRM displays with the main view. For example, the Search View that displays applets that it uses for find and search operations.

You can display multiple views in the following ways:

- Place multiple views in separate HTML frames.
- Share multiple views in the same frame.
- Display multiple views in the main view in the main browser window and display a single alternate view in a pop-up window.

It is recommended that you define alternate views as simple views that do not contain complex navigation links.

The examples in this topic describe how to create multiple view layouts if you use HTML frames. The procedure is similar to the procedure you use if you do not use HTML frames. If you do not use HTML frames, then to position the views, you can use HTML tables instead of frames and framesets.

To support multiple views, you must modify the structure of frames and framesets.

You cannot configure Siebel Open UI to display more than one view on a single page, but you can configure it to display multiple applets in a view. For more information, see *Creating a View* and *Configuring Siebel Open UI* .

To display multiple views on a page

1. Replace the first line of the code of the Page Container web template with the following example code:

   ```
   <div od-type="frameset" htmlAttr="rows='80,50,50,*' border='0' frameborder='No'">
   ```

2. Replace the view frame in the container page with a content frame.

   This frame defines the area where Siebel CRM loads one or more views. Initially this frame contains a frameset that includes a view type frame. You can replace this view frame with the following example content frame:

   ```
   <div od-type="frame" type="Content" htmlAttr="marginheight='0' marginwidth='0' noresize
   scrolling='Yes'">
    <div od-include="CCMainView"/>
   <!--od section frame close-->
   </div>
   ```

   For more information, see *Example Code of the CCMainView Web Template*.
3. Modify all the application container templates to use the content frame.
4. To display more views in the content area, load a different content container page in the content frame:

   a. Call the LoadContentContainer method from a control or page item.
   b. Make sure the User Property container loads the content container.

   For more information, see *Using the LoadContentContainer Method to Load Multiple Views*.

Siebel CRM behaves the same before and after you make this modification. You only add one more layer of frames in the content area. The unmodified application container page template included in the view frame without the outer content frame does not create errors. You cannot use it to display multiple views.

Using the LoadContentContainer Method to Load Multiple Views

You must use the Web Template Name of the content container page. For example, to display the search view with the main view, do the following:

- Create a content container page, such as CCSMainAndSearchView.
- Use the LoadContentContainer method to load this page.

To load the main view and search view into two frames, the CCSMainAndSearchView web template contains the following tags:

```
<div od-type="frameset" htmlAttr="cols='100%' border='0' frameborder='No'">
 <div od-type="frame" type="View" htmlAttr="noresize scrolling='Yes'">
 <div od-type="current-view"/>
 <!--od section frame close-->
 </div>
 <div od-type="frame" type="AltView" name="Search" htmlAttr="noresize scrolling='Yes'">
 <div od-type="view" name="Search View" id="Search"/>
 <!--od section frame close-->
 </div>
<!--od section frameset close-->
</div>
```

In this example, you still reference the main view in the tag. You reference alternate views in the tag.

To switch from displaying the search and main views to displaying only the main view, you can call the LoadContentContainer method again, but this time reference the container page that references the CCMainView web template.

OD View Tag

The tag uses the following format:

```
<div od-type="view" name="xxx" id="yyy">
```

The tag includes the following attributes:

- **Name.** Name of the alternate view.
- **Id.** Identifies the location that this view occupies. You use this Id to replace this view with another view.

The tag contains an alternate view named AltView. You can define only one alternate view for each frameset. If you add more than one alternate view, then you might encounter an error.

Example Code of the CCMainView Web Template

The CCMainView web template that you reference in the od-type=frameset code of the CCPageContainer page includes the following frameset. This frameset contains the main view:

```
<div od-type="frameset" htmlAttr="cols='100%' border='0' frameborder='No'">
 <div od-type="frame" type="View" htmlAttr="noresize scrolling='Yes'">
 <div od-type="current-view"/>
 <!--od section frame close-->
 </div>
<!--od section frameset close-->
</div>
```

Configuring How Siebel CRM Displays an Error That Occurs on the Siebel Server

If an error occurs on the Siebel Server when Siebel CRM submits a form, then the Siebel Web Engine displays the same page again and includes an error message. The tag specifies the location of this error message. If an error occurs outside of a form submission, then the Siebel Web Engine continues to use the value that is defined in the Error Web Page property of the application object.

You can use the tag to configure how Siebel CRM displays an error that occurs on the Siebel Server. Note the following behavior:

- If Siebel CRM encounters no errors when it displays a form, then it skips the contents of the tag.
- The only attribute of the tag is a property whose value must equal FormattedHtml. This configuration configures Siebel CRM to display the contents of the error message.
- If you do not use a tag in a Siebel web template, then the code creates an error node, which is an instance of the CSSSWEErrorSWX code. Siebel CRM inserts this error node as the first child of the enclosing page or form node.
- Siebel CRM displays an error message in plain text. It displays each error message in a separate paragraph.
- The enclosing HTML tags determine the font and style of the error message. If the font uses the same color as the background, then the error message is not visible.

To configure how Siebel CRM displays an error that occurs on the Siebel Server

1. To display the error message in a form, place the following tags in the tag:

```
<div od-type="error">
 <div od-property="FormattedHtml"/>
<!--od section error close-->
</div>
```

2. Make sure you use this tag in all tags.

Format of the OD Error Tag

The basic format of the tag is as follows:

```
<div od-type="error">
```

The format of the tag with the FormattedHtml property is as follows:

```
<div od-type="error" property="FormattedHtml"/>
```

or

```
<div od-type="error">
 <div od-property="FormattedHtml"/>
<!--od section error close-->
</div>
```

Example of Using the Error Tag

The following code is an example of using the tag:

```
<div od-type="form">
 <div od-type="error">
 <b><font color="red"> <div od-property="FormattedHtml"/> </font></b>
 <!--od section error close-->
 </div>
...
<!--od section form close-->
</div>
```

For another example, see *Example Code of a Nongrid Form Applet Template*.

Configuring Web Templates to Display Menus, Toolbars, and Thread Bars

This topic describes how to configure web templates to display menus, toolbars, and thread bars. For more information, see *About Menus and Toolbars*. It includes the following information:

- *Using Web Templates to Display Menus and Buttons*
- *Using Web Templates to Configure Toolbars*
- *Using Web Templates to Configure the Thread Bar*

Using Web Templates to Display Menus and Buttons

If the user clicks a menu that is defined as button or link in a Siebel web template, then the Siebel Web Engine uses the tag to activate a list of menu items. The tag displays menus in the following ways:

- Application container page for an application menu
- Applet for an applet menu

Siebel CRM displays an applet menu as an icon button, typically placed before other buttons, such as Edit and Delete. In the Siebel client, it uses the configuration in the Siebel runtime repository to create a set of menu items for an applet. The tag must be defined in an applet web template for applet menus.

Example Code to Display an Application Menu

To display an application menu, you can define the tag in any type of template other than an applet web template. Siebel CRM uses the Menu and Menu Item object definitions in the Siebel repository to display a set of menus from a single tag. The Menu property in the Application object definition references the Menu object definition. This Menu object definition specifies a set of application menus and menu items in each application menu. Siebel CRM predefines some menu items, such as the Logout menu item.

The following example code from the CCFrameBanner web template includes the tag at the start of the definition for a banner:

```
<!--Start Banner-->
<div od-type="menu"/>
<table class="banner" cellpadding="0" cellspacing="0" border="0">
<tr>
 <td width="50%">
 <img src="images/SiebelLogo.gif" width="282" height="57" border="0" alt="" />
 </td>
 <td width="50%">
```

```
 <img src="images/spacer.gif" width="100%" height="58" border="0" alt="" />
 </td>
</tr>
```

How the Menu Tag Displays a Menu

The tag displays menu buttons or links for all menus for the following:

- **Applications.** Displays one button or link for each application menu that is defined for the Siebel application in the menu object definition and children of the menu object definition.
- **Applets.** Displays the applet menu button.

The od-type=menu tag uses the following format:

```
<div od-type="menu" type="XXX" bitmap="XXX" width="XXX" height="XXX" bgcolor="XXX"
fgcolor="XXX"/>
```

The tag includes the following attributes:

- **type.** Can be set to one of the following values:
  - **Default.** Siebel CRM displays the menu and the application menu items. If no value is defined for the type attribute, then it uses the default value.
  - **Button.** Siebel CRM displays a button that displays a menu that includes the menu items if the user clicks the button.
- **bitmap.** Used only if the Type attribute is Button. It defines the name of a bitmap object that Siebel CRM uses as the label for the button. This bitmap is defined in Siebel Tools in the HTML Control Icons bitmap category.
- **width.** Defines the width of the menu in pixels. For more information, see *Localizing an Application Menu*.
- **height.** Defines the height of the menu in pixels.
- **bgcolor.** Defines the background color of the menu. You must use the hexadecimal triplet format that HTML requires. For example, #FFFFFF.
- **fgcolor.** Defines the foreground color of the menu. You must use the hexadecimal triplet format that HTML requires.

Example Code to Display Applet Buttons

The following code from the CCFormButtons web template is an example of the template that Siebel CRM uses to display applet buttons, including the menu button:

```
<!-- Buttons (Edit, Delete, Optional, Optional, Optional) --->
<!-- Menu,179 -->
 <td valign="middle" nowrap>
 <div od-type="menu"/>
 </td>
 <td valign="middle"><img src="images/bttns/div.gif" height="13" border="0"
 align="middle" alt=""></td>
<!-- EditRecord -->
<div od-type="control" id="132">
 <td valign="middle" nowrap>
 <div od-property="FormattedHtml" hintText="Edit" hintMapType="Control"/>
 </td>
 <td> </td>
<!--od section control close-->
</div>
(and so on...)
```

Using Web Templates to Configure Toolbars

Siebel CRM displays a toolbar as a customizable JavaScript toolbar.

A JavaScript toolbar object resides in the JSSApplication hidden frame, which typically does not reload during the life cycle of the Siebel application. Siebel CRM does not redraw when a page refresh occurs. The user interface part of the JavaScript toolbar resides in a visible HTML frame and redraws when the HTML frame reloads. It is recommended that the visible HTML frame is a persistent frame that reloads infrequently. HTML toolbars reside in the topmost frame in the application template that Siebel CRM reserves for this purpose.

Tags That You Use with a Toolbar

This topic describes the toolbar tags that you use with a web template.

OD Tag That Specifies a Toolbar

The tag specifies a toolbar where the name corresponds to the Name property in the Toolbar object definition. Siebel CRM currently supports two types of toolbars: HTML toolbars and Java applet toolbars, as defined in the javaapplet attribute.

The tag uses the following format:

```
<div od-type="toolbar" name="XXX" javaapplet="true/false" width="XXX" height="XXX"/>
```

The tag includes the following attributes:

- **Name.** The name of the toolbar as defined in Siebel Tools.
- **Javaapplet.** Must be set to true for a Java toolbar, and false for an HTML toolbar.
- **Width.** Width of the toolbar in pixels.
- **Height.** Height of the toolbar in pixels.

SWE Tag That Gets Toolbar Items

The tag recursively gets all of the toolbar items for that toolbar from the Siebel repository. It exists between the toolbar start tag and end tag.

The tag uses the following format:

```
<div od-type="toolbaritem">
```

The tag does not include any attributes.

Using Templates to Configure HTML and JavaScript Toolbars

You can define an HTML or JavaScript toolbar.

To use templates to configure HTML and JavaScript toolbars

1. Add the following code to the Siebel web template:

```
<div od-type="toolbar" name="xxx"> // where xxx is the name of toolbar in the
repository.
 // any HTML stuff here...
 <div od-type="toolbaritem">
```

```
 // any HTML stuff here...
<!--od section toolbar close-->
</div>
```

2. For combo box items, make sure you target the command to a service.

Configuring a Java Toolbar

You use a Java applet to configure the toolbar. This applet includes all the toolbar controls and the threads that interact with the Siebel Server. The Java applet calls the following methods:

- ShellUIInit method on the command target service when the applet attempts to initialize
- ShellUIExit method when the applet exits

A set of communication protocols is defined for the communication between the Java Applet and the service.

To configure a Java toolbar

- Add the following code to the Siebel web template:

```
<div od-type="toolbar" od-name="xxx" javaapplet="true"/>
```

Using Web Templates to Configure the Thread Bar

The thread bar includes thread buttons that Siebel CRM displays in the following format:

```
title: value
```

You can omit the title or value. Separators separate thread buttons. For example, the greater than symbol (>) is a separator.

If a thread applet or thread field is not defined for a view, then Siebel CRM does not update the thread button when it displays the view.

The following table describes how Siebel CRM responds to actions the user performs in the thread bar. For more information, see *Configuring the Thread Bar*.

User Action	Siebel CRM Reply
User requests a new screen.	Siebel CRM creates a new thread to replace the current thread.
User clicks a view button.	Siebel CRM replaces the last thread with the new view that the user requested.
User clicks a drilldown link.	Siebel CRM appends a new step on the thread bar for the view that the user requested.
User clicks a thread button.	Siebel CRM deletes all the thread buttons after the thread button that the user clicked and proceeds to the step view that SWEBMCount indicates.

How Siebel CRM Uses Bookmarks with the Thread Bar

A thread button can display a link that navigates the user to a previous page. The link requires the GotoBookmarkView Siebel Web Engine command. The link for each thread button must contain at least the following parameters:

```
SWECmd=GotoBookmarkView&SWEBMCount=2SWECount=3
```

where:

- *SWEBMCount=2* indicates that Siebel CRM uses bookmark number 2 to create the view.
- *SWECount=3* is the bookmark ID for the current view.

For example, Siebel CRM uses the od tags and thread link format to translate the thread button for the A.K. Parker account into the following HTML format:

```
<a href="https://www.mycompany.com/start.swe?SWECmd=GotoBookmarkView&SWEBMCount=2&
SWECount=3>Account: AK Parker</a>
```

If the user clicks the thread button to display a bookmarked view that the user previously accessed, then Siebel CRM creates a new bookmark that identifies the view that it currently displays. The bookmark ID for the new view is the current od count increased by 1. The od count is the count that Siebel CRM sends to the Siebel Server in the request.

Bookmark deletion policy is not modified with the bookmark ID assignment policy. Siebel CRM keeps the most recently created 20 bookmarks and deletes all other bookmarks, by default. If the od count in the user request is less than the od count on the Siebel Server, then Siebel CRM deletes all the bookmarks that contain an od count that is larger than the od count in the user request.

Configuring the Thread Bar

You can use the following od tags to configure an HTML thread bar:

- ** .** Defines the start and finish of the thread bar section.
- ** .** Defines the definition of a thread button on the thread bar. This tag includes the following properties:
  - **FormattedHtml.** Display the HTML link.
  - **Title.** Display the title and value pair of the thread button.
- ** .** Specifies the symbol that Siebel CRM displays to separate thread buttons.

Use the and tags only in the tag.

The usage of these od tags is similar to that of the screen bar and view bar tags.

To configure the thread bar

- Insert thread bar definitions into a Siebel web template. Use the , , and tags. For usage with frames, do the following:
  - **Application does not use frames.** Insert the definition in a container page. For example, CCPageContainer_NoFrames.
  - **Application uses frames.** Insert the definition in the Siebel web template for the Viewbar frame or the View frame.

Example Code to Configure the Thread Bar

The code in this topic creates a thread bar that uses the following format:

```
Home > Consumer:PCs > PCs:Laptops
```

The following code provides an example of how to insert thread bar definitions into a Siebel web template:

```
<!-- Begin Threadbar section -->
<table class="threadbar" width="100%" border="0" cellspacing="0" cellpadding="0">
<tr valign="left">
 <td nowrap bgcolor="#6666CC" width="110">
 <img src="images/spacer.gif" width="110" height="25" border="0">
 </td>
 <td width="99%">
 <div od-type="threadbar">
 <img src="images/spacer.gif" width="10" height="25" border="0"
 align="absmiddle">
 <div od-type="threadlink" property="FormattedHtml">
 <font color="#000000"><span> <nobr><div od-property="Title"/></nobr>
 </span></font>
 <!--od section threadlink close-->
 </div>
 <div od-type="threadseparator">&gt;<!--od section threadseparator close-> </div>
 <!--od section threadbar close-->
 </div>
 <img src="images/spacer.gif" width="10" height="25" border="0"
 align="absmiddle">
 </td>
</tr>
</table>
<!-- End Threadbar section -->
```

Configuring an HTML Control Type

This topic describes how to configure an HTML control type. It includes the following information:

- *Creating a New HTML Type*
- *How the Siebel Web Engine Uses a Custom HTML Type*
- *Configuring an HTML Type*

Siebel CRM supports different control types. For example, Check Box, Button, Mail To, Text Area, and so on.

Comparison of Using Cascading Style Sheets and Siebel Web Templates to Configure an HTML Control Type

You can use a cascading style sheet to define stylistic information about labels, titles, background colors, and so on.

You use the CCHtmlType web template definition, which contains Siebel Web Format (SWF) content, to define more complex attributes that determine the appearance or client functionality of a type of HTML element. For example, a button type that is associated with a GIF image, or a type of link that connects the user with an FTP site.

To add qualities to a page element, you can define tags and attributes in the Siebel web template. You can define types in the CCHtmlType web template definition, which you can reference in Siebel Tools for a control on an applet web template or web page object. This configuration preserves the generality of the Siebel web template by avoiding the

need to place HTML directly in the template. It reduces customization in the templates and stores more configuration information in the Siebel repository. This configuration reduces maintenance of Siebel CRM.

Creating a New HTML Type

You can create a new HTML type.

To create a new HTML type

1. Add the name of the new type to the List of Values used for the HTML Type property in Siebel Tools (REPOSITORY_HTML_CTRL_TYPE).

   MiniButton is an example of a name.
2. Modify a web template definition as follows:
   a. Add the format information for the new type.

      Siebel CRM uses two web template definitions that contain Siebel Web Format (SWF) content. One of these web template definitions contains the special types that Siebel CRM defines. The other contains custom definitions that you define to add more types or to override Siebel types.

      Use the swe:htmltype and <div od-property="<>"> tags to define how to display the custom type using the following format:

      ```
      <swe:htmltype name="XXX" mode="AAA" state="BBB">
       ..... HTML .....
       <div od-property="<YYY>"/>
       .... More HTML ....
      </swe:htmltype>
      ```

   b. Set the UserSWFName parameter to the name of the web template definition containing SWF content that the Application Object Manager must use.

      For more information about the Application Object Manager, see *Siebel System Administration Guide* .
3. In Siebel Tools, modify the HTML Type property of the control, list column, or page item to the new type.
4. In the template file, use the FormattedHTML property for the tag or the tag.

How the Siebel Web Engine Uses a Custom HTML Type

If the HTML type of a control, list column, or page item is a custom type, then the Siebel Web Engine uses the SWF format when it displays any element that is mapped to the control, and that defines the FormattedHtml property. The Siebel Web Engine does not use the format with any other property, such as Display Name. In the web template definition containing SWF content, the <div od-property="<>"> tag can reference these properties, except the FormattedHtml property.

Note the following examples:

- `<div od-type="control" id="1" od-property="FormattedHtml"/>`
- `<div od-type="control" id="1" ... od-property="FormattedHtml"/> ... <!--od section control close--> </div>`

- `<div od-type="pageitem" id="1" od-property="FormattedHtml"/>`
- `<div od-type="pageitem" id="1" ... od-property="FormattedHtml"/> ... <!--od section pageitem close--> </div>`

Format Requirements of Siebel Web Format (SWF) Content

For a web template definition that contains Siebel Web Format (SWF) content, you must make sure that each format specification includes the following parts:

- An enclosing XML element that names the type and optionally names the mode and state that Siebel CRM uses for the current format
- The enclosed format content

You must make sure the content format meets the following requirements:

- It must be a valid, regular Siebel Web Engine format.
- It can reference all the properties of the current control, except FormattedHtml. To prevent recursion, it cannot reference FormattedHtml.
- It can use the Data property in the <div od-property="<>"> tag.

Configuring an HTML Type

This topic describes examples of configuring an HTML type.

Configuring an HTML Type for a Control

The following example code creates a custom HTML type for the LabelRed control that displays the caption of the control in red:

```
<swe:htmltype name="LabelRed">
 <font color="red"> <div od-property="DisplayName"/> </font>
</swe:htmltype>
```

Configuring an HTML Type for an Applet Mode

You use the Mode attribute of the swe:htmltype tag to define a custom format for the Base, Edit, New, and Query applet modes. If you define a mode, then the Siebel Web Engine uses the format you define only if the current show mode matches the value defined for this attribute. For example, assume you create a new HTML type named SiebelText to display a control that Siebel CRM displays in the following ways:

- As a label and a text field in Edit mode
- As read-only text in Base mode

In this example, you use the following code:

```
<swe:htmltype name="SiebelText">
 <div od-property="Data" type="Text"/>
</swe:htmltype>
<swe:htmltype name="SiebelText" mode="Edit">
 <div od-property="DisplayName"/>: <div od-property="Data" type="Text"/>
</swe:htmltype>
```

For more information, see *Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data*.

Configuring an HTML Type when Siebel CRM Cannot Call a Method

To display a different image depending on the state of a control or list column, you can define an optional attribute of the swe:htmltype tag, such as State. You can use the following states:

- **Disabled.** For a control or list column that calls a method, when Siebel CRM cannot call the method on the record.
- **Required.** For a control or list column that is required.

For example, to display a gray button when Siebel CRM cannot call a method, add the following code in addition to the default definition described earlier in this topic:

```
<swe:htmltype name="MiniButton" state="Disabled">
 <img src="images/graybtn_left.gif" border="0" height="15" width="2">
 <div od-property="<Data>" type="Link"/>
 <img src="images/graybtn_right.gif" border="0" height="15" width="2">
</swe:htmltype>
```

Note how Siebel CRM handles calls differently:

- If it cannot call a method with a predefined HTML type, then it does not display the control or list item.
- With a custom HTML type, it always uses the format defined in the web template definition containing SWF content to display the control or list item. The HTML that it creates for the following Data property when it cannot call a method is the caption of the control or list item without any href tags:

```
<div od-property="<Data>" type="Link">
```

If Siebel CRM cannot call a method, then you can use a custom HTML type to hide a control or list column. For example, you can create the following empty swe:htmltype tag for the Disabled state:

```
 <swe:htmltype name="MiniButton" state="Disabled"></swe:htmltype>
```

This code hides only the tag or the <div od-property="<>"> tag that calls the FormattedHtml property.

Configuring an HTML Type to Indicate a Required Field

To display the SiebelText type with an asterisk (*) to indicate a required field, you can add the following example code in addition to the definitions for this type described earlier in this topic:

```
<swe:htmltype name="SiebelText" mode="Edit" state="Required">
 * 
 <div od-property="DisplayName"/>
 : 
 <div od-property="Data" type="Text"/>
</swe:htmltype>
```

The Siebel Web Engine uses the following order of precedence when it looks up HTML Type definitions in the web template definition containing SWF content:

1. Mode
2. State

It is recommended that you always create a default format definition for all custom HTML types. To create a default format definition, you define it without specifying the mode attribute and state attribute.

Using the Data Property of the OD This Tag

The Data property of the <div od-property="<>"> is similar to a macro that casts the current, custom type to one of the intrinsic types, then inserts the FormattedHtml property of the intrinsic type. To use the Data property, you add a Type attribute to the <div od-property="<>"> tag. This Type attribute names the intrinsic type. For example, use the following code to create a new type named MiniButton. This code adds a special format to the Siebel Web Engine intrinsic type named Link:

```
<swe:htmltype name="MiniButton">
 <img SRC="images/btn_left.gif" border="0" height="15" width="2">
 <div od-property="<Data>" type="Link"/>
 <img src="images/btn_right.gif" border="0" height="15" width="2">
</swe:htmltype>
```

The following code outputs the same HTML as if the template included a separate <div od-property="<>"> tag, where the property is FormattedHtml and the HTML type of the control is the predefined Link type:

```
<div od-property="<Data>" type="Link">
```

You can only define a predefined type and not a custom type for the type attribute of a Data element.

22 Improving the Performance of Siebel CRM

Improving the Performance of Siebel CRM

This chapter describes how to tune and improve the performance of a Siebel CRM application. It includes the following topics:

- *Using the Case Insensitivity Wizard to Improve Query Performance*
- *Improving the Performance of a Siebel Application*

For more information, see *Siebel Performance Tuning Guide* .

Using the Case Insensitivity Wizard to Improve Query Performance

This topic describes how to use the Case Insensitivity Wizard to configure columns to support a CIAI query. It includes the following information:

- *How a CIAI Index Can Improve a Query*
- *Overview of the Case Insensitivity Wizard*
- *Variables You Can Use with the Case Insensitivity Wizard*
- *Using the Case Insensitivity Wizard on a Table*
- *Using the Case Insensitivity Wizard on a Table Column*
- *Using the Case Insensitivity Wizard on Columns That Do Not Contain an Index*
- *Using the Case Insensitivity Wizard to Do Various CIAI Configuration Tasks*
- *Using the Case Insensitivity Wizard to Deactivate CIAI Configuration*
- *Choosing the Correct Repository when Running the Case Insensitivity Wizard*
- *Limiting the Length of Schema Object Names Manually*
- *Other Configurations to Set Case Sensitivity*

For more information, see *How a CIAI Index Can Improve a Query*.

Related Books

Siebel Database Upgrade Guide

Siebel Global Deployment Guide

Siebel Performance Tuning Guide

How a CIAI Index Can Improve a Query

The *CIAI query* is a feature that uses an index to support a case-insensitive and accent-insensitive (CIAI) query on some text columns. The purpose of the CIAI query is to improve query performance. If a database uses a CIAI index to perform a search, then the Siebel database is not required to perform table scans to locate records, and the database can do the search more quickly.

For example, in the S_CONTACT table, assume the LAST_NAME column is defined for a CIAI query and uses the LAST_NAME_CI column. Assuming that your deployment uses an IBM DB2 database, if you query for the name Smith, then the object manager creates a query that is similar to the following:

```
SELECT column list FROM S_CONTACT
WHERE LAST_NAME_CI = SMITH
```

The Siebel database then uses the CIAI index on LAST_NAME_CI to locate the records.

For Text and CLOB physical types, the Case Insensitivity Wizard does the following work:

- Accepts the Text or CLOB physical type
- Does not create a CIAI column or CIAI indexes for a Text or CLOB physical type
- Sets the Default Insensitivity property to DB Case & Accent

For more information, see *Types of Tables and Columns That CIAI Query Supports*.

How Siebel CRM Implements a CIAI Column and Index in a Database

The type of database determines how Siebel CRM implements a CIAI column or index that exists in the Siebel repository. For example, it implements a CIAI column in the following ways:

- On Microsoft SQL Server, as a calculated column
- On Oracle Database, as indexes that use functions
- On IBM DB2, as a schema column

Effect of CIAI Columns on Sorts

If querying on a column that is configured for CIAI, then the ORDER_BY clause might or might not include the CIAI column depending on the following situations:

- A view with the Visibility Applet Type property set to All uses the CIAI column in the ORDER_BY clause.
- A view with the Visibility Applet Type property set to Org does not use the CIAI column for sorts.

If querying on another column in the same view that is not configured for CIAI, then Siebel CRM does not use the CIAI column in the ORDER_BY clause.

Overview of the Case Insensitivity Wizard

The *Case Insensitivity Wizard* is a tool you can use to configure a column to support a CIAI query. This wizard does the following work:

- If you use an input file, then validates the format of all records in the input file. For more information, see *Input File You Can Use with the Case Insensitivity Wizard*.

- Validates that all tables and columns are eligible for CIAI configuration. For more information, see *How the Case Insensitivity Wizard Verifies Eligibility*.
- For each eligible base column, defines a new CIAI column and a CIAI index in the Siebel repository. The CIAI column contains data in the base column. The wizard converts this data to uppercase. For more information, see *Index Strategy Variable of the Case Insensitivity Wizard*.
- Sets the Default Insensitivity property for the base column to DB Case & Accent. You can run the Case Insensitivity Wizard in a special mode to set the Default Insensitivity property on columns that do not contain an index.
- Sets flags and does other configuration operations in the Siebel repository that Siebel CRM requires to support a CIAI query.

Siebel Tools or Web Tools does not create columns or indexes in the Siebel logical schema until you deliver your Workspace.

How the Case Insensitivity Wizard Verifies Eligibility

The Case Insensitivity Wizard verifies that all tables and columns it configures for CIAI meet the following eligibility criteria:

- The table and column exist in the Siebel repository.
- The column is active and belongs to the defined table.
- Siebel CRM supports the table type, column functional type, and column physical type for each CIAI configuration.
- The column already includes one or more indexes. If no index is defined on the column, but the column is otherwise eligible, then the wizard accepts the column but does not create a CIAI column or any CIAI indexes for the column. The Case Insensitivity Wizard sets the Default Insensitivity property to DB Case & Accent. For more information, see *Using the Case Insensitivity Wizard on Columns That Do Not Contain an Index*.

How the Case Insensitivity Wizard Applies a Naming Format

When the Case Insensitivity Wizard creates a CIAI column name or index name, it uses a naming format that is similar to the naming format that the EIM Table Mapping Wizard uses. The naming format is fixed. You cannot override it. For more information, see *Mapping a Custom Table to an Interface Table* and *Objects You Use with Enterprise Integration Manager*.

The Case Insensitivity Wizard uses the following naming format:

- **Column name.** Appends a _CI suffix to the CIAI column name. For example, if the parent column is LAST_NAME, then the CIAI column is LAST_NAME_CI.
- **Index name.** Uses the following format to append a string to the base table name:

  BASE_TABLE_NAME_C#

  where:

  - # is an integer starting at 1 and incremented as required to create a unique name.

  For example, S_CON_ADDR_C1 is a CIAI index that the wizard creates for the S_CON_ADDR table.

How the Case Insensitivity Wizard Limits the Length of An Object Name

The default length for a column name or index name is 30 characters. If a CIAI column name or index name exceeds the maximum length, then the Case Insensitivity Wizard does the following:

- Truncates the column base name for a column name
- Truncates the table base name for an index name

The wizard does the following:

- Deletes underscores one at a time, beginning with first underscore.
- Deletes vowels one at a time, beginning with the last vowel.
- Deletes characters one at a time, beginning with the last character. Characters include letters, numbers, and so on.

The Case Insensitivity Wizard does not truncate a prefix or a suffix.

You can manually limit the length of schema object names to 18 characters. For more information, see *Limiting the Length of Schema Object Names Manually*.

How the Case Insensitivity Wizard Makes Sure Each Name Is Unique

If the Case Insensitivity Wizard truncates a column or index name, then the name might not be unique. If this situation occurs, then the wizard truncates the farthest character in the base column name or base table name. The wizard replaces the truncated character with an integer, starting with 1. To maintain the overall string length, the wizard does the following:

- Increments the integer if the truncated name is not unique
- Truncates the name to make room for more digits if the wizard requires more digits to make the name unique

How the Case Insensitivity Wizard Reports an Error

The Case Insensitivity Wizard reports errors in a pane in the Case Insensitivity Wizard. The errors list provides information so that you can identify the column and the cause of the problem. You can do one of the following:

- Correct the errors and rerun the Case Insensitivity Wizard.
- Ignore the errors. When the Case Insensitivity Wizard configures columns, it skips each column that creates an error.

You can export errors that the Case Insensitivity Wizard reports to a text file. Errors typically fall into one of the following categories:

- **Input file format error.** Punctuation error or improper use of configuration options.
- **Table and column eligibility problem.** Occurs if you choose tables and columns that the Case Insensitivity Wizard does not support.
- **Project not locked.** You must lock any table you define in the Table variable before you run the wizard. The wizard displays the list of projects that you must lock.

Variables You Can Use with the Case Insensitivity Wizard

The Case Insensitivity Wizard includes the following variables:

- Table

- Column
- Method
- Index Strategy
- Operation

Method Variable of the Case Insensitivity Wizard

The method variable of the Case Insensitivity Wizard determines how the wizard configures a CIAI query for a column. This topic describes the methods that are available.

About the Force Case Method

If you set the method to Force Case, then the Case Insensitivity Wizard does not create a CIAI column or index. You can use the Force Case method for a column where the Force Case property of a table column is already set.

If the Force Case property is FirstUpper, Lower, or Upper, then Siebel CRM forces the column data to the case that is set in the Force Case property before it writes data to the Siebel database. All data in the base column is in the same case. The object manager can use the base column and the base column indexes for a query that is not case-sensitive. A CIAI column and CIAI indexes are not required. To get records, the object manager uses the indexes that Siebel CRM already defines on the base column.

If Force Case is FirstUpper, LOWER, or Upper, then the Case Insensitivity Wizard considers Force Case to be set for a column. If Force Case is empty, then the wizard does not consider Force Case to be set.

About the Database Method

The Database method defines a CIAI column for the base column. It uses the index strategy variable to create indexes. The Case Insensitivity Wizard does the following work for indexes that contain multiple columns as keys:

- For the first key where the column becomes CIAI enabled, the wizard defines a copy of the index. In the copy, the key references the CIAI column instead of the base column.
- For each additional key that is CIAI enabled, the wizard deletes the index copy in the Siebel repository and redefines it so keys reference the additional CIAI columns.

For example, assume the following:

1. The S_CONTACT table contains Base Column A. This column includes Index A, which uses the LAST_NAME and FST_NAME columns as keys.
2. You choose the LAST_NAME column for a CIAI query and define the Copy All index strategy.
3. The Case Insensitivity Wizard defines the LAST_NAME_CI column and a CIAI index for the new column.
4. To create Index B for Base Column A, the wizard copies Index A and specifies LAST_NAME_CI and FST_NAME as keys.
5. You choose the FST_NAME column for a CIAI query.
6. As part of configuring FST_NAME for a CIAI query, the wizard does the following work in the Siebel repository:
    - Defines a new FST_NAME_CI column.
    - Deletes Index B on Base Column A and redefines it with the LAST_NAME_CI and FST_NAME_CI keys.

For more information, see *Index Strategy Variable of the Case Insensitivity Wizard*.

Index Strategy Variable of the Case Insensitivity Wizard

The *index strategy* is a variable that determines how the Case Insensitivity Wizard defines indexes for the CIAI column.

The following table describes the index strategies you can use with the method variable set to Database. The wizard sets the Default Insensitivity property to DB Case & Accent no matter which index strategy you use.

Index Strategy	Work That the Case Insensitivity Wizard Performs
None	Defines no new CIAI columns or indexes.
Single	Defines a new CIAI column and defines a single CIAI index on it. For every index that includes the base column, the wizard does not create another index that references the CIAI column.
Copy All	Defines a new CIAI column and a CIAI index for the column. For every index that includes the base column, the wizard defines a copy of that index. The copy references the CIAI column instead of the base column.

How the Case Insensitivity Wizard Uses Default Values

The Table Name and Column Name are the only required variables for the Case Insensitivity Wizard. If you omit the other variables, then the Case Insensitivity Wizard uses the defaults that this topic describes.

Default Values for the Method Variable

The Case Insensitivity Wizard uses the following default values for the method variable:

- If the Force Case property is set on the table column, then the wizard uses the Force Case method.
- If the Force Case property is not set on the table column, then the wizard uses the Database method.

Default Values for the Index Strategy Variable

The Case Insensitivity Wizard uses the following defaults for the index strategy variable:

- If the method is Force Case, then the wizard sets the index strategy to None.
- If the method is Database, and if the base column does not contain an index, then the wizard sets the index strategy to None.
- If the method is Database, and if the base column contains an index, then the wizard uses the Copy All index strategy.

If the Case Insensitivity Wizard uses None as an index strategy, then the wizard does not define new columns or indexes. It sets the Default Insensitivity property to DB Case & Accent.

The Case Insensitivity Wizard runs the following default logic:

- If the Force Case property is set on a column, then the wizard does not define columns or indexes.
- If the column contains an index, then the wizard does not define columns or indexes.

In these situations, the Case Insensitivity Wizard accepts the column as eligible but does not define columns or indexes. These default behaviors define implicit eligibility requirements.

Default Values for the Operation Variable

If you do not include the Operation variable, then the Case Insensitivity Wizard sets Operation to On, regardless of the method or index strategy.

Using the Case Insensitivity Wizard on a Table

If you run the Case Insensitivity Wizard to configure a large number of columns for a CIAI query, or if you must use a nondefault method or index strategy, then you can use an input file. The wizard reads the input file, and then configures the columns in the file. Using an input file allows you to control the configuration options that the Case Insensitivity Wizard uses. Oracle provides recommended input files. For more information, see *Input File You Can Use with the Case Insensitivity Wizard*.

To use the Case Insensitivity Wizard on a table

1. In Siebel Tools, open the repository.

   For more information, see *Choosing the Correct Repository when Running the Case Insensitivity Wizard*.
2. Lock the tables that are listed in the input file.
3. Click the Tools menu, click Utilities, and then click Case Insensitivity.
4. Choose Administer the Columns Listed in This File, and then click Browse.

   Siebel Tools displays the `tools\objects` directory, which contains the default csv input files.
5. Choose the csv file, click Open, and then click Next.

   The Case Insensitivity Wizard validates the following:

   - Format of the input file
   - Eligibility of all tables and columns

   If the file contains an error, then the wizard lists the records that contain the errors. If you continue, then the wizard skips records that contain errors. If errors exist, then click Export to export the error list to a text file, correct the errors, and then restart the Case Insensitivity Wizard.

   For more information, see *Input File You Can Use with the Case Insensitivity Wizard*.
6. Click Next.

   The Case Insensitivity Wizard displays the records in the input file.
7. Review the configuration settings and verify they are correct.

   If you must modify any configuration settings, do the following:

   a. Click Export.

      The Case Insensitivity Wizard exports the list to a text file.
   b. Edit the text file, and then restart the Case Insensitivity Wizard, specifying the edited text file as the input file.
8. Click Next.

   The Case Insensitivity Wizard displays the modifications it will make to repository tables and indexes.
9. Optional. If you must save a record of the modifications, then click Export.

The Case Insensitivity Wizard writes the modifications to a text file.

10. Click Finish.
    The Case Insensitivity Wizard configures the columns in the Siebel repository to support a CIAI query.
11. In the Tables list, click Apply/DDL.
12. Test and then deliver your Workspace.

Input File You Can Use with the Case Insensitivity Wizard

The Case Insensitivity Wizard can accept a comma-delimited (.csv) file as input. Each line in the file is one record that defines one column that the wizard configures for a CIAI query. Oracle provides a recommended input file. The input files include a csv file extension and are located in the `objects` subdirectory of your Siebel Tools installation. These files list columns that Siebel CRM frequently uses for queries. You can edit these files or create new input files.

You must use the following format for each record:

```
TABLE_NAME,COLUMN_NAME,Method,Index Strategy,Operation
```

where:

- *TABLE_NAME* and *COLUMN_NAME* are required.

For example:

```
S_CONTACT,EMAIL_ADDR,Database,Copy All,On
```

The Case Insensitivity Wizard inserts the default value for any optional variable that you omit.

If you omit a variable from a record, then you must provide a delimiting comma that represents the placeholder for the variable. In the following example, the Index Strategy variable is omitted:

```
S_CONTACT,EMAIL_ADDR,Database,,On
```

The Case Insensitivity Wizard does not perform special handling for a denormalized column. To configure a CIAI query on denormalized columns, you must include them in an input file.

For more information, see *Variables You Can Use with the Case Insensitivity Wizard*.

Using the Case Insensitivity Wizard on a Table Column

To run the Case Insensitivity Wizard, you can manually choose a table column. The wizard uses configuration defaults to configure the column. To modify the configuration options, you can export the configuration strings to a text file, edit them, and then use the edited file as an input file to run the wizard.

To use the Case Insensitivity Wizard on a table column

1. In Siebel Tools, open a repository.
   For more information, see *Choosing the Correct Repository when Running the Case Insensitivity Wizard*.
2. In the Object Explorer, click Table.
   As an alternative, you can display the Object Explorer in Flat mode, click Column, and then locate the column in the Columns list.
3. In the Tables list, choose the table you must modify.

4. In the Object Explorer, expand the Table tree, and then click Column.
5. In the Columns list, right-click the column you must modify, and then click Case Insensitivity.

   The Case Insensitivity Wizard does the following:

   - Validates the eligibility of the chosen column
   - Lists any column that contains an eligibility error in the Configure Case Insensitivity dialog box.

     If an error occurs, then you can export the error list to a text file, correct the error, and then restart the Case Insensitivity Wizard. To export the error list, click Export. If no error occurs and you continue, then the Case Insensitivity Wizard skips any column that contains an error.

     To run the Case Insensitivity Wizard for multiple columns, hold down the CTRL key while you choose each column in the Columns list.
6. Continue with Step 7 of the procedure in *Using the Case Insensitivity Wizard on a Table*.

Using the Case Insensitivity Wizard on Columns That Do Not Contain an Index

The Case Insensitivity Wizard defines a CIAI column and index only on a column that already includes an index. You can run the Case Insensitivity Wizard for a column that does not include an index but that meets all the other eligibility criteria. This mode modifies the Default Insensitivity property from None to DB Case & Accent. In a query, Siebel CRM converts column values to uppercase before it does the comparison. This work allows a search to be case and accent insensitive.

For example, in the S_CONTACT table, assume the LAST_NAME column does not include an index. You run the Case Insensitivity Wizard to set the Default Insensitivity property to DB Case & Accent. If you query for the name Smith, or any case variant such as SMITH or smiTH, then the object manager uses a query similar to the following:

```
SELECT column list FROM S_CONTACT
WHERE UPPER(LAST_NAME) LIKE UPPER(Smith)
```

To use the Case Insensitivity Wizard on columns that do not contain an index

1. In Siebel Tools, open a repository.

   For more information, see *Choosing the Correct Repository when Running the Case Insensitivity Wizard*.
2. Click the Tools menu, click Utilities, and then click the Case Insensitivity menu item.
3. Choose the Enable for All Unindexed Columns option, and then click Next.

   The Case Insensitivity Wizard does the following:

   - Locates unindexed columns that meet CIAI eligibility criteria
   - Displays a list of tables that you must lock
4. Click Export to export the list of tables to a text file, then exit the Case Insensitivity Wizard.
5. Open the text file in a text editor.
6. In Siebel Tools, lock all tables that the text file lists.
7. Start the Case Insensitivity Wizard again, choose the Enable for All Unindexed Columns option, and then click Next.

   The Case Insensitivity Wizard does the following:

- Locates unindexed columns
- Displays a list that describes how the unindexed columns are configured

8. Verify that method is Database and Index Strategy is None for all columns.

   If the index strategy is None, then the Case Insensitivity Wizard does not create a CIAI column or index.
9. Click Next.

   The Case Insensitivity Wizard displays a list that describes the Siebel repository modifications it will make.
10. Verify that the Default Insensitivity property is DB Case & Accent for all columns.
11. Click Finish.

    The Case Insensitivity Wizard makes the modifications to the Siebel repository.

Using the Case Insensitivity Wizard to Do Various CIAI Configuration Tasks

You can use the Case Insensitivity Wizard to do various CIAI configuration tasks.

To use the Case Insensitivity Wizard to do various CIAI configuration tasks

- Use the following table to determine how to run the wizard.

Work You Must Perform	How to Run the Case Insensitivity Wizard
Define new columns to support a CIAI query.	Use an input file or choose files manually.
Deactivate CIAI for defined columns.	Use an input file that specifies the columns. For each column, set Operation to Off.
Modify the Default Insensitivity property from None to DB Case & Accent for eligible columns without indexes.	Choose the Tools menu, Utilities, and then the Case Insensitivity menu item. Choose the Enable for All Unindexed Columns option.
Modify the method from Force Case to Database for columns that are already defined.	Use an input file or choose files manually.
Modify the method from Database to Force Case for columns that are already defined.	Use an input file that specifies the columns. For each column, set Operation to Off. This configuration deactivates the CIAI column and CIAI indexes. Verify that the Force Case property is set for base columns.

Work You Must Perform	How to Run the Case Insensitivity Wizard
Modify the index strategy from Single to Copy All for columns that are already defined.	Use an input file that specifies to modify the index strategy from Single to Copy All.
Modify the index strategy from Copy All to Single for columns that are already defined.	Do the following: o Run the Case Insensitivity Wizard, using an input file that specifies the columns. For each column, set Operation to Off. This configuration deactivates the CIAI column and CIAI indexes. o Next, run the Case Insensitivity Wizard on the same base columns with method set to Database and Index Strategy set to Single. This configuration activates the index on the CIAI columns.

Using the Case Insensitivity Wizard to Deactivate CIAI Configuration

You can use the operation variable to deactivate CIAI configuration of columns. The operation variable determines if the columns and indexes that the Case Insensitivity Wizard creates are or are not active. The available values are On or Off. The default value is On.

To use the Case Insensitivity Wizard to deactivate CIAI configuration

- Run the Case Insensitivity Wizard against a column. Set the Operation to Off.

  The wizard does the following work:

  - Deactivates the CIAI index on the CIAI column.
  - Sets the related CIAI indexes to inactive for indexes that reference the base column.
  - Does not deactivate CIAI columns that reference the base column. You must manually set inactive to TRUE for each of these columns.
  - Does not delete CIAI columns or CIAI indexes in the Siebel repository.

You cannot manually modify the Default Insensitivity property of a predefined Siebel column to None. You must run the Case Insensitivity Wizard for this column with Operation set to Off. This column is the last column in the csv file.

Choosing the Correct Repository when Running the Case Insensitivity Wizard

This topic describes how to choose the correct repository.

To choose the correct repository when running the Case Insensitivity Wizard

1. If you are not upgrading a development environment, then do the following:
   a. In the development environment, run the Case Insensitivity Wizard on the Siebel Repository.
   b. Create another schema.ddl file, and then use it to update your test and production environments.
2. If you are upgrading a development environment, then run the Case Insensitivity wizard on the New Customer Repository.

   Later in the upgrade process, Siebel CRM renames this repository to Siebel Repository. To revise the columns you configured for a case insensitive query, you can run the Case Insensitivity Wizard after an upgrade is complete. For more information, see *Siebel Database Upgrade Guide* .

Limiting the Length of Schema Object Names Manually

You can manually limit the length of schema object names to 18 characters.

To limit the length of schema object names manually

1. In Siebel Tools, click the View menu, and then click Options.
2. Choose the Database tab of the Development Tools Options dialog box.
3. Make sure the Limit Schema Object Names to 18 Characters option contains a check mark, and then click OK.

Other Configurations to Set Case Sensitivity

This topic describes other configurations you can use to set case sensitivity.

Using CIAI with Siebel Remote

This topic describes how to configure Siebel CRM to use CIAI with a Siebel Remote client.

To use CIAI with Siebel Remote

1. Extract the local database.

   For information about how to extract the local database, see *Siebel Remote and Replication Manager Administration Guide* .
2. Access the remote client, and then use a text editor to open the application configuration file.

   For example, open uagent.cfg for Siebel Call Center.
3. In the Siebel section, set the following parameter:

   ```
   WATCOM_CIAI_FUNC = CIAI_UPPER
   ```
4. Save, and then close the application configuration file.

Using the ForceCase Property of a Field to Force Case Sensitivity

You can use the ForceCase property of a business component field to force case sensitivity. If the user steps off the record, then Siebel CRM applies the case that is defined in the ForceCase property. If the ForceCase property contains no value, then the text remains in the same case that the user uses when this user enters the text.

To use the ForceCase property of a field to force case sensitivity

- Set the ForceCase property of a business component field to one of the following values:
  - Upper to force the text to all uppercase
  - Lower to force the text to all lowercase
  - FirstUpper to force the first letter of each word to uppercase

Improving the Performance of a Siebel Application

This topic describes how you can improve the performance of a Siebel application. It includes the following information:

- *Preventing a Secondary Query on a Foreign Key*
- *Defining the Primary ID Field of a Multi-Value Link*
- *Modifying Custom Search Specifications*
- *Using Declarative Configuration to Enable a Button*
- *Improving Performance When Using Applet Toggles*
- *Deactivating Unused Screens*
- *Considering Factors That Affect Chart Performance*
- *Considering Factors That Affect MLOV Performance*

For more information about performance tuning, see also *Siebel Performance Tuning Guide* .

Preventing a Secondary Query on a Foreign Key

Siebel CRM typically configures a multi-value link with a primary join. In this situation, the foreign key that this join uses to identify the primary record might not find the primary. For example, this problem can occur in the following situations:

- The primary record is deleted from the multi-value group.
- The multi-value group is new and does not contain any records.

You can define the multi-value link to update the primary foreign key to a value of NULL, or to a special value of NoMatchRowId, depending on your requirements. The purpose of the NoMatchRowId value is to prevent a secondary query on a foreign key value that failed. This configuration improves performance in the same way that a primary join improves performance.

For more information, see *Configuring the Auto Primary Property of a Multi-Value Link* and *How Siebel CRM Creates a Multi-Value Group*.

To prevent a secondary query on a foreign key

1. If most parent records in the multi-value group do not include any child records, then do not set the Check No Match property of the multi-value link to TRUE.

   In this situation, if you set Check No Match to TRUE, then performance is almost as slow as not having a primary join at all, and you might encounter serious negative performance consequences. In most situations, you must not set Check No Match to TRUE.

   For more information, see *Configuring the Check No Match Property of a Multi-Value Link*.
2. If a user can add a record to the multi-value group other than through the multi-value group, then consider setting Check No Match to TRUE.

   Consider the following examples:

   - If a user can add a record to the same multi-value group through a multi-value link that is defined on the Contact business component in addition to a multi-value link that is defined on the Account business component
   - If Enterprise Integration Manager adds records to the child business component
3. If you set CheckNoMatch to TRUE, then set the Use Primary Join property of the multi-value link to TRUE.

   If the CheckNoMatch property is TRUE, and if the Use Primary Join is FALSE, then to find the child records, Siebel CRM always does the secondary query. For more information, see *Configuring the Use Primary Join Property of a Multi-Value Link*.

Configuring the Check No Match Property of a Multi-Value Link

If you set the Check No Match property of the multi-value link to TRUE, then Siebel CRM does the following work:

- If it encounters a parent record where the value of the primary foreign key is NoMatchRowId, then it does not perform a secondary query because a value of NoMatchRowId indicates that no child records exist in the multi-value group.
- If it encounters a parent record where the primary foreign key is empty, NULL, or invalid, then it does a secondary query to determine whether child records exist in the multi-value group:
  - If it finds no child records, then it sets the Primary ID Field to NoMatchRowId. For more information, see *About the Primary ID Field*.
  - If the Auto Primary property of the multi-value link is DEFAULT, and if the secondary query locates a matching detail record, then it updates the foreign key with the row ID of the record that the query located.
  - If the Auto Primary property of the multi-value link is NONE, and if the secondary query does not locate a matching detail record, then it leaves the current value intact.

Defining the Primary ID Field of a Multi-Value Link

If you define a primary ID, then Siebel CRM can get one primary record more quickly from the master business component through a join than it can get all records through a subquery. The primary ID converts a one-to-many relationship into a one-to-one relationship. This configuration simplifies retrieval of the row from a query with subqueries to a simple join query. This configuration improves performance, especially if the user scrolls through the records of a list applet that displays the parent.

For example, in the Account business component the Primary ID Field property of the Business Address multi-value link is Primary Address Id. The Account Address Mvg Applet displays the corresponding multi-value group. Siebel CRM stores the row ID for the primary record in the Primary Address Id field in the account record. The Primary check mark in the list column identifies the primary. Each time Siebel CRM displays a different account record, the multi-value fields for the Address load only the values from the record of the primary Business Address. It is not necessary to query the Business Address business component for multiple rows. This configuration can significantly improve performance, especially in a list applet.

Most predefined multi-value links designate a primary record. A multi-value link that does not designate a primary record uses the first record that Siebel CRM gets from the child business component. The link and multi-value link include a set of properties that you can define to configure Siebel CRM to get the record Id of the first record that displays records from the child table each time it modifies the parent record. You can create a primary field for a one-to-many or a many-to-many relationship.

In a multi-value group applet, the list column that displays the check mark indicates the primary or nonprimary status of each record. This list column gets data for the column from the SSA Primary Field system field. Siebel Tools does not display this field in the Object Explorer or Object List Editor, but you can reference it from a list column.

For more information, see the following topics:

- *About Multi-Value Links*
- *How Siebel CRM Creates a Multi-Value Group*
- *Creating Multi-Value Groups and Multi-Value Group Applets*
- *System Fields of a Business Component*

To define the Primary ID field of a multi-value link

1. Create a Primary Id column.
2. Create a new field that references the Primary Id column you created in Step 1.
3. In a multi-value link, set the Primary Id Field property to the field you created in Step 2.

   The Primary Id Field property specifies the name of the field in the parent business component that contains the row IDs that reference primary records in the child business component.

   Do not display the Primary ID Field in the Siebel client. If you display the Primary ID Field in an editable control or list column on an applet, then the multi-value group applet does not update the primary. If you must display the Primary ID Field in the Siebel client, such as for testing, then use a read-only control or list column.
4. Set the Use Primary Join property of the multi-value link to TRUE.

   For more information, see *Configuring the Use Primary Join Property of a Multi-Value Link*.
5. Set the Auto Primary property.

   For more information, see *Configuring the Auto Primary Property of a Multi-Value Link*.

Configuring the Use Primary Join Property of a Multi-Value Link

The Use Primary Join property of a multi-value link enables the primary join feature. If you set Use Primary Join to TRUE, then Siebel CRM gets the primary child record for each parent record through a join with the Primary ID Field. If you set Use Primary Join to FALSE, then Siebel CRM queries the child table each time it modifies a parent.

Configuring the Auto Primary Property of a Multi-Value Link

The Auto Primary property determines how Siebel CRM enters row IDs into the Primary ID Field. It uses the Primary system list column in the multi-value group applet. The user can manually choose the primary. You can set the Auto Primary property to one of the following values:

- **DEFAULT.** The first record becomes the primary.
- **SELECTED.** If the user views the multi-value group applet, and then exits, then the highlighted record becomes the primary. For more information, see the *About the Selected Option of the Auto Primary Property* section of this topic.
- **NONE.** The user must use the pick map to include the Primary Owner Id manually. For more information, see the *Configuring Siebel CRM to Not Override the Pick Map Value section* of this topic.

About the Selected Option of the Auto Primary Property

The SELECTED option only applies if several multi-value links reference the same child business component. For example, with the predefined Bill To Business Address multi-value link and the predefined Ship To Business Address multi-value link. These multi-value links exist in the Order business component and in the Account business component. In this example, if a primary is not set for the Bill To address, then when Siebel CRM does a separate query to bring back all addresses that it associates with the account or order, it determines if one of the addresses is or is not already chosen as the primary for the Ship To address. If it is, then Siebel CRM sets that address as the primary for Bill To address.

How the Auto Primary Property Affects the Read-Only Status of the Primary ID Field

If the Auto Primary property of a multi-value link contains a value of SELECTED, then defining a read-only property at the applet level does not force the SSA Primary Field to be read-only. If the destination business component of the multi-value link is read-only, then Siebel CRM might display an error message that is similar to the following:

```
This operation is not available for a read-only field 'SSA Primary Field'
```

This errors occurs because Siebel CRM updates the Primary ID Field through the SSA Primary Field system field, which is part of the destination business component. If this business component is read-only, then the Primary ID Field is read-only and Siebel CRM cannot update it. For more information, see *System Fields of a Business Component*.

Configuring Siebel CRM to Not Override the Pick Map Value

If you set the Auto Primary property to DEFAULT in some configurations, then Siebel CRM might populate the Primary Owner Id with No Match Row Id. To avoid this problem, it is recommended that you set the Auto Primary property to NONE. For example, assume that you do the following:

- In Siebel Tools, create the following multivalue field in the Action business component:
  - **Name.** Owned By Division.
  - **Multivalue Link.** Employee.
  - **Field Division.**

  The Field in the Employee business component must be a multivalue field.
- Expose the new multivalue field in the Contact Activity List Applet applet.
- Log in to the Siebel CRM client as SADMIN, click Contacts, Contacts List, and then click the Last Name of any contact.
- Create a new record in the Activities list. Note that Siebel CRM sets the value of the Employee field to SADMIN. This field is the Owned By multivalue field.

- Click Query, and then click Go. Note that the Employee field is empty.
- Open the Employee multivalue group applet. Note that one record exists for Siebel Administrator (SADMIN) but it is not the primary, and the Primary Owner Id field contains No Match Row Id.

In this situation, Siebel CRM incorrectly populates the Primary Owner Id with No Match Row Id. To avoid this problem, you can set the Auto Primary property to NONE instead of DEFAULT for the Employee multivalue list.

Modifying Custom Search Specifications

You can improve performance by modifying custom search specifications.

To modify custom search specifications

- Examine your custom search specifications:
  - Avoid a field in the search specification that references a join.

    If a business component defines an outer join, and if the Search Specification property of a business component includes a joined field that uses this outer join, then the SQL code modifies the inner join for performance reasons.
  - Avoid using a business component field that is calculated.
  - Avoid using a NOT or OR operator in the search specification. These operators force the Siebel database to run a full table scan that can adversely affect performance.

For more information, see *Siebel Performance Tuning Guide* .

Using Declarative Configuration to Enable a Button

To enable a button, you can script the WebApplet_PreCanInvokeMethod to set the CanInvoke parameter of an event to TRUE. It is recommended you use declarative configuration to enable a button because a performance cost occurs when scripting the WebApplet_PreCanInvokeMethod event.

To use declarative configuration to enable a button

- Use declarative configuration rather than scripting to enable a button.

  You can configure the CanInvokeMethod applet user property to enable a button. For more information on using a minibutton to call a custom method, see *Siebel Object Interfaces Reference* .

Improving Performance When Using Applet Toggles

Siebel CRM loads all available applet toggles each time the user navigates to an applet. It cannot cache an applet toggle, so a negative affect on performance might occur. For more information, see *Siebel Performance Tuning Guide* .

To improve performance when using applet toggles

- Make sure no applet toggles are defined that Siebel CRM does not use or that are not necessary.

Deactivating Unused Screens

This topic describes how to deactivate predefined screens that Siebel CRM does not use in your implementation.

Note: Responsibilities, Views, and their relationship can be Workspace enabled in your Development environment. If you have Workspace enabled Responsibilities, the changes below must be done in an editable Workspace while in your Development environment. If you are editing Responsibilities, Views, and their relationship in your Runtime Repository environment, there is no need for an editable Workspace.

To deactivate unused screens

- Use one of the following configurations:
    - Log in to the Siebel client, and then use the Responsibility Administration Screen to disassociate all views of the unused screen from the responsibilities that your organization uses. This configuration does not require you to deploy changes to the Siebel runtime repository. It provides an easy upgrade path if you decide to use the screen or views later. At that time, no configuration or software upgrade is required. You only need to reassign the views to the relevant responsibility.
    - Use Siebel Tools or Web Tools to deactivate the screen. This configuration requires you to deploy changes to the Siebel runtime repository. When you deliver your Workspace, Siebel CRM does not include the inactive screen in the Siebel runtime repository.
    - Use Siebel Tools or Web Tools to deactivate the screen. This configuration requires you to deploy changesto the Siebel runtime repository. When you deliver your Workspace, Siebel CRM does not include the inactive screen in the Siebel runtime repository.

Considering Factors That Affect Chart Performance

When a chart traverses records in the business component, Siebel CRM monitors the progress in a display at the lower quadrant of the window. Traversing all records of a business component might require significant time, so a chart is not appropriate for a data set that contains more than one thousand records.

When you design your implementation, consider how the following factors affect the performance of a chart in Siebel CRM:

- The number of records in the business component
- If the chart must or must not search a multi-value group to get data
- If a data point field is or is not defined
- If the data point field is a currency field, then consider the number of records whose currency is not the functional currency
- The processor, operating system, and database system you use

Considering Factors That Affect MLOV Performance

A custom MLOV can affect performance especially if the field that the list references is part of a search or sort. If you configure an MLOV, then you must consider and verify performance qualities. For more information, see *Siebel Performance Tuning Guide* .

23 Mapping a Custom Table to an Interface Table for Siebel EIM

Mapping a Custom Table to an Interface Table for Siebel EIM

This chapter describes how to map a custom table to an interface table for Siebel EIM. It includes the following topics:

- *Overview of Using Siebel EIM for Bulk Import and Export of Data*
- *Mapping a Custom Table to an Interface Table*

Overview of Using Siebel EIM for Bulk Import and Export of Data

This topic describes an overview of using Siebel Enterprise Integration Manager for bulk import and export of data. It contains the following information:

- *About Interface Tables*
- *Object Types That Enterprise Integration Manager Uses*

Note: For information about how to use Siebel EIM, see *Siebel Enterprise Integration Manager Administration Guide* .

Related Topics

How the S_Party Table Controls Access

Guidelines for Configuring a Foreign Key That Affects Enterprise Integration Manager

Adding an Extension Column to a Base Table

Guidelines for Using Enterprise Integration Manager with an MLOV

About Interface Tables

Siebel Enterprise Integration Manager (EIM) is a server component in the Siebel EAI component group that uses interface tables to transfer data between the Siebel database and other corporate data sources.

An *interface table* is an intermediate database table that provides a staging area between the Siebel database and other databases. It includes the following qualities:

- A Siebel administrator uses it to perform bulk imports, exports, updates, and deletes.
- The name of an interface table begin with the EIM_ prefix.

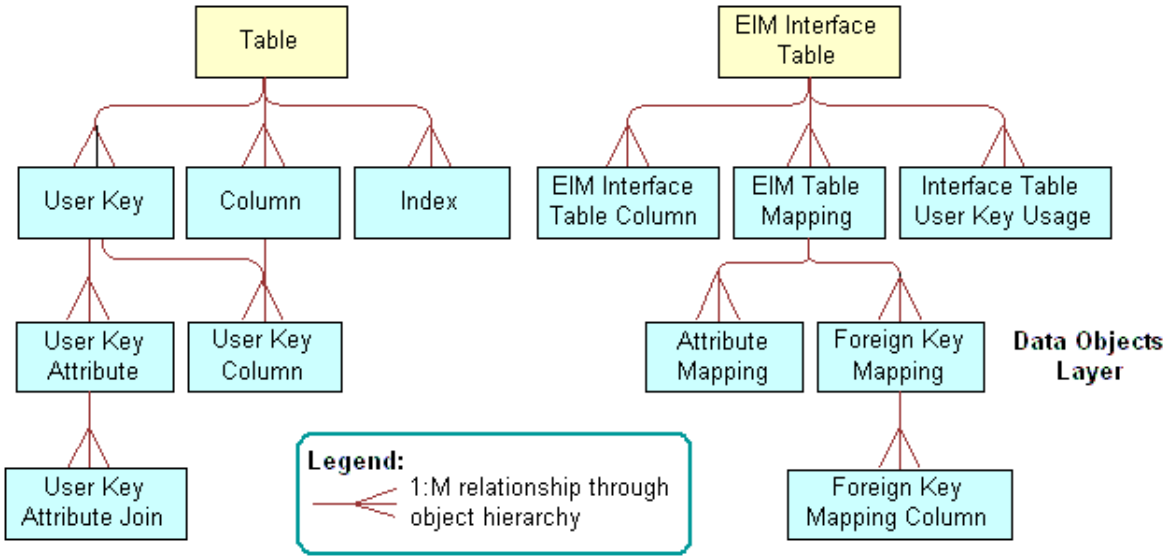
- The Type property of an interface table is set to Interface.
- A database administrator typically uses a third-party tool to enter values in an interface table. SQL Loader is an example of a third-party tool.

To use EIM to enter values in custom extension tables and extension columns, you create mappings between the new columns and EIM interface tables. You use the EIM Table Mapping Wizard to create these mappings. For more information, see *Mapping a Custom Table to an Interface Table*.

Object Types That Enterprise Integration Manager Uses

The following image shows the objects and relationships that EIM uses, which are as follows:

- **Tables:** There is a 1:M relationship (through object hierarchy) between:
  a. Table and each of the following: User Key, Column, Index.
  b. User Key and each of the following: User Key Attribute, User Key Column.
  c. User Key Attribute and User Key Attribute Join.
- **EIM Interface Tables:** There is a 1:M relationship (through object hierarchy) between:
  a. EIM Interface Table and each of the following: EIM Interface Table Column, EIM Table Mapping, Interface Table User Key Usage.
  b. EIM Table Mapping and each of the following: Attribute Mapping, Foreign Key Mapping.
  c. Foreign Key Mapping and Foreign Key Mapping Column.



EIM Interface Table Object Type

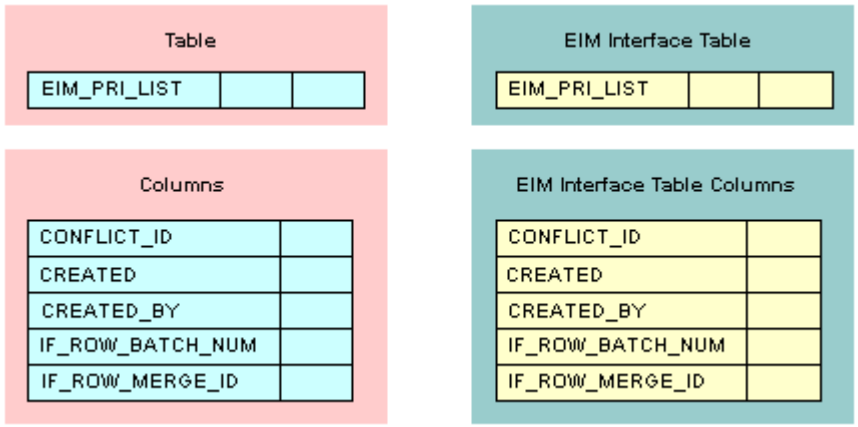
An *EIM interface table* is an object that provides an alternative representation of a table. It includes some of the same properties as a table plus other properties of an interface table.

EIM Interface Table Column Object Type

An *EIM interface table column* is an object that provides an alternative representation of a column. It contains all the properties of a column in addition to some properties that are specific to EIM.

The following image shows how the child columns of an interface table are the same as the child columns of a table. The EIM_PRI_LST price list interface table is an example.

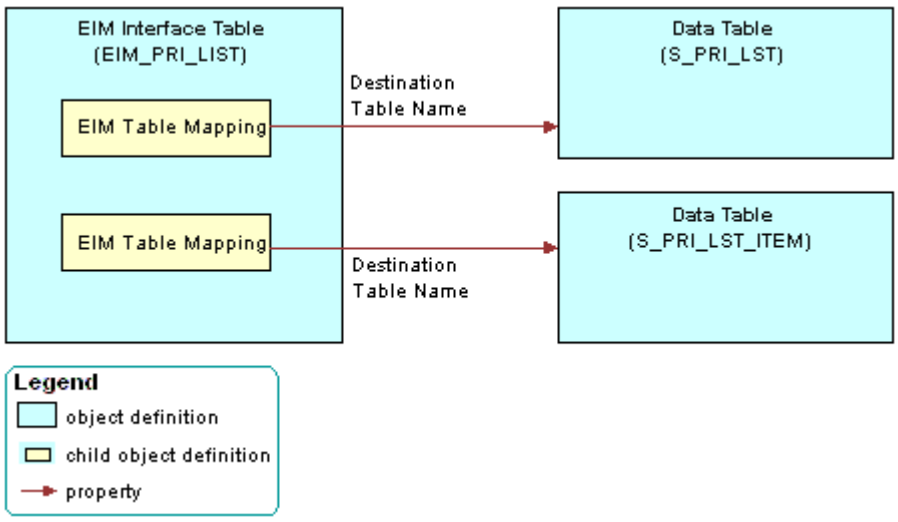
- The EIM_PRI_LIST *interface* table has the following columns: CONFLICT_ID, CREATED, CREATED_BY, IF_ROW_BATCH_NUM, AND IF_ROW_MERGE_ID
- The EIM_PRI_LIST table has the same columns.



EIM Table Mapping Object Type

An *EIM table mapping* is an object that references a data table that the parent EIM interface table object definition updates. One EIM interface table can update one or more data tables.

The following image shows how the Destination Table property of each EIM table mapping object identifies the name of the data table to update (that is, the S_PRI_LST data table and S_PRI_LST_ITEM data table in this example).



Interface Table User Key Usage Object Type

An *interface table user key usage* is an object that provides support for alternative user keys for base tables. It defines the use of a nontraditional user key for a base table that are specific to an interface table.

CAUTION: Do not modify the object definition of an interface table user key usage. Any modification can adversely affect performance and operation.

Attribute Mapping Object Type

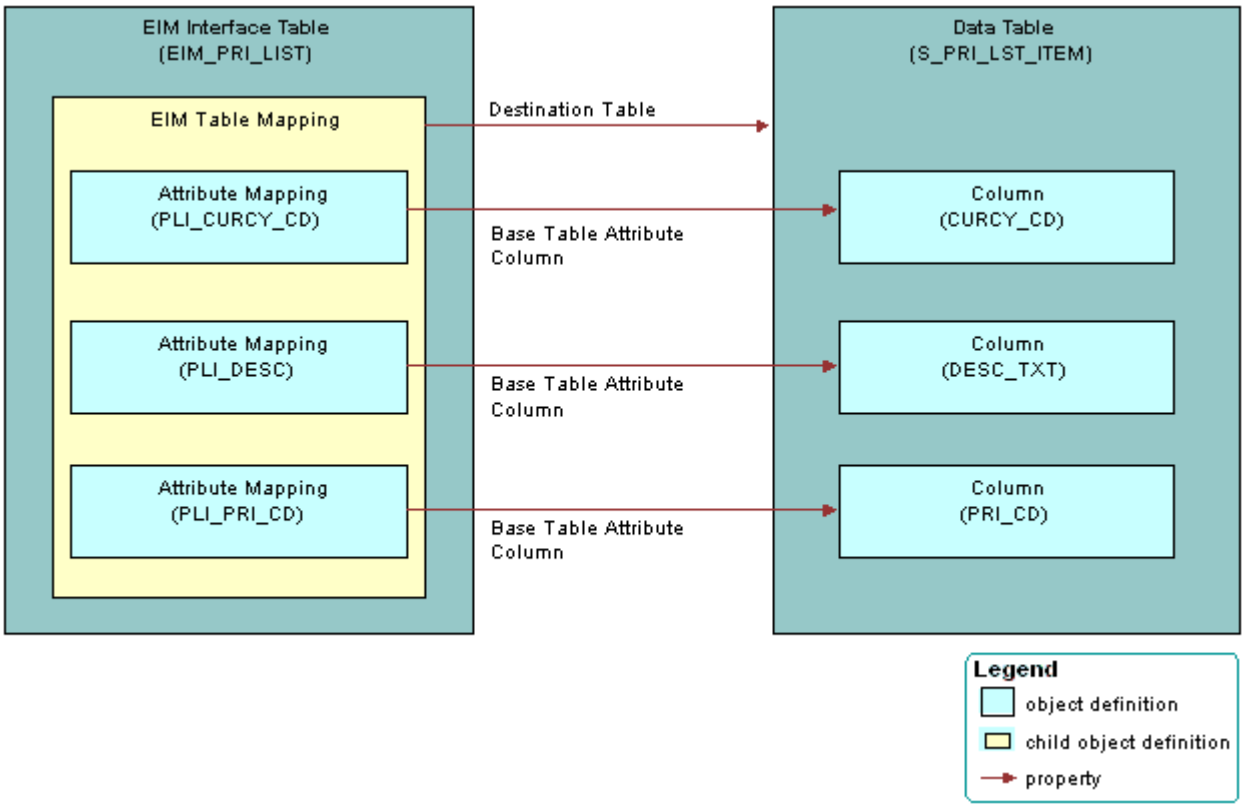
An *attribute mapping* is an object that identifies a column in a data table that EIM updates. This column resides in the destination table that is defined in the parent EIM table mapping. An attribute mapping includes the following properties:

- **Interface Table Data Column.** Identifies the column in the interface table that supplies the data.
- **Base Table Attribute Column.** Identifies the column in the destination table that receives the data.

If you add an extension column to a table, and if an interface table must provide data to the extension table, then you must add a corresponding attribute mapping.

The following table and image shows an example of how an EIM table mapping references a data table.

EIM_PRI_LIST Interface Table Attribute Maps to:	S_PRI_LIST_ITEM Data Table Column
PLI_CURCY_CD	CURCY_CD
PLI_DESC	DESC_TXT
PLI_PRI_CD	PRI_CD



EIM Explicit Primary Mapping Object Type

An *EIM Explicit Primary Mapping* specifies a column to be set as the Primary ID record for a multi-value link (MVL). For more information about MVLs and primaries, see *About Multi-Value Links*.

The EIM Explicit Primary Mapping object type contains these attributes:

- **Name.** This should be a concatenation of the destination table name and primary column name, separated by a period.
- **Base Table Primary ID Column Name.** Identifies the destination column to store the primary in.
- **Inf Tbl Exp Primary Flg Col Name.** Identifies the interface table column that indicates whether a particular record should be the primary.

Configure an EIM Table to Be Able to Set a Primary ID for a Multi-Value Link

To configure an EIM table to be able to set a Primary ID for a MVL, use these steps:

1. Validate that there's a primary field for the MVL defined on the business component. For example, Account has an MVL to Position and the primary is stored in the Primary Position ID field.
2. By inspecting the primary field, identify the underlying column where the primary is stored – in this example, S_ORG_EXT.PR_POSTN_ID.
3. On the EIM table, ensure that there's a column available to indicate which interface table reference will be the primary. If required, create a new column.
   - The column type should be Character(1), as it will have a value of Y or N.
   - The default should be N.
   - **Do not** create an Attribute Mapping for this column.
   - **Note:** For the Account Position example, the column already exists and is named ACC_PR_POSTN.
4. Create an EIM Explicit Primary Mapping as follows:

Attribute	Description	Sample Value
Name	Target table and column that you're setting	S_ORG_EXT.PR_POSTN_ID
Base Table Primary ID Column Name	Name of column supporting the Primary ID field on the target business component	PR_POSTN_ID
Inf Tbl Exp Primary Flg Col Name	Source column in the interface table that indicates whether a given record is the primary	ACC_PR_POSTN

To use the EIM Explicit Primary Mapping to set a primary, you'll need to determine which record in your EIM batch represents the primary for each set of records with the same user keys.

To continue the Account/Position example: If you were to have an account with three positions, you'd need to load your EIM interface table with three rows that would have identical account name and location. This ensures that only one account is created, but different position information exists on each record.

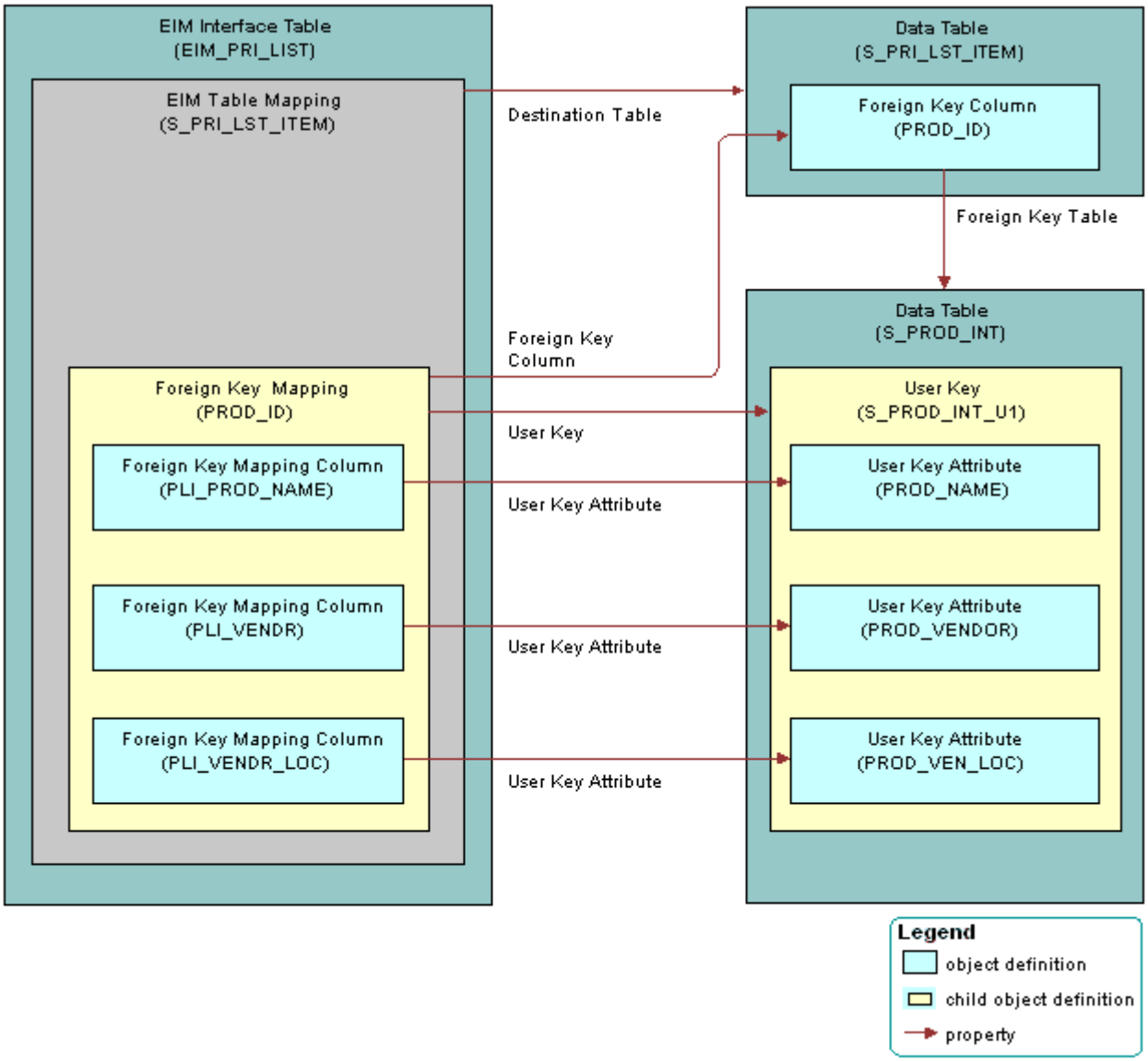
To make one the primary, set the value to Y for the Inf Tbl Exp Primary Flg Col Name attribute above, such as ACC_PR_POSTN. After running the EIM job, that position will be marked as the primary for that account.

CAUTION: Do not mark multiple records as the primary for the same logical target record, as results will be unpredictable.

Foreign Key Mapping Object Type

A *foreign key mapping* is an object that identifies a foreign key column in the destination table. EIM pulls data from an interface table and enters it into this foreign key column. EIM stores a foreign key as a numeric row ID value in a data table. To use data from an interface table in a foreign key, you must map the interface column to a combination of user key columns in the destination table rather than directly to the foreign key column.

The following image shows an example of how a foreign key map references a data table. To access the row, Siebel CRM uses a combination of attribute columns in the destination table of the foreign key. EIM gets the foreign key value from that row. A foreign key mapping is not a one-to-one column mapping from an interface table to a destination table. The numeric foreign key does not exist in the interface table, so you cannot map it.



Foreign Key Mapping Column Object Type

A *foreign key mapping column* is an object that does the following:

- To locate rows in the table that the foreign key references, identifies one of the attribute columns EIM uses. EIM combines values from the user key columns to form a key that uniquely identifies rows in that table.
- Identifies the user key columns so EIM can get foreign key values during an import or export.

User Key Object Type

A *user key* is an object that provides a set of attribute columns and related information that specifies how EIM can access the table rows. For more information, see *How a User Key Creates a Unique Set of Values*.

User Key Column Object Type

A *user key column* is an object can be an attribute or a foreign key. In most situations user key columns constitute the columns in the user key index with the exception of the CONFLICT_ID column. A user key index typically includes a _U1 suffix.

User Key Attribute Object Type

A *user key attribute* is an object that the parent user key specifies in the set of attribute columns that collectively identifies rows in the grandparent table. The column name is defined in the Name property of the user key attribute.

User Key Attribute Join Object Type

A *user key attribute join* is an object that specifies a join operation that EIM can use to convert a user key attribute that is a foreign key to another table into attribute column values in that table.

For example, the S_PROD_INT products table includes the S_PROD_INT_U1 user key. This user key references the following columns:

- PROD_NAME
- PROD_VENDOR
- PROD_VEN_LOC

EIM gets the PROD_NAME column from the S_PROD_INT table. No join is required.

EIM must use a join to get the PROD_VENDOR and PROD_VEN_LOC columns from the S_ORG_EXT accounts table. EIM uses a join on VENDR_OU_ID, which is a foreign key from the S_PROD_INT table to the S_ORG_EXT table.

Mapping a Custom Table to an Interface Table

This topic describes how to map a custom table to an interface table. It includes the following information:

- *Mapping a Custom Table to an Interface Table Using the EIM Table Mapping Wizard*
- *Relations That the EIM Table Mapping Wizard Creates*
- *Guidelines for Using the EIM Table Mapping Wizard*
- *Starting the EIM Table Mapping Wizard for a Table That Does Not Use the Foreign Key*
- *Deactivating Instead of Deleting an EIM Attribute Mapping*

- *Modifying Data from NULL to No Match Row Id*

Mapping a Custom Table to an Interface Table Using the EIM Table Mapping Wizard

To map custom columns and tables to a predefined EIM interface table, you use the EIM Table Mapping Wizard.

To map a custom table to an interface table

1. Make sure the table you must map is the appropriate type, includes a user key attribute, and that Siebel CRM supports the mapping.

   For more information, see *Guidelines for Using the EIM Table Mapping Wizard*
2. In Siebel Tools, display all child object types of the EIM Interface Table object type.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
3. In the Object Explorer, click Table.
4. In the Tables list, locate the table that must reference an EIM table.

   This table is the primary table where EIM imports data from the predefined interface table.
5. Right-click the record, and then click EIM Table Mapping.

   Siebel Tools displays the Interface Table Mapping dialog box. It enters data into the Base Table name window of this dialog box. It gets this data from the table you located in Step 4.
6. In the *Enter Column Name Prefix* window, enter a prefix.

   Siebel Tools does the following:

   - If a prefix does not already exist for the EIM table, then Siebel Tools adds the new prefix that you enter for the EIM interface table columns that reference the table.
   - If a prefix already exists for the EIM table, then Siebel Tools uses the existing prefix.

   If you specify a prefix, then the EIM Table Mapping Wizard adds this prefix to the new columns and makes the column properties uneditable.
7. In the *Select the Interface Table* window, choose a value from the list, and then click Next.

   Siebel Tools constrains the list you use to choose the EIM interface table. It displays only the interface tables that include a foreign key relationship with your new custom Table. Siebel Tools sorts this list by EIM table name. If the Exist field of the interface table is Y, then the EIM table is already mapped to the base table. If you extend a predefined Siebel table, then a table with a Y in the Exist field is an ideal candidate for EIM mapping.
8. Click Finish to accept the configuration and create the EIM Interface Table object.

   Siebel Tools begins the mapping, which might take several minutes. For more information, see *Relations That the EIM Table Mapping Wizard Creates*.
9. To verify the mappings, do the following:

   a. In the Object Explorer, click EIM Interface Table.
   b. In the EIM Tables list, run a query for all modified records.

      When you run the query, make sure the Changed property contains a check mark. Leave all other properties empty.
   c. To verify the mapping, examine child objects.

d. Identify any new mappings that are not necessary.

10. If any new mappings are not necessary, then do the following:

    a. Deactivate the unnecessary mappings.
    b. Rename or delete the *ORACLE_HOME* `\bin\diccache.dat` file on the Siebel Server.
    c. Run the following query to review any more columns that Siebel Tools created:

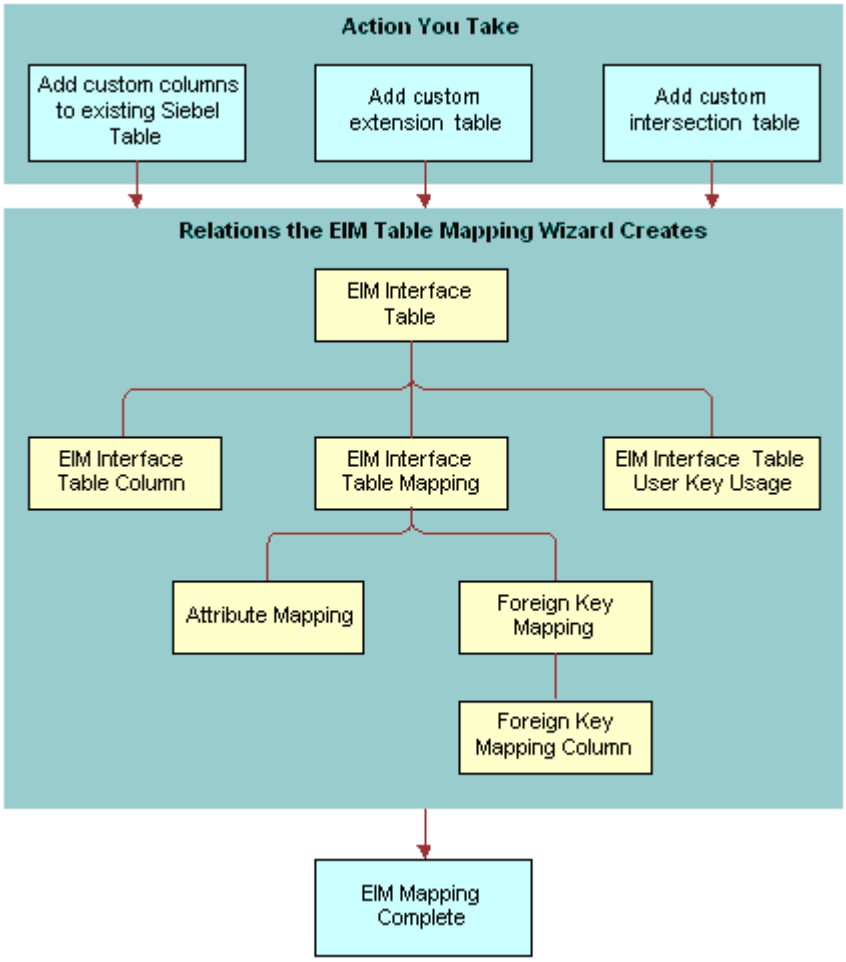
       `T_*`

Relations That the EIM Table Mapping Wizard Creates

The following figure describes the relations that exist between objects that the EIM Table Mapping Wizard creates. The wizard maps objects and adds child objects to the predefined EIM interface table object. The figure contains the following information:

1. **Actions You Take:** Add custom columns to existing Siebel table, Add custom extension table, Add custom intersection table.
2. **Relations that the EIM Table Mapping Wizard Creates:**
   - There is 1:1 relationship between EIM Interface Table and each of the following: EIM Interface Table Column, EIM Interface Table Mapping, EIM Interface Table User Key Usage.
   - There is a 1:1 relationship between EIM Interface Table Mapping and each of the following: Attribute Mapping, Foreign Key Mapping.
   - There is a 1:1 relationship between Foreign Key Mapping and Foreign Key Mapping Column.
3. EIM Mapping is now complete.

For more information about EIM objects that the wizard creates, see *Objects You Use with Enterprise Integration Manager*.



Guidelines for Using the EIM Table Mapping Wizard

If you use the EIM Table Mapping Wizard, then use the following guidelines:

- You must set the Type property for any table you use with the EIM Table Mapping Wizard to one of the following values:
  - Data (Public)
  - Data (Intersection)
  - Extension
  - Extension (Siebel)
- You cannot use the EIM Table Mapping Wizard with a custom table because no EIM table exists to choose in the EIM Table Mapping Wizard.
- You can map a single column in an interface table to multiple base tables or extension tables. Do not map multiple interface table columns to a single column in a target table because it can create ambiguity for EIM.
- EIM does not validate an interface table or a column definition. EIM validates a list of values against the lists of values that are defined for the base columns where the values are mapped.

Restrictions on Adding or Modifying EIM Mappings

The following table describes restrictions on adding or modifying EIM mappings.

From	To	Restriction
Interface table column	Base column	Supported if predefined mappings exist from the interface table to the data table.
Interface table extension column	Base column	Supported if no other mappings exist to the base column. Use with caution.
Interface table column	Extension table column	Supported if predefined mappings exist from the interface table to the base table of the extension table.
Interface table extension column	Extension table column	

Starting the EIM Table Mapping Wizard for a Table That Does Not Use the Foreign Key

To start the EIM Table Mapping Wizard for a Siebel base table that does not use the foreign key as part of the user key, you must create a temporary column, and then run the wizard.

To start the EIM Table Mapping Wizard for a table that does not use the foreign key

1. Create a temporary column. Use properties described in the following table.

Property	Value
Inactive	Contains a check mark.
User Key Sequence	<> NULL
Foreign Key Table Name	Choose the target table for the interface table. In many, but not all, situations, this table is the parent table of the temporary column.

2. Run the EIM Table Mapping Wizard.

   By creating the temporary column, The EIM Table Mapping Wizard lists predefined EIM interface tables that are already mapped to this table as the target or destination table. The wizard lists the EIM tables that Siebel CRM

maps to the tables that this table uses as a foreign key. The foreign key must be part of the Traditional U1 Index user key of this table.

For more information, see *Mapping a Custom Table to an Interface Table*.

3. After the EIM Table Mapping Wizard finishes, delete the column you created in Step 1.

Deactivating Instead of Deleting an EIM Attribute Mapping

Do not delete any attribute mapping. Instead, you can deactivate an attribute mapping if you no longer require it.

To deactivate instead of deleting an EIM attribute mapping

1. In Siebel Tools, display all child object types of the EIM Interface Table object type.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. In the Object Explorer, click EIM Interface Table.
3. In the EIM Interface Tables list, locate the table that contains the attribute mapping you must modify.
4. In the Object Explorer, expand the EIM Interface Table tree, and then click EIM Table Mapping.
5. In the EIM Table Mappings list, locate the table mapping that contains the attribute mapping you must modify.
6. In the Object Explorer, expand the EIM Table Mapping tree, and then click Attribute Mapping.
7. In the Attribute Mappings list, locate the attribute mapping you must modify.
8. Make sure the Inactive property contains a check mark.

Modifying Data from NULL to No Match Row Id

If a primary child column includes no match, then Siebel CRM labels the columns differently depending on how you load data:

- If you load data through EIM and a primary child column includes no match, then EIM labels the column with NULL.
- If you load data through the Siebel client and a primary child column includes no match, then Siebel CRM labels the column with No Match Row Id.

You must fix this problem.

To modify data from NULL to No Match Row Id

1. In the Siebel client, open the record set.
2. Manually step through each record that EIM created.

Siebel CRM replaces each instance of a NULL value with No Match Row Id.

Mapping a Table to an EIM Interface Table in Siebel Web Tools

In this example of mapping an EIM Interface Table to a Siebel Table, we use the S_CONTACT Table as the Siebel base Table and the EIM_CONTACT Table as the EIM Interface Table. The idea here is to map the columns from EIM_CONTACT to Columns in S_CONTACT so that EIM jobs can move data from EIM_CONTACT to S_CONTACT properly.

To Map a Table to an EIM Interface Table in Siebel Web Tools

1. Make sure the table you must map is the appropriate type, includes a user key attribute, and that Siebel CRM supports the mapping. For more information, see *Guidelines for Using the EIM Table Mapping Wizard*
2. Lock the Project for the EIM Interface Table involved. Once you have decided which table you are going to use for the EIM Interface Table, you must lock the Project for that Table.

   > **Note:** In later updates when Tables have been Workspace-enabled, Projects will be removed from the Repository as configuration work will be logically grouped by Workspaces. Once that happens, locking the Projects for Tables will not be possible or necessary.

3. In the Object Explorer click the Table object type.
4. Query for your EIM Interface Table.
   - **Example:** If you want to update the mappings between the EIM_CONTACT and S_CONTACT tables, you will query for the EIM_CONTACT Table.
5. Record the Project for the Table. In this case it is *EIM Person*.
6. In the Object Explorer click on the Projects object type.
7. Query for the Project associated to the EIM Interface Table. In this case, *EIM Person*.
8. Click the *Locked* checkbox on the row and step off the row to save the change.
9. Start the EIM Table Mapping Wizard by clicking on the magic wand icon.
10. Click the EIM Table Mapping icon to select it. You can tell it has been selected as it will now have a selection rectangle around it.
11. Click the Start button.

    > **Note:** This is a wizard like other wizards in Web Tools and contains a Navigation Toolbar at the top to navigate to the next and previous screens.

12. The first screen of the wizard contains three fields.

Field Name	Description
Select the base Table	This is the Table to which you want to move EIM data. In the example above the base Table is S_CONTACT because we will be moving data from EIM_CONTACT to S_CONTACT.
Enter Column name prefix	This is the prefix that will be used for the mappings so that you can identify the mapped columns. If a prefix has been used before for this mapping, it will be used again here and the field will both read only and populated with that previous prefix. If none has been used, you are free to enter up to ten (10) characters in the field.

Field Name	Description
Select the interface Table	This is the Table from which the data should be drawn. In this case it is EIM_CONTACT

Note: If you did not lock the Project for EIM Interface Table, when you select an EIM Interface Table, you will receive the error: The selected EIM table is not locked (SBL-DEV-62536)

13. The Next button for the wizard will only become enabled once both Tables have been chosen.
14. Once you have chosen both Tables and, optionally, add a column prefix, Click Next.
15. This will take you to the confirmation screen where you can verify your choices. If you have chosen the wrong table or entered a column prefix that needs adjusting, click the Previous button, make those changes and click Next. If you are satisfied click the Submit button to create the mappings in the Repository.
16. After the mappings are created the application navigates to the Summary screen and you can see all the mappings created in the Repository.
17. You can view the mappings under EIM Interface Table > EIM Table Mapping > Attribute Mappings by searching for the base Table you supplied to the Wizard. For more information, see *Relations That the EIM Table Mapping Wizard Creates*.

24 Configuring Dock Objects for Siebel Remote

Configuring Dock Objects for Siebel Remote

This chapter describes how to configure docking rules for Siebel Remote. It includes the following topics:

- *About Dock Objects*
- *Configuring Dock Objects*

About Dock Objects

This topic describes dock objects. includes the following information:

- *Dock Object Table*
- *Dock Object Visibility Rule*

Synchronization is the process that Siebel Remote performs to allow a Siebel Mobile Web Client to connect to a Siebel Server and exchange updated data and files. This client typically operates on a remote laptop that is not connected to the Siebel Server. To support remote computing, Siebel Remote allows field personnel to share current information with members of virtual teams of other remote and connected users across an organization.

A *dock object* is an object that is a logical grouping of tables that contain special schema structures that synchronize data between a server database and a local database in a coherent manner.

When Siebel CRM updates data on the Siebel Server, Siebel Remote synchronizes the local database when the remote user connects to the Siebel Server and does a synchronization. Siebel Remote only synchronizes the data that it must download to the local database. During the synchronization, Siebel Remote uploads any updates that exist in the local database to the Siebel Server. For more information, see *Siebel Remote and Replication Manager Administration Guide* .

Types of Dock Object

This topic describes the types of dock objects.

Private Dock Object

A *private dock object* is a type of dock object that routes data that cannot be configured. It makes sure that Siebel Remote never routes the rows in the dock objects to any remote user. During synchronization, Siebel Remote does the following:

- Uploads to the Siebel Server all records from tables that are part of a private dock object.
- Does not download any of these records to the remote user.

Enterprise Dock Object

An *enterprise dock object* is a type of dock object that distributes records without restriction. During synchronization, Siebel Remote uploads to the Siebel Server all records from tables that are part of an enterprise dock object. Only an administrator must update these tables. Remote users typically download these tables from the Siebel Server but they do not upload them to the server. To minimize synchronization time, you must use an enterprise dock object only with the following tables:

- Tables that contain small amounts of data.
- Tables that contain static data or data that Siebel CRM modifies only occasionally.

Limited Dock Object

A *limited dock object* is a type of dock object that contains individual rules that identify the records that Siebel Remote must download to a user. For more information, see *Dock Object Visibility Rule*.

Dock Object Table

A *dock object table* is an object that identifies the tables that contain records that Siebel Remote transfers. It is a child of the dock object. Foreign keys in the data objects layer relate all the tables that appear in the Dock Object Tables list in Siebel Tools to the primary table that is defined in the Primary Table Name property of the dock object. A dock object table can reference other tables in the Table Name property of the dock object table.

For example, the Primary Table Name property of the Opportunity dock object is set to S_OPTY. Dock object tables that are children of the Opportunity dock object reference other tables, such as the S_NOTE_OPTY table and the S_OPTY_REL table. This example describes how a dock object is a set of logical records. In this example, opportunities are the logical records. Each logical record is a collection of one or more physical database records that are spread across multiple tables.

Dock Object Visibility Rule

A *dock object visibility rule* is an object that Siebel Remote uses to determine whether it must download records to the user. It is a child of the dock object. If you use limited dock objects, then Siebel Remote downloads different data to different local databases depending on the following items:

- Employee identity of each local database owner
- Position
- Organization
- Visibility to data from different dock objects
- Relationship between dock objects

For more information, see *Siebel Remote and Replication Manager Administration Guide* .

Types of Dock Object Visibility Rules

The following table describes the values you can enter in the Type property of the dock object visibility rule when you use a limited dock object.

Type Property	Description
Calendar	Examines remote user access to the calendar of the user who owns the record. Applies only to calendar appointment records.
Category	Examines the category that is visible to the user.
Check Dock Object	Examines the relationship to another record that the user receives. For more information, see the section titled *Check Doc Object Visibility Rule* in this topic.
Employee	Examines the foreign key to the employee record of the remote user, and downloads data depending on the identity of the remote user. To find all candidate rules, Siebel Remote identifies all columns that Siebel CRM uses as foreign keys to the S_USER table, except CREATED_BY and LAST_UPD_BY.
Employee Manager	Examines the foreign key to the employee record of someone who directly reports to the remote user, and downloads data according to the employees who report to the remote user. To find all candidate rules, Siebel Remote identifies all columns that Siebel CRM uses as foreign keys to the S_USER table, except CREATED_BY and LAST_UPD_BY.
Organization	Examines the same business unit where the remote user resides.
Position	Examines the foreign key to the primary Position of the remote user, and downloads data according to the position of the remote user. To find all candidate rules, Siebel Remote identifies all columns that Siebel CRM uses as foreign keys to the S_POSTN table.
Position Manager	Examines the foreign key to the Position of someone who reports directly to the remote user, and then downloads data according to the positions that report to the remote user position. To find all candidate rules, Siebel Remote identifies all columns that Siebel CRM uses as foreign keys to the S_POSTN table.
SQL	Handles special exceptions through custom SQL.

Check Dock Object Visibility Rule

Siebel Remote uses the Check Dock Object visibility rule to download data depending on data from other dock objects. The relationship between data in other dock objects and the current dock object determines the records from the current dock object that Siebel Remote downloads.

The Foreign Key Table Name property of the table columns determines the candidate Check Dock Object rules that the Docking Wizard can find. For each foreign key, the following Check Dock Object rules exist regardless of where the foreign key column resides:

- Rules that use the dock object as the destination dock object. To determine these rules, Siebel Remote uses the foreign keys on the primary table of one of the following objects:
  - **The current dock object.** To find this kind of rule, Siebel Remote uses an algorithm that finds all foreign key columns except columns that reference the S_USER table or the S_POSTN table. It finds these columns in the table of the current dock object. For these foreign key columns, the algorithm finds the foreign key table that these foreign key columns reference. The dock object of the foreign key table becomes the object for the Check Dock Object of the newly created Check Dock Object rule in the current dock object.
  - **Other dock objects.** To find this kind of rule, Siebel Remote uses an algorithm that finds all foreign key columns that reference the primary table of the current dock object, on any table that is part of a limited dock object. The algorithm adds the appropriate Check Dock Object visibility rules to these limited dock objects, with the current dock object being the object for the Check Dock Object.
- Rules that use this dock object as the Check Dock Object rules. To determine these rules, Siebel Remote uses the foreign keys on the primary table of one of the following objects:
  - The current dock object
  - Other dock objects

  The algorithm for these types of rules is similar to the algorithm for rules that use this dock object as the destination dock object. The main difference involves switching the source table or column and target table or column.

Example of a Dock Object Visibility Rule

The example in this topic describes how Siebel Remote compares the overall visibility strength of a set of dock object visibility rules to the strength of each dock object table. Siebel Remote then uses this comparison to determine the records that it downloads to the user.

A dock object visibility rule includes the Visibility Strength property. For most situations, the value for this property can be 0, 100, or any integer between 0 and 100. A visibility strength of 100 indicates full visibility. A visibility strength of 0 indicates no visibility. It is recommended that you use a value of 100 or less. If your configuration requires a higher value, then you can use any value up to 254.

To examine an example of a dock object visibility rule

1. In Siebel Tools, display the object type named Dock Object and all child objects of the Dock Object.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. In the Object Explorer, click Dock Object.
3. In the Dock Objects list, query the Name property for Opportunity.
4. In the Object Explorer, expand the Dock Object tree, and then click Dock Object Table.
5. Note the values in the Visibility Strength property.

   For example, note the following:
   - Visibility strength for the S_NOTE_OPTY table is 100.
   - Visibility strength for the S_OPTY table is 50.

6. In the Object Explorer, click Dock Object Visibility Rule.
7. Note the following values for the first record that Siebel Tools displays in the Dock Object Visibility Rules list.

Property	Value
Sequence	1
Visibility Strength	100
Comment	You are on the sales team of the Opportunity

8. Note the following values for the sixth record that Siebel Tools displays in the Dock Object Visibility Rules list.

Property	Value
Sequence	6
Visibility Strength	50
Comment	Opportunity for an Account you have full visibility on

How Siebel Remote Processes the Rules in This Example

Siebel Remote evaluates each record in the Dock Object Visibility Rules list in ascending order according to the value in the Sequence property:

- If a dock object visibility rules meets the criteria, then Siebel Remote stops evaluating the rules and uses the value in the Visibility Strength property of the rule that passed as the overall strength for the rule.
- If none of the dock object visibility rules pass, then Siebel Remote uses zero for the overall strength for the rule.

In this example, Siebel Remote does the following:

1. Determines the overall visibility strength of the dock object visibility rules. For example:
   - If the user is on the sales team for the opportunity, then Siebel Remote uses the value in the Visibility Strength property of the first record in the Dock Object Visibility Rules list. This value is 100.
   - If the user is not on the sales team for the opportunity, then Siebel Remote evaluates each visibility rule in sequence until a rule meets the criteria. For example, assume visibility rules with 2, 3, 4, and 5 in the Sequence property all fail. Assume the user does possess full visibility to the account for the opportunity, so rule 6 meets the criteria. In this situation, Siebel Remote uses the value in the Visibility Strength property for the rule that contains 6 in the Sequence property. This value is 50.
2. Compares the visibility strength it gets in the previous step to the Visibility Strength property of the first table that Siebel Tools displays in the Dock Object Tables list, and then does the following:
   - If the visibility strength from the visibility rule is greater than or equal to the visibility strength defined for the table, then Siebel Remote downloads all records from the table to the user.

- If the visibility strength from the visibility rule is less than the visibility strength defined for the table, then Siebel Remote does not download any records from the table to the user.

3. Repeats the previous step for each subsequent record that Siebel Tools displays in the Dock Object Tables list.

For example, assume the overall visibility strength from Step 1 is 50. In this situation, Siebel Remote does the following:

- Does not download any records from the S_NOTE_OPTY table.
- Downloads all records from the S_OPTY table.

In this situation, the user receives all opportunity records but no notes for any opportunity. If the user is on the sales team of the opportunity, then the user receives all notes for the opportunities.

Configuring Dock Objects

This topic describes how to configure dock objects. It includes the following information:

- *Reusing a Predefined Dock Object*
- *Creating a New Dock Object*
- *Adding a Dock Object Table to an Existing Dock Object*
- *Verifying That Siebel Tools Created Dock Objects*
- *Rebuilding the Databases After You Run the Docking Wizard*
- *Cleansing Dock Objects*
- *Creating a Table for a Dock Object*

For more information, see the following topics:

- *Guidelines for Creating a Custom Docking Rule*
- *Determining Technical Fit for Reusing a Predefined Object*
- *Downloading a Data Layer Customization to Remote Users*

Reusing a Predefined Dock Object

Siebel CRM includes dock objects with a predefined Siebel application. Before you create a new dock object, review the predefined dock objects and associated visibility rules thoroughly to determine whether they meet your visibility requirements.

The following table lists some of the business components and their associated dock objects.

Business Component	Dock Object	Primary Table	Visibility Level
Action	Activity	S_EVT_ACT	Limited
Account	Party	S_PARTY	Limited
Asset Mgmt - Asset	Asset	S_ASSET	Limited

Business Component	Dock Object	Primary Table	Visibility Level
Contact	Party	S_PARTY	Limited
Employee	Party	S_PARTY	Limited
Opportunity	Opportunity	S_OPTY	Limited
Position	Party	S_PARTY	Limited
Internal Product	Product	S_PROD_INT	Limited
Service Request	ServiceRequest	S_SRV_REQ	Limited

The Party dock object represents the Employee and Position records. The visibility level for the Party dock object is Limited. The SQL rules in the Party dock object determine visibility for employee and position records as Enterprise.

To reuse a predefined dock object

1. In Siebel Tools, display the object type named Dock Object and all child objects of the Dock Object.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. In the Object Explorer, click Business Component.
3. In the Business Components list, locate the business component you seek.

   For example, query the Name property for Opportunity.
4. Note the value in the Table property.

   For example, Siebel Tools displays the S_OPTY table in the Table property for the Opportunity business component.
5. In the Object Explorer, click the Flat tab, and then click Dock Object.
6. In the Dock Objects list, query the Dock Object Table property for the table you identified earlier in this procedure.

   If a dock object references the table, then Siebel Tools displays the dock object. For example, the Opportunity dock object references the same table that the Opportunity business component references.
7. If your query provides a result, then you might be able to reuse the predefined dock object. Do the following:

   a. Note the value in the Name property.
   b. In the Object Explorer, click the Types tab, and then click Dock Object.
   c. Query the Name property for the value you noted in the first step of this procedure.
   d. Examine the dock object and the child objects of the dock object to determine whether they meet your requirements.

Creating a New Dock Object

The *Docking Wizard* is a tool you can use to do the following:

- Create a new dock object for a custom extension table that is not already in a dock object.
- Create a new dock object table for a custom dock object.
- Create new dock object visibility rules for a custom or predefined dock object.

The Docking Wizard creates or updates the following objects:

- Dock object.
- Dock object table.
- Dock object visibility rule. For more information, see *How the Docking Wizard Creates Visibility Rules*.

You can use the Docking Wizard to create public, private, and limited dock objects.

For more information, see *Guidelines for Using the Docking Wizard* and *How the Docking Wizard Behaves Depending on Where You Start It*.

To create a new dock object

1. Review the predefined dock objects.

   It might not be necessary to create a new dock object. If a dock object already references a table, then Siebel Tools disables the Docking Wizard menu item and you cannot choose it. For more information, see *Reusing a Predefined Dock Object*.
2. Make sure related projects are locked.

   For more information, see *Locking Related Projects*.
3. In the Object Explorer, click Table.
4. In the Tables list, locate the custom extension table that you must associate with a docking object.
5. Optional. Start the Docking Wizard from a table:
   a. In the Tables list, right-click the record, and then click the Docking Wizard menu item.

      For example, right-click the CX_TEST_PRI table. You must choose a custom extension table that includes the CX_ prefix in the name column.
   b. In the Add Table to Dock Object dialog box, enter the name of the dock object into the Dock Object field.

      You must use the DOX prefix. For example, DOX PRI.
   c. Choose a project for the dock object.

      Siebel Tools displays all locked projects in the Project list.
   d. In the Visibility level section, choose Private, Enterprise, or Limited.

      If you chose Limited, then the Docking Wizard creates the visibility rules. For more information, see *How the Docking Wizard Creates Visibility Rules*.
6. Optional. Start the Docking Wizard from a table column:
   a. In the Object Explorer, expand the Table tree, and then click Column.
   b. In the Columns list, locate the column that you must associate with a dock object.

A custom extension column includes an X_ prefix in the Name property.

c. Right-click the record, and then click the Docking Wizard menu item.

The Docking Wizard menu item is active only if one of the following situations is true:

- The column name includes an X_ prefix.
- The table name includes a CX_ prefix and a dock object already references the table.

You can start the Docking Wizard multiple times regardless of how many times you start it for a column.

7. Click Next, review your modifications, and then click Finish.

Siebel Tools creates the dock object.

8. Verify that Siebel Tools created the new objects.

For more information, see *Verifying That Siebel Tools Created Dock Objects*.

9. Rebuild the databases.

For more information, see *Rebuilding the Databases After You Run the Docking Wizard*.

Guidelines for Using the Docking Wizard

If you use the Docking Wizard, then use the following guidelines:

- For a custom extension table, make sure a dock object does not already reference the table. If a dock object does already exist, then do not start the Docking Wizard from the table.
- Do not start the Docking Wizard on a predefined Siebel table.
- You can start the Docking Wizard from a custom extension column that is added to a predefined table.
- You cannot add a custom intersection table to the dock object of a table that Siebel Remote downloads. If you require this functionality, then see *Getting Help From Oracle*.
- You can create a new dock object for a custom table that includes a mandatory foreign key to another custom table that is already part of a custom dock object. You can add it to the predefined custom dock object. This configuration depends on your business requirements.
- The Docking Wizard creates rules with the following visibility strengths:
  - Visibility strength of 50 for a dock object visibility rule
  - Visibility strength of 50 for a custom dock object table
  - Visibility strength of 100 for a check dock object

  You must get assistance from Oracle to modify these strengths. For more information, see *Getting Help From Oracle*.

How the Docking Wizard Behaves Depending on Where You Start It

The behavior of the Docking Wizard differs depending on if you start it from a table or a table column.

If you start the Docking Wizard from a table, then the following applies:

- If the custom table is a stand-alone table, then you must create a new dock object for the table, and then create the dock object visibility rules.
- If the custom table includes foreign keys to other custom tables that already exist in some dock objects, then you can do one of the following:

    - Create a new dock object.
    - Add the table to a predefined custom dock object.

- If you start the Docking Wizard from a stand-alone custom table, then only the Create a New Dock Object option is active in the Add Table to Object dialog box. The Add the Table to an Existing Dock Object option is not active.

If you start the Docking Wizard from a column, then you do not need to make any choices. The Docking Wizard adds the following dock object visibility rules:

- For a regular foreign key, the Docking Wizard adds the following dock object visibility rules:

    - One rule from the dock object of the table to the dock object of the foreign key table
    - One rule from the dock object of the foreign key table to the dock object of the table

  These rules are for a Check Dock Object visibility type.

- For a foreign key to the S_POSTN table, the Docking Wizard only adds a position dock object visibility rule.

How the Docking Wizard Creates Visibility Rules

You do not manually create new dock object visibility rules. Siebel Tools adds visibility rules to the dock object depending on the visibility type of the dock object and the structure of the tables involved. Siebel Tools does this in the following situations:

- You use the Docking Wizard to add a dock object table to a custom dock object.
- You start the Docking Wizard from a custom extension column that is a foreign key to another table.

You can use the Docking Wizard to create the following types of limited dock object visibility rules:

- Employee
- Employee Manager
- Position
- Position Manager
- Check Dock Object

For more information, see *Dock Object Visibility Rule*.

Locking Related Projects

The Docking Wizard creates visibility rules on associated dock objects. If another project requires a new visibility rule, and if that project is not locked, then Siebel Tools displays a dialog box that prompts you to lock the project. Note the following requirements:

- If you create a new dock object for a stand-alone custom table, then you must lock the project that the new dock object references before you create the new dock object.
- If you create a new dock object for a custom table that is not a stand-alone table, then you must do the following before you create the new dock object:

    - Lock the project that the new dock object references.
    - Lock all projects for the dock objects where the parent table of the custom table resides.

Adding a Dock Object Table to an Existing Dock Object

You can add a new dock object table to an existing dock object. If the user possess access to the parent record in the existing table, then Siebel Remote downloads records from the new dock object table.

The Docking Wizard adds new dock object visibility rules for a predefined dock object. Siebel Remote uses the new dock object visibility rules to determine to download or not download records from an existing table to the Remote user. This configuration is appropriate in the following situations:

- If the new table acts as a parent to the primary table of another, limited visibility dock object.
- If the new table includes a foreign key to the primary table of another limited visibility dock object.

To add a dock object table to an existing dock object

1. Complete Step 1 through Step 4 in *Creating a New Dock Object*.
2. In the Tables list, right-click the record, and then click the Docking Wizard menu item.

   For example, right-click the CX_TEST_PRI table. You must choose an existing extension table that includes the CX_ prefix in the name column.
3. In the Add Table to Dock Object dialog box, choose the Add the Table to an Existing Dock Object option.
4. Choose an entry from the Dock Object list.

   Siebel Tools displays a list of all Dock Objects that contain the tables that the new table references as a foreign key.
5. Choose an entry from the Source Column list.

   This list allows you to choose a column from the new table that is a foreign key to the parent table that is contained in the chosen Dock Object Table. Typically, Siebel Tools only displays one column, but there might be more in some situations.

   If you choose the Source Column, then Siebel Tools displays a value in the Target Table field.
6. Click Next, review your modifications, and then click Finish.

   Siebel Tools creates a dock object table object, and then displays it in the Dock Object Tables list.
7. Verify that Siebel Tools created the new objects.

   For more information, see *Verifying That Siebel Tools Created Dock Objects*.
8. Rebuild the databases.

   For more information, see *Rebuilding the Databases After You Run the Docking Wizard*.

Verifying That Siebel Tools Created Dock Objects

After you create a new dock object, dock object table, or dock object visibility rule, you can verify that Siebel Tools created the new objects.

To verify that Siebel Tools created dock objects

1. Display the object type named Dock Object and all child objects of the Dock Object.

For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.

2. In the Object Explorer, click Dock Object.
3. In the Dock Objects list, locate the new dock object.
4. In the Object Explorer, expand the Dock Object tree, and then click Dock Object Table.
5. In the Dock Object Tables list, locate the new table.
6. In the Object Explorer, click Dock Object Visibility Rule.
7. In the Object Visibility Rules list, locate the new visibility rules.

For more information, see *Dock Object Visibility Rule* and *Siebel Remote and Replication Manager Administration Guide* .

Rebuilding the Databases After You Run the Docking Wizard

After you run the Docking Wizard you must rebuild the following databases:

- Visibility database, which uses the dobjinst.dbf file
- Visibility ID database, which uses the visdata.dbf file

This rebuild operation allows database extract and download to work properly. For more information, see *Siebel Remote and Replication Manager Administration Guide* .

To rebuild the databases after you run the Docking Wizard

1. Stop the Siebel Server services.
2. Delete or rename the following files, if they exist:

```
ORACLE_HOME\bin\diccache.dat
ORACLE_HOME\bin\dicdata.dat
```

3. Start the Siebel Server services.
4. Stop the Transaction Processor component.
5. Stop the Transaction Router component.
6. To rebuild the dobjinst.dbf file, start the Transaction Processor.
7. To rebuild the visdata.dbf file, start the Transaction Router with the ldDbRecreate parameter set to TRUE.
8. Reextract the Remote clients.
9. Make sure the tables now include data that references extraction and download information.

Cleansing Dock Objects

Dock object integrity might be compromised in the following situations:

- If you delete a custom table, column, or dock object.
- If you redefine a foreign key column to reference a different table.

In these situations, you must cleanse the dock objects before you can use the Docking Wizard again or before you can use Siebel Remote.

To cleanse dock objects

1. In Siebel Tools, in the Object Explorer, click Dock Object.

2. In the Dock Objects list, click Cleanse.

   Siebel Tools does the following:

   a. Examines all dock objects in the Dock Objects list.
   b. Prompts you to make sure all the dock objects are clean. If they are not, then Siebel Tools deletes some objects.
   c. If the projects are not locked, then Siebel Tools prompts you to lock them. After Siebel Tools finishes, it repeats the previous step.

Creating a Table for a Dock Object

You can create a table for a dock object.

To create a table for a dock object

1. In Siebel Tools, make sure related projects are locked.

   For example, lock the project that the new dock object will reference, such as Dock Opportunity. For more information, see *Locking Related Projects*.
2. Click the File menu, and then click New Object.
3. In the New Object Wizards dialog box, choose the Table icon in the General tab, and then click OK.
4. In the first General dialog box, do the following:

   a. Enter the name of your custom extension table with a CX_ prefix.

      You must include a CX_ prefix. For example, CX_TEST_PRI.
   b. Choose the project.
   c. Choose the type of table.
5. Click Next, and then click Finish. Siebel Tools creates the table, and then displays the Tables list.

25 Localizing Siebel CRM

Localizing Siebel CRM

This chapter describes how to configure your Siebel CRM application so that you can deploy it in a localized environment. It includes the following topics:

- *Overview of Localizing a Siebel Application*
- *Localizing a Multilingual List of Values*
- *Converting Your Current Data for an MLOV*
- *Configuring Certain Siebel Modules to Use MLOV Fields*

For more information about localizing Siebel CRM, see also *Siebel Global Deployment Guide* .

Overview of Localizing a Siebel Application

This topic describes an overview of localizing Siebel CRM. It includes the following information:

- *About Localization in the Development Environment*
- *Deleting a Control or List Column While in Language Override Mode*
- *Localizing an Application Menu*
- *Localizing Help*

For more information, see the following topics:

- *Creating Browser Scripts*
- *Troubleshooting a View That Siebel CRM Does Not Display in the Siebel Client*
- *Setting the Language Mode of the Applet Web Template Editor*
- *Setting the Input Method Editor Mode on a Control or List Column*
- *Using the Conversion Wizard to Convert a Form Applet to Grid Layout*
- *Guidelines for Modifying a Predefined Query*
- *Properties of a Command*

About Localization in the Development Environment

Localization is the process of configuring Siebel CRM so that you can deploy it into an environment that requires information be displayed in a format that is specific to the local environment, such as the natural language that a set of users use to communicate. Siebel CRM maintains, in the same repository, a translatable text string and data that is specific to a language for a Siebel object. Siebel Tools allows you to edit a property that is specific to a locale for an object, such as an applet, view, or control. For more information, see *Using Siebel Tools* and *Siebel Global Deployment Guide* .

Locale Object Types

A *locale* is an object that you use to define locale data for the parent object of an object type that contains localizable data, such as a symbolic string. Siebel CRM stores data for a locale object in a set of repository tables that it designates specifically for storing locale data. These tables use a naming format that includes the name of the base table followed by the suffix _INTL. For more information, see *Using Siebel Tools* .

Siebel Tools Language Mode

To determine the localizable data to use with translatable data, Siebel Tools runs in a language mode. Siebel Tools runs in an English-American user interface, but you can edit localizable data in the language of your choice. The default edit language is English-American. Siebel Tools includes a language mode that you can choose from the Development Tools Options dialog box. If you add more languages, then you must enter the language code in all capital letters. For more information, see *Using Siebel Tools* .

Checking Out and Checking in Locale Data

The Siebel Server tracks the language that you use when you check out the project. Siebel Tools displays this information in the Server Language column of the Check Out dialog box. This feature allows your team to work with data in a language other than the language that you use when you check out the project.

You can get locale data for a project. You can do this if you modify your current working language. For example, assume you only use language data for English-American as your current working language in Siebel Tools, and you must switch to French. To view any localizable data in Siebel Tools, you must get the locale data for French. For more information, see *Using Siebel Tools* .

Locale Management Utility

The Locale Management Utility is a tool you can use in Siebel Tools. It allows you to export and import text strings and locale information to an external file. You typically use this utility to export strings to send out for translation, and then to import the translated strings back into the Siebel repository. It facilitates a concurrent application configuration and localization process. You use this option if you deploy in multiple languages. To start this utility, you choose the Tools menu, Utilities, and then the Locale Management menu item. For more information, see *Using Siebel Tools* . See also *Siebel Global Deployment Guide* .

Compiling and Deploying

Every time you configure a column to be multilingual, you must deploy your changes to the Siebel Runtime Repository. You only need to migrate the Table ListOfValues project. You must deploy modifications to users so that they can view the configured lists in the required language.

The Replication Level field of a multilingual list of values (MLOV) determines the replication level of the list of values record. Setting this field to All routes the record to the regional databases and mobile clients. If you run the MLOV Converter Utility in translation mode to update the target columns and S_LST_OF_VAL table, then the utility does not log modifications in the transaction log table. You must reextract the regional databases and remote clients. For more information, see *Converting Your Current Data for an MLOV*.

Deleting a Control or List Column While in Language Override Mode

If you work in language override mode, then do not delete a control or list column from an applet web layout.

CAUTION: If you work in language override mode, then do not delete a control or list column from an applet web layout. Instead, make sure the Visible property for the control does not contain a check mark. If you delete a control or list column while working in language override mode, then Web Tools deletes the corresponding object for all languages, not just for the language that you use. If you undo after you cut items from the applet layout, then close the Applet Web Template Editor without saving your modifications. For more information, see *Deleting a Control or List Column*.

Localizing an Application Menu

When Siebel CRM translates an application menu to a language other than English, more space might be required to fit all the characters that the other language uses in the menu. To allow for this requirement, you can increase the width parameter in a od tag.

Localizing an application menu

- Increase the width parameter in the tag in the CCFramebanner web template.

  For example, to localize an application menu for Japanese, increase the width parameter from 275 to 405. For more information on the CCFrameBanner web template, see *Configuring Web Templates to Display Menus, Toolbars, and Thread Bars*.

Localizing Help

This topic describes how to deploy help in different languages. If you must use Siebel CRM in a language that is not available from Oracle, and you must deploy help in that language, then you must localize the help. For more information, see *Siebel Global Deployment Guide* .

To localize help

1. If the predefined localized help meets your requirements, then use that predefined help and exit this task.

   Siebel CRM comes with predefined localized help. For more information, see *Predefined Localized Help*.
2. If the predefined localized help does not meet your requirements, then configure the ENU (American English) help to meet your requirements.
3. To translate the HTML source files, modify the flat files.

   These HTML files constitute the help.
4. Test you modifications and correct any errors.
5. Distribute the localized help to the Siebel Servers and Siebel clients.

Predefined Localized Help

Siebel CRM comes with predefined localized help. Localized help files are located in the language folders on the Siebel Server or the Siebel client. Help files are installed in the following location on the Siebel Application Interface:

```
SIEBEL_AI_ROOT\applicationcontainer_external\install_language\help
```

where:

- SIEBEL_AI_ROOT is the directory where you installed Siebel Application Interface.
- install_language is the language you chose during installation.

Localizing a Multilingual List of Values

This topic describes how to localize a multilingual list of values. It includes the following information:

- *Overview of Language-Independent Code*
- *Configuring a Multilingual List of Values*
- *Defining Properties of an MLOV*
- *Adding Records for All Supported Languages*
- *Searching a Multilingual List of Values*
- *Searching Fields That an MLOV Controls*
- *Deactivating an MLOV Record Instead of Deleting It*
- *Guidelines for Localizing a Multilingual List of Values*

A *multilingual list of values* (MLOV) is a type of list of values that allows you to display values in the natural language that the user uses to communicate. It allows a user who works in a particular language to get values for another language.

Siebel CRM displays an MLOV in a static list. To configure an MLOV for a predefined static list, the list must meet the following requirements:

- It must be bounded.
- It must not be hierarchical.

For more information about the active language, see *Siebel Global Deployment Guide* . For more information about list of value fields, see *Siebel Applications Administration Guide* .

For more information, see the following topics:

- *About Static Lists*
- *Creating a List of Values*
- *Guidelines for Modifying a Predefined Query*

Overview of Language-Independent Code

The list of values table contains the following columns:

- Display Value
- Language Independent Code

Monolingual and multilingual lists of values display values from the Display Value column. If the user chooses a value in a list, then the actual value that Siebel CRM stores in the Siebel database is different for monolingual and multilingual lists of values:

- A monolingual list stores the display value.
- A multilingual list stores the language-independent code.

Language-independent code (LIC) is a mechanism that allows Siebel CRM to do the following:

- Store data in a form that a user working in another language can get
- Roll up of data for management reports regardless of the language of the user who enters the data

The following table describes an example of how language-independent code works. In this example, a multilingual list displays the Display Value of Mr., Señor, or Herr, depending on the active language of the user. The list stores the value Mr. in the Siebel database because Mr. is the value that is defined in the Language Independent Code column.

Display Value	Language-Independent Code
Mr.	Mr.
Señor	Mr.
Herr	Mr.

The language-independent code value for predefined list of values data is typically the same as the American-English version.

You define an MLOV on a column basis. The columns that are not configured for multilingual continue to store display values instead of language-independent codes.

Configuring a Multilingual List of Values

To configure an MLOV, you modify objects in Siebel Tools, and then perform administration tasks in the Siebel client. If your implementation uses certain Siebel modules, such as Siebel Workflow, then you must perform more configuration. For more information, see *Configuring Certain Siebel Modules to Use MLOV Fields*.

To configure a multilingual list of values

1. Consider potential performance issues.
   For more information, see *Considering Factors That Affect MLOV Performance*.
2. In Siebel Tools, display the following object types:
   - Dock Object
   - Dock Object Visibility Rule

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
3. Make sure the list of values is translatable:
   a. In Siebel Tools, in the List of Values list, locate the list of values you must modify.
      For more information, see *Creating a New List of Values*.
   b. Make sure the Translate property contains a check mark.
      For more information, see *Modifying the Translate Property*.
   c. Make sure the Multilingual property contains a check mark.

4. Make sure the list is bounded:
    a. In the Object Explorer, click the Flat tab, and then click Pick List.
    b. In the Pick Lists list, query the Type Value property for the type of list of values you must modify.
    c. Make sure the Bounded property contains a check mark.
5. Make sure the columns that the list references are bounded and consistent:
    a. In the Object Explorer, click Column.
    b. In the Columns list, query the LOV Type property for the list of values type you must modify.
    c. Make sure the LOV Bounded property for each column contains a check mark.

       For more information, see *Example of Determining Whether the List Is Bounded*.
    d. Make sure the Translation Table Name property is set to S_LST_OF_VAL for all columns.
    e. Make sure the LOV Type for the list matches the LOV Type of the column that the field for the list references.
6. Make sure you can use the column with an MLOV.

   For more information, see *Columns That You Cannot Use with an MLOV*.
7. Make sure the column that is referenced by the field that uses the list contains the following property.

Property	Value
Translation Table Name	S_LST_OF_VAL

8. Check the visibility rules for references to the list of values that is a part of your MLOV configuration:
    a. In the Object Explorer, click the Flat tab, and then click Dock Object Visibility Rule.
    b. In the Dock Object Visibility Rules list, query the SQL Statement field for literals across all rows that are not empty.
    c. Identify the values that Siebel CRM must translate.
    d. If necessary, modify the Display Value to the language-independent code.

   You must modify the display value for any reference in a visibility rule that references a list of values entry for a type that you configure for multilingual support. Note that you cannot modify visibility rules.
9. Test and then deliver your Workspace.
10. Configure display values for each language you must support:
    a. Open the Siebel client, navigate to the Administration - Data screen, and then click the List of Values link.
    b. Create a new record for each display value for the type of list of values that you use for a language.
    c. Locate the list of values you must modify.

       For more information, see *Adding Records for All Supported Languages*.
    d. Repeat the previous step for each language you must support.
11. Use the MLOV Converter Utility to convert data for the current lists of values.

    For more information, see *Converting Your Current Data for an MLOV*.
12. Test your modifications.

Example of Determining Whether the List Is Bounded

The following information lists columns for the AVAILABILITY_STATUS list of values type. Three of the columns are bounded, but you cannot configure these columns as multilingual because the NEXT_AVAIL_CD column is not bounded. If you run the MLOV Converter Utility on this configuration, then the utility displays an error message similar to `columns are inconsistently bounded`. For more information, see *Fixing an Inconsistently Bounded List of Values or an Improperly Set Translation Table Property*.

Name	LOV Type	LOV Bounded
CURR_AVAIL_CD	AVAILABILITY_STATUS	Y
NEXT_AVAIL_CD	AVAILABILITY_STATUS	Y
CURR_AVAIL_CD	AVAILABILITY_STATUS	Y
NEXT_AVAIL_CD	AVAILABILITY_STATUS	N

You can modify the LOV Bounded and LOV Type properties of the column in the following situations:

- For a predefined column that is not already assigned to a predefined list of values type.
- For a predefined column that is already assigned to a predefined list of values type and that has the LOV Bounded property set to FALSE, you can modify the LOV Bounded property to TRUE. Siebel CRM supports this configuration only in the context of enabling an MLOV.

You can configure a custom extension column for use with an MLOV. Do not configure a column for an MLOV unless you are sure that you intend to use that column in your implementation.

Example of Translating Names That Siebel CRM Displays in a List of Values

The Tactics GanttChart Ax Applet - Home Page applet is a standard Gantt chart applet that is part of the Home Page View (DBM) view. This applet is similar to the FS DB Planned GanttChart AX Applet in the predefined FS AxGanttChart View. The FS Dispatch Board Screen includes the FS AxGanttChart View.

The following LOV types control how Siebel CRM displays information in the Tactics GanttChart Ax Applet - Home Page applet:

- The MONTH_NAME LOV type controls the month names.
- The DAY_NAME LOV type controls the day names.

Siebel CRM displays the month and day names in the second frame of the Gantt chart applet. This frame includes scheduled time periods in a calendar. You can translate the month and day names.

To translate names that Siebel CRM displays in a list of values

1. In Siebel Tools, in the List of Values list, locate the list of values you must modify.

   For more information, see *Creating a New List of Values*.
2. Add translated display values for the languages that Siebel CRM must display.

   For more information, see *Adding Records for All Supported Languages*.

3. Make sure the Multilingual property contains a check mark for the LOV type row and the display value rows.

Columns That You Cannot Use with an MLOV

The following information lists columns that you cannot use with an MLOV.

Table	Column	LOV Type	Bounded?
S_AGREE_POSTN	APPR_ROLE_CD	AGREEMENT_APPR_ROLE	Yes
S_AUDIT_ITEM	You cannot use any columns in the S_AUDIT_ITEM table.	Not applicable	Not applicable
S_CONTACT	PREF_LANG_ID	No LOV type	No
S_CONTACT_X	ATTRIB_48	No LOV type	No
S_CS_RUN	STATUS_CD	CALL_SCRIPT_SAVE_STATUS	Yes
S_DOC_ORDER	TAX_EXEMPT_REASON	GLOBAL_TAX_EXEMPTION	Yes
S_ONL_LAYOUT	CONTROL_TYPE_CD	No LOV type	No
S_ORG_EXT	DIVN_CD	SAP_DIVN_CD	Yes
S_ORG_EXT	DIVN_TYPE_CD	DIVISION_TYPE	Yes
S_ORG_EXT_XM	NAME	No LOV type	No
S_PRI_LST_ITEM	PRI_METH_CD	SRVC_PRICING_METHOD	Yes
S_PROD_INT_CRSE	CRSE_TYPE_CD	SOURCE TYPE (Internal)	Yes
S_PROD_INT_X	ATTRIB_50	No LOV type	No
S_PROD_INT_X	ATTRIB_51	No LOV type	No
S_PROD_INT_X	ATTRIB_53	No LOV type	No
S_PROJ_ORG	PROJ_ROLE_CD	PS_SUBCONTRACTOR_ROLE	No
S_PROJITEM	PROD_AREA_CD	PROD_DEFECT_SUB_AREA	Yes
S_PROJITEM	STATUS_CD	No LOV type	No

Table	Column	LOV Type	Bounded?
S_SRC	SRC_CD	SOURCE_TYPE	Yes
S_SRC	STATUS_CD	CAMPAIGN_STATE	No
S_SRC_EVT	FORMAT_CD	EVENT_FORMAT	Yes
S_SRCH_PROP	NAME	No LOV type	No

Note: LOVs that will be stored in columns defined as **Number** cannot be MLOV-enabled.

Defining Properties of an MLOV

You can define properties of an MLOV in Siebel Tools.

To define properties of an MLOV

1. In Siebel Tools, locate the list of values you must modify in the List of Values list.

   For more information, see *Creating a New List of Values*.
2. Define properties for the MLOV using values from the following table.

Property	Description
Multilingual	Indicates the list of values is multilingual. The MLOV Converter Utility sets this flag for the values in the list of values. For more information, see *Converting Your Current Data for an MLOV*. If you add a new MLOV record after the MLOV Converter Utility runs, then you must manually add a check mark to the Multilingual property to make sure it is consistent with the previously created records.
Language Name	Indicates the natural language. In the Siebel client, Siebel CRM gets the values for this list from the Languages view in the Administration - Data screen.
Translate	If you add a list of values type that must work as an MLOV, then make sure the Translate property contains a check mark. Do not modify the Translate field for a predefined list of values. For more information, see *Modifying the Translate Property*.
Language-Independent Code	The internal language-independent code for a list of values. Siebel CRM stores it in the Siebel database for an MLOV that a Siebel application enables and references. The language-

Property	Description
	independent code must be 30 characters or less. It is typically the English-American version. You cannot modify the language-independent code. If you click the List of Values Explorer link in the Siebel client, then the Code field displays the language-independent code.
Display Value	Contains the text that Siebel CRM displays in a list. It stores it in the Siebel database for an MLOV that is not enabled. To determine the display value, if display values exist for more than one language for a list of values, then Siebel CRM uses the current active language.

Modifying the Translate Property

A *translatable list of values* a list of values that Siebel CRM can translate into another language without affecting the functionality of Siebel CRM. The Translate property indicates if Siebel CRM is allowed to translate the display value to another language. If the Translate property contains a check mark, then Siebel CRM can translate. You must update this information manually to reflect your configuration for any MLOV you add.

Modifying the Translate Property for a Predefined List of Values

The following situations apply for a predefined list of values:

- Do not modify the Translate property for a predefined list of values.
- Siebel CRM does not translate an MLOV whose Translate property does not contain a check mark.
- The Translate property is an information-only property that Siebel Engineering uses.
- No client or server functionality is associated with the Translate property.
- To translate the chosen text to the language-independent code, Siebel CRM hard codes translate functionality to use the Display Value property. You cannot use a different value for translation.

Adding Records for All Supported Languages

If you add a new list of values record for a multilingual list of values type, and if you do not add records for each supported language, and if a user who uses one of these languages attempts to view the information, then Siebel CRM displays the language-independent code instead of the display value.

If you use Assignment Manager, then you must add records for all the languages you support. For more information, see *Configuring Siebel Assignment Manager to Use MLOV Fields*.

For more information about adding records to the list of values table, see *Siebel Applications Administration Guide* .

To add records for all supported languages

- If you add a new list of values record for a multilingual list of values type, then you must add records for all supported languages.

For example, assume you must support German, French, and English. In this situation, you create two new records for each display value: one record for German and one record for French.

Make sure the language-independent code for each new record is the same as the original record. Make sure the Language and Display Value fields are set differently to reflect the language.

For more information, see *Defining Properties of an MLOV*.

Searching a Multilingual List of Values

Searching a multilingual list of values (MLOV) can consume significant resources because this search requires a join to the S_LST_OF_VAL list of values in the SQL code. To improve performance if many records and multiple languages exist, you can use the Fieldname.TransCode function in the Search Specification property of the business component. This function gets the untranslated language-independent code (LIC) from a column in the base table instead of getting the display value in the current language. It uses the following format:

```
[Fieldname.TransCode] = 'lang_ind_code'
```

where:

- TransCode is case-sensitive. You cannot use the following formats:

```
Fieldname.Transcode
```

```
Fieldname.transcode
```

Example of Searching a Multilingual List of Values

For example, the Status field in the Service Request business component references the SR_STAT_ID column. This column stores the language-independent code for the Open value. A query on the Status field with a multilingual list of value always returns the display value for Open in the current language. For example, if the language is DEU, then it returns Offen. The Language Parameter sets this language.

Assume you must set a language-independent search specification and you use the following code:

```
[Status] = LookupValue('Open')
```

In this situation, to find the DEU display value for the Open value for the language-independent code, and to compare this value to the language-independent code in the SR_STAT_ID column, the DEU object manager adds a join to the S_LST_OF_VAL list of values in the SQL. This configuration makes the query more complex and difficult to support with an index.

To simplify the SQL command, you can use the following search specification:

```
[Status.TransCode] = 'Open'
```

The Status.TransCode function in the SQL query can get the value for the value directly from the database column. Siebel CRM does not translate the value for the language-independent code into the display value.

The Fieldname.TransCode function improves the performance of a query on an MLOV column, particularly if the query or search specification includes other columns from the same base table. This improvement occurs because Siebel CRM can use a combined index that includes the MLOV column. It cannot do this if it uses the LookupValue function or if it queries directly on the MLOV.

Using a Calculated Field or Script to Search a Multilingual List of Values

You can use the Fieldname.TransCode function with a calculated field or the GetFieldValue business component method. For example, you can:

- Create a calculated field that uses the Priority.TransCode calculated value in the Service Request business component.
- Use the this.GetFieldValue("Priority.TransCode") statement in a script.

This configuration only works if you use an MLOV field, such as the Translation Table Name property of the column that this field references.

Searching Fields That an MLOV Controls

To perform a search, Siebel CRM applies the following function to the language-independent code:

```
LookupValue (LOV Type, Language-Independent Code)
```

You can use this function to configure search with a predefined query and search expression.

For more information about the LookupValue function, see *Guidelines for Using Code in an MLOV Configuration*. For more information, see *Guidelines for Modifying a Predefined Query*.

For more information about query operators and expressions, see *Siebel Developer's Reference* and *Siebel Fundamentals* .

To search fields that an MLOV controls

1. Use the display value for the search specification.

   Do not use the language-independent code to query. A query translates the search specification to the appropriate language-independent code. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.
2. Make sure the Display Value you use as the search specification corresponds to the language that Siebel CRM uses to perform the query.

   If the query runs through a Siebel interface, such as COM, then the language that Siebel CRM uses for this translation is configured in the configuration file that it uses with the interface.

Deactivating an MLOV Record Instead of Deleting It

When you administer an MLOV during the lifetime of a Siebel application, it might be necessary to deactivate an MLOV record that Siebel CRM no longer requires. It is recommended that you deactivate the record instead of deleting it. Siebel CRM can correctly display an inactive record in other tables that use the record. It does not include inactive records in any lists. Enterprise Integration Manager ignores inactive records when it validates a list of values.

If you delete an MLOV record, then Siebel CRM does the following:

- Uses the display value in the list of values entries to display the language text. Note that it cannot correctly display records in other tables that use the MLOV record.

- To display the language-independent code, it uses the language-independent codes in the target columns that refer to the deleted record.

To deactivate an MLOV record instead of deleting it

1. In the Siebel client, navigate to the Administration - Data screen, and then choose the List of Values link.
2. Make sure the Active field does not contain a check mark.

   Siebel CRM includes a check mark for the Active field, by default.

Guidelines for Localizing a Multilingual List of Values

This topic describes guidelines for localizing an MLOV.

Guidelines for Configuring a Multilingual List of Values

If you configure an MLOV, then use the following guidelines:

- Make sure language-independent code is unique. For more information, see *Language-Independent Code Must Be Unique*.
- If the header row entry is inactive, then make sure the Display Value rows are not active.
- You cannot configure a hierarchical MLOV. Siebel CRM does not support this configuration. If you require a hierarchical MLOV, then see Article ID 473813.1 on My Oracle Support, which was previously published as Siebel Technical Note 632. For help with a hierarchical MLOV, see *Getting Help From Oracle*.
- Do not create more than one header row for an MLOV type. For example, assume an MLOV named ACCOUNT_STATUS includes nine Display Value rows, three rows each for English, Spanish, and German. Siebel CRM requires only one header row for these nine values. If you create three header rows for the ACCOUNT_STATUS MLOV, then the MLOV will fail.
- Make sure the length of the table column that stores the language-independent code is equal to the longest display value for the MLOV.

  > **CAUTION:** The length of the table column that stores the language-independent code must equal the longest display value for the MLOV. This length is 30 characters. If it does not, then Siebel CRM truncates the display value. If the predefined column does not meet your requirements, and if you use a custom extension column, then you must make sure the column is a VARCHAR column and has a maximum length of 30.

- Make sure any customization you perform that directly involves the list of values table is compatible with other MLOV functionality in your Siebel application. For display, Siebel CRM uses a lookup to convert the underlying language-independent code to the corresponding display value. For search and sort, it performs a database join to the list of values table.
- Associate an MLOV with only one business component field. Siebel CRM uses only one multilingual list type for each column. Multiple business components can reference a table, and multiple business component fields can reference the same column in a table. When run in validation mode, the MLOV Converter utility makes sure an MLOV is associated with only one field. For more information, see *Converting Your Current Data for an MLOV*.

Guidelines for Using Code in an MLOV Configuration

If you use code in an MLOV configuration, then use the following guidelines:

- Do not hard-code the conditions for a dynamic drilldown or toggle applet. Instead, use the LookupValue function. A drilldown or toggle applet references a business component field that includes a value from a list

of values. These values are dynamic. You must not hard-code them. For example, a dynamic drilldown might navigate the user in the following ways:

- To a Credit Card screen if the account type is *Credit Card*
- To a Savings screen if the account type is *Savings*

- Never use Siebel Visual Basic to hard-code the Display Value. Instead, use the language-independent code. Siebel Visual Basic does not include a function that gets a Display Value that is specific to a particular language. To write Siebel Visual Basic code using only language-independent code, you must create a calculated business component field that contains the language translation for a language-independent code.
- Use the LookupName function only in a calculated field or in a search specification. You cannot use it with Siebel scripting. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.
- For the Pre Default Value and Post Default Value properties of a business component field that uses a list of values list, always prepend the LookupValue function with *Expr:*. The first argument is the LOV Type. The second argument is the language-independent code. The function returns the Display Value for the language. For example:

  ```
  Expr: "LookupValue (""FS_PROD_ALLOC_RULES"", ""Default"")"
  ```

- If you define a search specification for a business component, link, applet, or list, then use the LookupValue function. For example:

  ```
  [Invoice Code] = LookupValue('FS_INVOICE_CODE', 'Auction')
  ```

  For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.

Guidelines for Using Enterprise Integration Manager with an MLOV

Enterprise Integration Manager (EIM) can import and export data. You can import data into the list of values table and other tables in Siebel CRM. If you use Enterprise Integration Manager with an MLOV, then use the following guidelines:

- If you import data into the list of values table, then you must make sure the source table includes a language code and a name-value pair. This pair includes the display value and the language-independent code.
- If you import data into any other table, then you must provide a language code for the LANGUAGE command-line parameter for EIM. The source table must include the display value for multilingual columns in the language defined in the parameter. EIM validates imported data against list of values entries. It converts incoming data to the related language-independent code during the import.
- When EIM validates MLOV values during an import, it ignores list of values entries that are marked inactive.
- During an export, you must define a language code for the LANGUAGE parameter so that EIM can correctly translate the language-independent code in the table to the display value.

For more information, see *Siebel Enterprise Integration Manager Administration Guide* .

Language-Independent Code Must Be Unique

If you create a multilingual list of values or a hierarchical list of values, then use a unique language-independent code. For a monolingual list where no language-independent code exists, you can use unique display values. For example, assume a customer requires the following country and state hierarchical list of values:

- Parent list of values is COUNTRY.

  Required values include Australia and USA.
- Child list of values is STATE.

Required values include Western Australia and Washington State.

In this example, WA is not assigned to the language-independent code for both child list of value entries. The meaning is different for each location, so Siebel CRM assigns a unique language-independent code. For example, WESTERN_AU and WASH_STATE_USA.

Situations exist where it is appropriate to use duplicate values in the child list of values, but you must assign these the same display values in the same language. For example:

- Parent list of values is DEFECT_TYPE

  Required values include Product Defect and Documentation Defect
- Child list of values is STATUS

  Required values include Open and Closed

The status can be Open for each type of defect, so the list of values table contains multiple entries with the display value Open, one for each time that Siebel CRM can use it with each parent list of values entry. The LookupValue function returns the first value in the list of values table that matches the supplied LOV_TYPE and language-independent code values, so it is essential that Siebel CRM assigns the same display value.

Converting Your Current Data for an MLOV

This topic describes how to convert your current data for an MLOV. includes the following information:

- *Guidelines for Converting Your Current Data for an MLOV*
- *Parameters You Use to Run the MLOV Converter Utility*
- *Resuming the MLOV Converter Utility If an Error Occurs*
- *Using the MLOV Converter Utility to Convert Multiple Languages*
- *Troubleshooting Problems with an MLOV Conversion*

After you configure Siebel CRM to use an MLOV, you use the MLOV Converter Utility to convert data for the current lists of values. The MLOV Converter Utility does the following:

- For each column configured for an MLOV, locates values in lists of values in user data that the S_LST_OF_VAL table does not contain.
- Inserts these values into the S_LST_OF_VAL table as inactive.
- Modifies the display value of bounded columns to the language-independent code and sets the value for the Multilingual property to true.

To convert your current data for an MLOV

1. In Siebel Tools, delete all indexes that reference the columns you must convert.

   You will recreate these indexes after you finish the MLOV conversion.
2. Start the Siebel Database Configuration Wizard.

   For information, see *Siebel Installation Guide* .
3. Define the required parameters.

For more information, see *Parameters You Use to Run the MLOV Converter Utility*.

4. Choose Run Database Utilities.
5. Choose Multi-Lingual List of Value Conversion.
6. Choose Validate Mode.
7. Run the MLOV Converter Utility.
8. Review the log file and resolve errors, as necessary.

   The MLOV Converter Utility checks for errors and writes them to a log file. The default name of the log file is `mlovupgd_verify.log`. The default location of the file is the `siebsrvr\LOG` directory.
9. If the utility reports an error, then resume the utility in validation mode.

   For more information, see *Resuming the MLOV Converter Utility If an Error Occurs*.
10. Repeat Step 2 through Step 9 until the utility does not report any errors.
11. Repeat Step 2 through Step 5 to restart the Siebel Database Configuration Wizard.
12. Specify Translation Mode, and then run the MLOV Converter Utility.
13. Recreate the indexes you deleted in Step 1.
14. Test and then deliver your Workspace.

    For more information, see *Using Siebel Tools* .

Guidelines for Converting Your Current Data for an MLOV

If you convert your current data for an MLOV, then use the following guidelines:

- It is recommended that you backup the Siebel database before you run the utility. You cannot reverse or undo a conversion.
- You must perform this conversion even if you recently completed a new installation of your Siebel application.
- You can run the utility as often as necessary. The utility only processes data that is not already converted.

You run the MLOV Converter Utility in one of the following modes:

- **Validation.** Validates the current repository for data inconsistencies. If the utility finds inconsistencies, then the utility writes errors to a log file, and then stops.
- **Translation.** Does the following:
  - If a column is configured for an MLOV, then the utility modifies the display value for the column to the language-independent code.
  - If you set the target column for a LOV Type to multilingual, then to make sure the multilingual state of the target column and the corresponding list of values in the S_LST_OF_VAL table are consistent with each other, the utility sets the MULTILINGUAL flag to TRUE in the S_LST_OF_VAL table.

    The MLOV Converter Utility sets the multilingual flag to TRUE for the header row and the Display Value rows for the MLOV.
  - Verifies that target columns that use the MLOV type are configured. A *target column* is a column that stores the display value or the language-independent code as part of user data.

Parameters You Use to Run the MLOV Converter Utility

The following table describes the parameters you use to run the MLOV Converter Utility.

Name of Dialog Box	Values You Must Choose
Siebel Enterprise Parameters: Gateway Name Server Address	Gateway Address Enterprise Server Address
Installation and Configuration Parameters: Siebel Server Directory	Siebel Server Directory
Installation and Configuration Parameters: Siebel Database Server Directory	Database Server Directory
Database Server Options: Siebel Database Operation	Run Database Utilities
Database Utilities: Database Utility Selection	Multilingual List of Values Conversion
MLOV Parameters: MLOV Operation	Validate or Translate, depending on the mode you must run.
Installation and Configuration Parameters: Language Selection	Base language of your Siebel application.
Installation and Configuration Parameters: RDBMS Platform	RDBMS Platform
Installation and Configuration Parameters: OBDC Data Source Name	OBDC Data Source Name
Installation and Configuration Parameters: Database User Name	Database User Name Database Password
Installation and Configuration Parameters: Table Owner	Table Owner Name Table Owner Password
MLOV Parameters: Repository Name	Repository Name
Configuration Parameter Review	Review the parameters you defined, and then click Finish.

Resuming the MLOV Converter Utility If an Error Occurs

If an error occurs, then you can resume running the MLOV Converter Utility in validation mode or in translation mode.

To resume the MLOV Converter Utility if an error occurs

1. Open a DOS prompt, and then navigate to the following directory:

   ```
   ORACLE_HOME\BIN
   ```

   If you use UNIX, then open a shell prompt.
2. If you use Windows, then do one of the following at the command prompt:
   - To resume running in validation mode, type the following command:

     ```
     siebupg /m master_mlov_verify.ucf
     ```
   - To resume running in translation mode, type the following command:

     ```
     siebupg /m master_mlov_translate.ucf
     ```
3. If you use UNIX, then do one of the following at the shell prompt:
   - To resume running in validation mode, type the following command:

     ```
     srvrupgwiz /m master_mlov_verify.ucf
     ```
   - To resume running in translate mode, type the following command:

     ```
     srvrupgwiz /m master_mlov_translate.ucf
     ```

Using the MLOV Converter Utility to Convert Multiple Languages

The MLOV Converter Utility only upgrades one language at a time. If the target columns include data in more than one language, then you must run the utility for each language. For example, assume the ENU and DEU display values exist in a column that is enabled for an MLOV. If you run the converter in ENU, then the utility does the following work:

- For each value that the converter finds, it checks if this value exists as a Display Value for a LOV record of the LOV Type and the language, which in this example is ENU.
- If the value exists, then the converter updates the column with the LIC value.
- In this example, DEU does not exist in the value, and the converter creates a new LOV record for the LOV Type and ENU.

  You do not need these new LOV records, and you must remove them.

To use the MLOV Converter utility to convert multiple languages

1. In Siebel Tools, click the Screens menu, System Administration, and then click the List of Values menu item.
2. In the List of Values list, query the Display Value property for the name of the MLOV you must convert.

   As a result of your query, Siebel Tools displays the display value rows for the MLOV.

3. Set the Multilingual property for each record in the List of Values list using values from the following table.

Property	Value
Multilingual	Does not contain a check mark.

4. In the List of Values list, query the Display Value property for the name of the MLOV you must convert.

   As a result of your query, Siebel Tools displays the header row for the MLOV.
5. Make sure the Multilingual property in the List of Values list does not contain a check mark.
6. Run the MLOV Converter utility for one of the languages you must convert.

   For more information, see *Converting Your Current Data for an MLOV*.
7. Remove the check mark from the Multilingual property for each object you modified in Step 5 and Step 6.

   Note that the MLOV Converter utility adds a check mark to the Multilingual property.
8. Delete the unwanted records that the converter utility created.
9. Repeat Step 2 through Step 7 for each additional language you must convert.

Troubleshooting Problems with an MLOV Conversion

When the MLOV Converter Utility finishes, it writes log files to the following directory:

```
siebsrvr\log\mlov_verify_validation\output or siebsrvr\log\mlov_translate\output
```

It writes errors to the `mlovupgd_verify.log` file.

Fixing an Inconsistently Bounded List of Values or an Improperly Set Translation Table Property

If a List of Values is not bound consistently, or if the Translation Table property is not set to S_LST_VAL, then the utility logs the follow message in the `mlovupgd_verify.log` file:

```
The following Validation checks for:

1- Two or more columns defined in the same LOV domain are inconsistently bounded
(one bounded, one not)

2- Two or more columns are defined in the same LOV domain and at least one of them
does not have a Translation Table Name of S_LST_OF_VAL
```

The utility includes a log entry for each error it encounters. The utility includes the LOV type, column, and table.

To fix an inconsistently bounded list of values or an improperly set Translation Table property

1. Make sure the list is consistently bounded.
2. For more information, see Step 5 in *Configuring a Multilingual List of Values*.
3. Make sure the Translation Table property is set properly.
4. For more information, see Step 5 in *Configuring a Multilingual List of Values*.
5. To make sure you corrected all errors, run the MLOV Converter Utility in validation mode.

Fixing a LOV Domain That Is Not in the S_LST_OF_VAL Table

If a list of values domain is not represented in the S_LST_OF_VAL table, then the utility logs the following message in the `mlovupgd_verify.log` file:

```
The following Validation checks for:
LOV domains in the repository that are not represented in S_LST_OF_VAL
```

In this situation, a list of values domain does reside in the Siebel repository but it fails one of the following tests:

- It is not represented as a value in the list of values table.
- The list of values type is not LOV_TYPE.

This problem can occur in the following situations:

- You delete a record in the list of values table instead of deactivating it. For more information, see *Deactivating an MLOV Record Instead of Deleting It*.
- You enter an incorrect entry in the LOV Type property for a column added using a database extension.

To fix a LOV domain that is not in the S_LST_OF_VAL table

1. Correct the LOV Type property:
   a. In Siebel Tools, correct the entry in the LOV Type property.

      For more information, see *Adding Records for All Supported Languages*.
   b. Test and then deliver your Workspace.

      A script creates a matching record in the list of values table for any values it finds in the target tables that do not include matching records in the list of values table. The script marks these records as inactive.
   c. In the Siebel client, navigate to the Administration - Data screen, and then add language-specific entries for the base records you just compiled.

      This allows Siebel CRM to display the values in the active language.
2. Add the list of values domain:
   a. In the Siebel client, navigate to the Administration - Data screen, and then click the List of Values link.
   b. Add the list of values domain and set the Type field to LOV_TYPE.

Configuring Certain Siebel Modules to Use MLOV Fields

This topic describes how to configure certain Siebel modules, such as Siebel Workflow, to use MLOV fields. It includes the following information:

- *Configuring Siebel Workflow to Use MLOV Fields*
- *Configuring Siebel Assignment Manager to Use MLOV Fields*
- *Configuring Siebel Anywhere to Use MLOV Fields*

Configuring Siebel Workflow to Use MLOV Fields

To determine whether a condition is true, Siebel Workflow compares values in target tables against values in the Business Process administration tables. Siebel Workflow cannot compare the language-independent code to the display value because of the following differences:

- Siebel CRM stores the language-independent code in the MLOV column of the database table.
- Siebel CRM stores the display value in the Business Process Administration table.

To allow Siebel Workflow to work with an MLOV column, you must configure workflow objects so that they compare the language-independent code in the target table with the language-independent code in the Business Process Designer administration table. You must do this for the following objects:

- Conditions for the workflow policy
- Argument for the workflow policy

For more information, see *Siebel Business Process Framework: Workflow Guide* .

Preparing Policy Conditions and Action Arguments for an MLOV

In this topic, you prepare policy conditions and action arguments for an MLOV.

To prepare policy conditions and action arguments for an MLOV

1. In Siebel Tools, display the following object types:
   - Workflow Policy
   - Workflow Policy Program
   - Workflow Policy Program Arg

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. In the Object Explorer, click Business Component.
3. In the Business Components list, locate the relevant business component.
4. In the Object Explorer, expand the Business Component tree, and then click Field.
5. In the Fields list, identify the fields that are enabled for an MLOV.
6. Of the fields that are enabled for an MLOV, identify the fields that reference the workflow policy conditions and action arguments.
7. For each field that references a workflow policy condition, do the following:
   a. *Creating an Applet That Uses Language-Independent Code*.
   b. *Creating a List That Uses Language-Independent Code*.

Creating an Applet That Uses Language-Independent Code

In this topic, you create an applet that uses language-independent code.

To create an applet that uses language-independent code

1. In the Object Explorer, click Applet.
2. In the Applets list, locate an applet that resembles the functionality you require.

   For example, Account Status Pick Applet.
3. Right-click the applet, and then click Copy Record.

4. Set properties using values from the following table.

Property	Value
Name	Append LIC to the name. For example, Account Status Pick Applet LIC.

5. In the Object Explorer, expand the Applet tree, expand the List tree, and then click List Column.
6. In the List Columns list, locate a list column that resembles the functionality you require.
7. Right-click the list column, and then click Copy Record.
8. Set properties using values from the following table.

Property	Value
Name	Name
Field	Name

9. Create a list that uses language-independent code.
   For more information, see *Creating a List That Uses Language-Independent Code*.

Creating a List That Uses Language-Independent Code

In this topic, you create a list that uses language-independent code.

To create a list that uses language-independent code

1. In the Object Explorer, click Pick List.
2. In the Pick Lists list, locate a picklist that resembles the functionality you require.
   For example, Picklist Account Status.
3. Right-click the picklist, and then click Copy Record.
4. Set the properties using values from the following table.

Property	Value
Name	Append LIC to the name. For example, Picklist Account Status LIC.
Sort Specification	Name

5. Configure the workflow policy and workflow policy program argument.
   For more information, see *Configuring the Workflow Policy and Workflow Policy Program Argument*.

Configuring the Workflow Policy and Workflow Policy Program Argument

In this topic, you configure the workflow policy and workflow policy program argument.

To configure the workflow policy and workflow policy program argument

1. Configure the workflow policy:
    a. In the Object Explorer, click Workflow Policy Column.
    b. In the Workflow Policy Columns list, locate the workflow policy column that you must use with an MLOV.
    c. Set the properties for the workflow policy column. Make sure you set them in the order that the following table lists them, starting with the Applet property.

Property	Value
Applet	Choose the applet you created in Step 3 in *Creating an Applet That Uses Language-Independent Code*.
PickList	Choose the picklist you created in Step 3 in *Creating a List That Uses Language-Independent Code*.
Source Field	Name

2. Configure the workflow policy program argument:
    a. In the Object Explorer, click Workflow Policy Program.
    b. In the Workflow Policy Programs list, locate the workflow policy program that contains the argument you must enable for use with an MLOV.
    c. In the Object Explorer, expand the Workflow Policy Program tree, and then click Workflow Policy Program Arg.
    d. In the Workflow Policy Program Arguments list, choose the argument you must enable for use with an MLOV.
    e. Set properties for the argument using values from the table in Step 1.
3. Test and then deliver your Workspace.
4. Administer the values:
    a. Open the Siebel client, navigate to the Administration - Business Process screen, and then click the Workflow Policies link.
    b. In the Policies List, locate the policy you must modify.
    c. In the Conditions List, choose the condition, and then enter the value.
    d. In the Arguments List, choose the argument, enter the value, and then step off the record.

    Siebel CRM stores the language-independent code.

Configuring Siebel Assignment Manager to Use MLOV Fields

To determine whether a condition is true, Siebel Assignment Manager compares values in target tables against values in the Assignment Manager administration tables. Assignment Manager cannot compare the language-independent code to the display value because of the following differences:

- Siebel CRM stores the language-independent code in the MLOV column of the database table.
- Siebel CRM stores the display value in the Assignment Manager administration table.

To allow Assignment Manager to work with an MLOV column, you must configure the criteria values and criteria skills so that they compare the language-independent code in the target table with the language-independent code in the Assignment Manager administration table. This situation is similar to configuring Siebel Workflow. For more information, see *Configuring Siebel Workflow to Use MLOV Fields*.

For more information, see *Siebel Assignment Manager* and *Siebel Assignment Manager Administration Guide* .

Preparing Criteria Values and Criteria Skills for an MLOV

In this topic, you prepare criteria values and criteria skills for an MLOV.

To prepare policy conditions and action arguments for an MLOV

1. Complete *Preparing Policy Conditions and Action Arguments for an MLOV* with the following modifications:
    a. In Step 1 in *Preparing Policy Conditions and Action Arguments for an MLOV*, display the Workflow Policy Column and Assignment Attribute object types.
    b. In Step 6 in *Preparing Policy Conditions and Action Arguments for an MLOV*, identify the fields that reference the following object types:
        - Criteria Values
        - Criteria Skills
        - Workload Rules
2. Set the criteria for each field that references criteria values and criteria skills:
    a. In the Object Explorer, click Assignment Attribute.
    b. In the Assignment Attributes list, locate the assignment attribute that must work with an MLOV field.
    c. Set properties of the assignment attribute using values from the following table.

Property	Value
Translate	Contains a check mark.
Translate Pick Field	Choose the field that stores the language-independent code. The Name field typically stores the language-independent code.

    d. Repeat Step 2 for each field until you configured all criteria.

3. Set the workload rules for each field you identified in Step b:

    a. Create a new list to display language-independent code values.
    b. Create a new applet to display language-independent code values.
    c. Configure the workflow policy column to use the new list and applet.
    d. Choose the values for predefined records.

    Work you perform in this step is similar to work you perform to configure a workflow policy. For more information, see *Preparing Policy Conditions and Action Arguments for an MLOV*.
4. Test and then deliver your Workspace.

    For more information, see *Using Siebel Tools* .

Configuring Siebel Anywhere to Use MLOV Fields

You can configure Siebel CRM to use MLOV fields with Siebel Anywhere. After you complete this task, you can perform typical tasks associated with Siebel Anywhere, such as creating and distributing a Siebel client repository upgrade kit. You must perform more configuration to create and distribute a Siebel client repository upgrade kit. For more information, see *Siebel Anywhere Administration Guide* .

To configure Siebel Anywhere to use MLOV fields

1. In Siebel Tools, in the Object Explorer, click Table.
2. In the Tables list, locate the S_UPG_KIT table.
3. In the Object Explorer, expand the Table tree, and then click Column.
4. In the Columns list, locate the STATUS column, and then set properties using values from the following table.

Property	Value
Translation Table Name	S_LST_OF_VAL

5. Test and then deliver your Workspace.

    For more information, see *Using Siebel Tools* .

26 Configuring Data Visualization

Configuring Data Visualization

This chapter includes information about data visualization and how to create or configure dashboards containing data visualization components. It includes the following topics:

- *About Data Visualization*
- *About Data Visualization Components*
- *About Sample Industry Dashboards*
- *Configuring Data Visualization Components*
- *Creating a New Data Visualization Dashboard*

Note: The data visualization feature is new as of Siebel CRM 22.5 Update.

About Data Visualization

This topic summarizes the data visualization feature. This feature is new as of Siebel CRM 22.5 Update. This release also provides new data visualization components.

Dashboards provide an intuitive approach to consuming aggregated data and insight and are often used to track key performance indicators (KPIs) and other metrics relevant to core business objectives. The data visualization feature provides a framework to help you to build dashboards faster and more efficiently.

The data visualization feature consists of data visualization components and dashboards.

Data Visualization Components

Data visualization components are the building blocks of the sample industry dashboards provided and of any other dashboards you configure. The following data visualization components are provided with this feature:

- **Infolet components.** For more information, see *Infolet Components*.
- **Timeline components.** For more information, see *Timeline Components*.
- **Hierarchy components.** For more information, see *Hierarchy Components*.

For more information, see *About Data Visualization Components* and *Configuring Data Visualization Components*.

Sample Industry Dashboards

Dashboards can include various types of data visualization components. Sample dashboards are provided that have been designed to support users in key roles for the following Siebel CRM industry applications:

- **Siebel Financial Services.** This dashboard provides a 360-degree view of a customer. For more information, see *Siebel Financial Services Dashboard*.

- **Siebel High Tech and Industrial Manufacturing.** This dashboard is an agent dashboard and provides a 360-degree view for an agent. It can be used by any sales or relationship partner. For more information, see *Siebel High Tech and Industrial Manufacturing Dashboard*.
- **Siebel Telecommunications.** This dashboard provides a 360-degree view of an account. For more information, see *Siebel Telecommunications Dashboard*.

In each case, dashboard content supports typical agent tasks and relevant data relating to customers, accounts, and other key entities. The sample dashboards can be adapted for use in other Siebel CRM applications. For more information, see *About Sample Industry Dashboards* and *Creating a New Data Visualization Dashboard*.

Also refer to information about applet visualization and tile visualization in *Configuring Siebel Open UI* .

About Data Visualization Components

This topic describes the data visualization components. Data visualization components are the building blocks of the sample industry dashboards provided and of any other dashboards you configure. The predefined components and sample dashboards can be further extended and customized to match your specific business objectives. The framework supports the ability to add, edit, or delete data visualization components in a dashboard.

This topic contains the following information:

- *Infolet Components*
- *Timeline Components*
- *Hierarchy Components*

Other components, such as charts and calendars, can also be used in dashboards.

Related Topics

Configuring Data Visualization Components

Infolet Components

This topic provides information about infolet components, which can be used in data visualization dashboards. This topic is part of *About Data Visualization Components*.

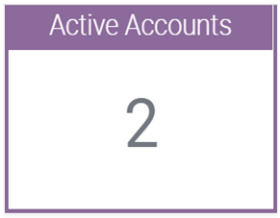
Infolets are applets that provide an ability to display one or more field values (such as Number, Text, or Date), from which a user can drill down to the relevant view, according to customer requirements. Infolets also support database aggregation functionality.

The following types of infolets are provided:

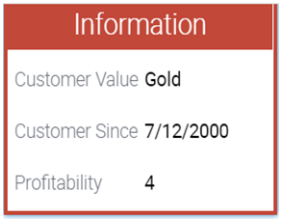
- **Infolets displaying a value for a business component field.** The example infolet shown in the following image displays the Customer Tier field for a contact, with the value Gold. For configuration information, see *Configuring an Infolet to Display a Value for a Field*.



- **Infolets displaying aggregate values.** This type of infolet supports specific database aggregation functionality such as sum, min, max, and avg. This example infolet shown in the following image displays the sum of active accounts for a contact. For configuration information, see *Configuring an Infolet to Display an Aggregate Field Value*.



- **Infolets displaying the values of multiple fields.** The example infolet shown in the following image displays values for the Customer Value, Customer Since, and Profitability fields for a contact. For configuration information, see *Configuring a Form Infolet to Display Values for Multiple Fields*.



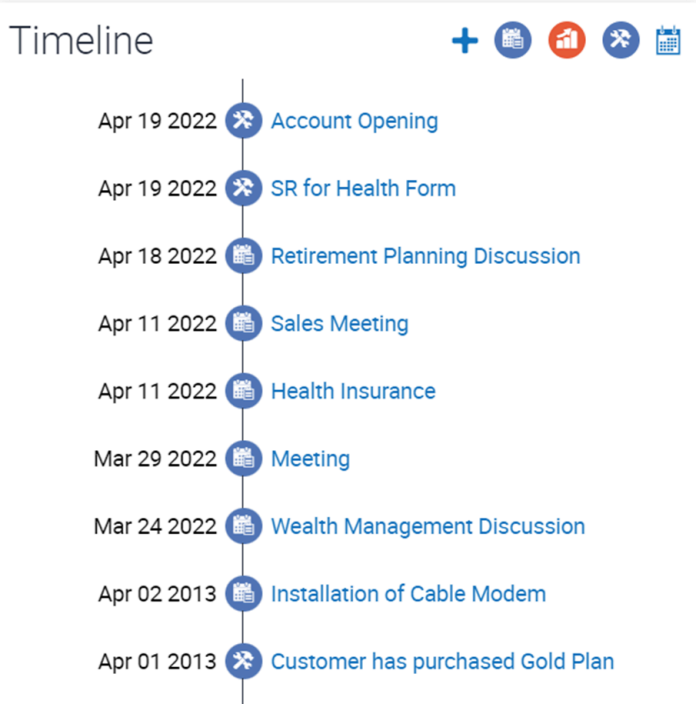
- **Infolets displaying a list of records.** The example infolet shown in the following image displays a list of offers. A drilldown link is also provided. For configuration information, see *Configuring a List Infolet to Display a List of Records*.



Timeline Components

This topic provides information about timeline components, which can be used in data visualization dashboards. This topic is part of *About Data Visualization Components*.

The timeline applet can display various entities (like activities, opportunities, sales orders, and service requests) for a business object, such as Account or Contact. The entities are displayed in reverse chronological order, with the most recent record at the top. Each type of entity is displayed using a different icon. Icons are configurable.



Note the following:

- Clicking the calendar icon allows users to filter the data based on a date range. Users can also filter the data based on the type of entity. The entity filter buttons are toggle buttons. Clicking a button hides entity data from the timeline and the button dims. Clicking the dimmed button displays that entity on the timeline again.
- Clicking the plus button (+) allows the user to enter a new activity. Clicking the + icon in the timeline brings up the Add Activity pop-up applet. Once the activity is added, it appears in the timeline.
- Clicking the description for a timeline item allows the user to drill down and navigate to the Detail view. For example, clicking an opportunity item navigates to the Opportunity Details view, while clicking a sales order navigates to the Sales Order Details view.

For configuration information, see *Configuring Timeline Components*.

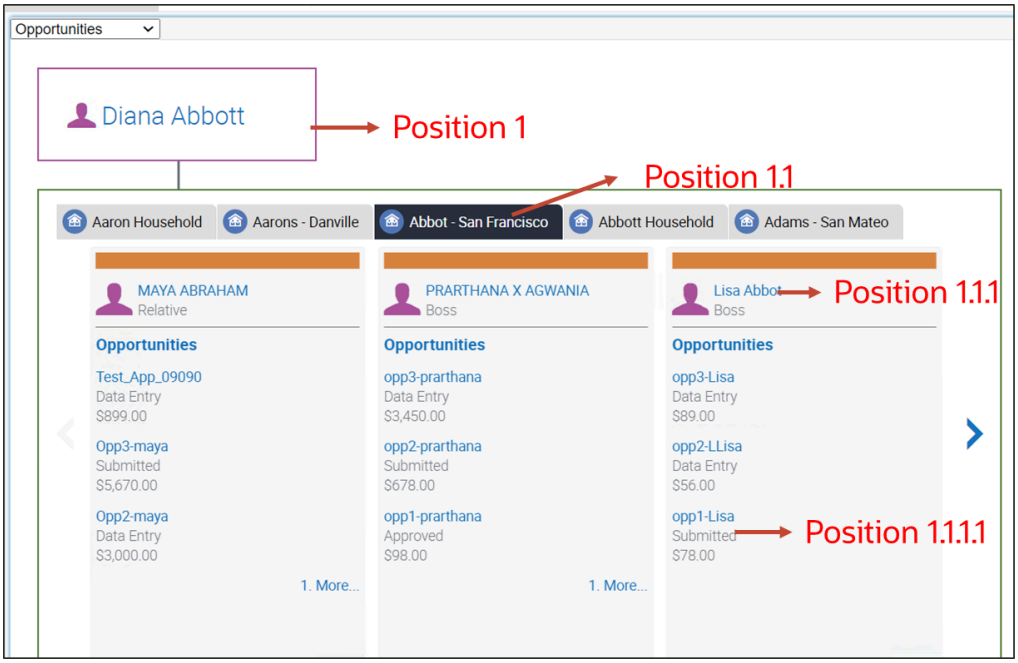
Hierarchy Components

This topic provides information about hierarchy components, which can be used in data visualization dashboards. This topic is part of *About Data Visualization Components*.

A hierarchy applet displays linked data in an ordered four-level hierarchy. The specific form of a hierarchy depends on your use case. For example, a hierarchy might display the following:

- Contact (level 1)
- Household (level 2)
- Household contacts (level 3)
- Opportunities (level 4)

The following figure illustrates such a hierarchy applet.



Note the following:

- In a hierarchy, fields that are mapped to business components can be configured and displayed based on specific requirements.
- A drop-down menu allows the user to select and display level-4 data, such as Opportunities or Service Requests, depending on how you have configured the particular hierarchy.
- If no level-2 hierarchy data exists, then the last level data is displayed, such as Contact and Opportunities.
- As illustrated in the example figure and in sample configurations, positions 1.1.1 and 1.1.1.1 (and 1.2) typically display values from multiple fields. This is done through configuring calculated fields.

For configuration information, see *Configuring Hierarchy Components*.

About Sample Industry Dashboards

This topic describes the sample data visualization dashboards that are prebuilt for use in particular industry applications or can be customized for any application or use case. These dashboards leverage existing and new visual components and provide a summary of the most important information related to specific personas or entities such as customers, agents, or accounts.

This topic contains the following information:

- *Siebel Financial Services Dashboard*
- *Siebel High Tech and Industrial Manufacturing Dashboard*
- *Siebel Telecommunications Dashboard*

Siebel Financial Services Dashboard

This topic provides information about the Siebel Financial Services dashboard. This dashboard provides a 360-degree view of information about a customer (contact). This topic is part of *About Sample Industry Dashboards*.

For more information about Siebel Financial Services, see *Siebel Finance Guide* .

> **Note:** In addition to the configuration information in this topic, see also the topics about configuring each supported data visualization component in *Configuring Data Visualization Components*.

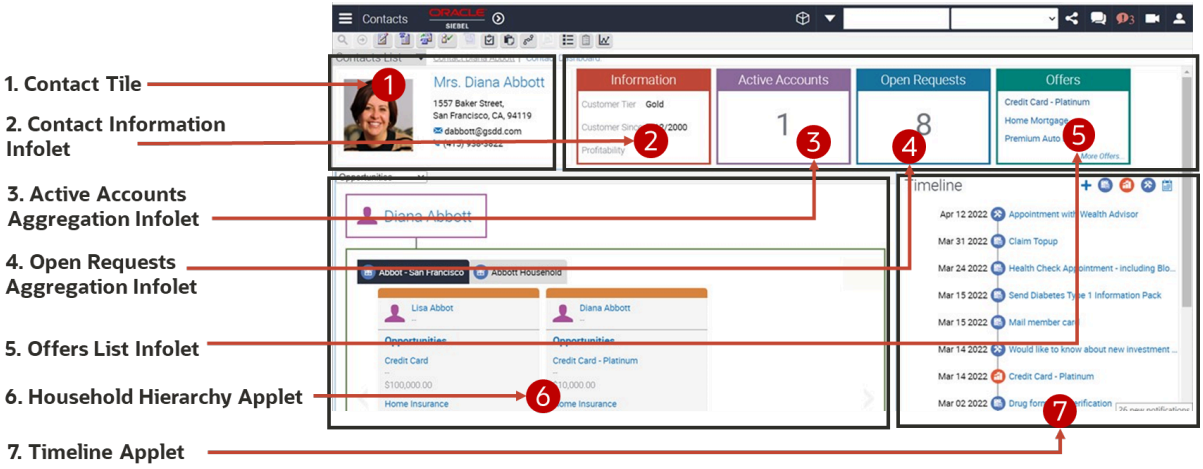
Navigating to the Siebel Financial Services Dashboard

To navigate to the Siebel Financial Services dashboard:

1. Drill down on a contact.
2. Select Dashboard from the third-level menu.

Siebel Financial Services Dashboard

The following image illustrates the Siebel Financial Services dashboard. The following table lists the primary configurable components used to build the Siebel Financial Services dashboard. All of the components listed are part of the Contact business object. The key numbers in the following table correspond to the callouts in the following image.



Key	Control Type	Component Name	Description	Business Component
1	Contact Tile	Contact Information Tile	Contact Address, Email, and Mobile	Contact
2	Form Infolet	Information Infolet	1. Customer Value 2. Customer Since 3. Profitability	Contact
3	Aggregate Infolet	Active Account Infolet	Number of active accounts for Diana Abbott.	FINCORP Account
4	Aggregate Infolet	Open Request Infolet	Number of open service requests.	Service Request
5	List Infolet	Offers Infolet	Three next best offers.	My Contact Offer
6	Hierarchy Applet	Household Hierarchy	Household Hierarchy view, members in household and their opportunities, service requests, and so on.	FINS DB Hierarchy Root Contact FINS DB Hierarchy Household FINS DB Hierarchy Contact FINS DB Hierarchy Opportunity 1 FINS DB Hierarchy Opportunity 2
7	Timeline Applet	Component Name: Contact Timeline	Timeline view of activities, opportunities, and service requests.	Timeline VBC
8	Timeline Activity Pop-up Applet invoked from + icon on Timeline	Timeline New Activity Creation	Used to create a new Activity	Action Copy

Key	Control Type	Component Name	Description	Business Component

Additional Configurable Components in the Siebel Financial Services Dashboard

The following is the configuration for the Contact Information Tile component (item 1 in the first table in this topic).

Component Name	Property Name	Value
Applet	Name	FINS Dashboard Contact Form Applet
Applet	Business Component	Contact
Applet	Class	CSSFrame
Web Template	Name	TOUI Dashboard Account Form Alt
Controls	Control Name	Field
Controls	AddressLine	DBUI Address 1
Controls	AddressLine1	DBUI Address
Controls	EmailAddress	EmailAddress
Controls	Icon	Thumbnail Source Path
Controls	M/M First Name Last Name	M/M First Name Last Name
Controls	WorkPhoneNum	Work Phone #
Controls	Name	Control
Applet Web Template Items	Address Line	DBUI Address 1 Item Identifier: 1203
Applet Web Template Items	Address Line 1	DBUI Address Item Identifier: 1204
Applet Web Template Items	Email	EmailAddress Item Identifier: 1208

Component Name	Property Name	Value
Applet Web Template Items	Image	Thumbnail Source Path Item Identifier: 100
Applet Web Template Items	Name	M/M First Name Last Name Item Identifier: 90
Applet Web Template Items	Work Phone Number	Work Phone # Item Identifier: 1209

The following is the configuration for the drilldown object for the Contact Information Tile component (item 1).

Name	Drill Down View	Hyperlink Field	Business Component
M/M First Name Last Name	Contact Detail View	M/M First Name Last Name	Contact

The following is the configuration for the Open Request Infolet component (item 4).

Component Name	Property Name	Value
Applets	Name	FINS Contact Open Request Applet
Applets	Business Component	Service Request
Applets	Class	CSSSWEFrameInfolet
Applets	Search Specification	[Status]<>LookupValue('SR_STATUS','Closed') AND [Status]<>LookupValue('SR_STATUS', 'Cancelled')
1. Controls - Title Configuration	Name	AppletTitle
1. Controls - Title Configuration	HTML Type	Caption
1. Controls - Title Configuration	Caption - String Reference	SBL_OPEN_REQUESTS
2. Controls - Record Count (Value Configuration)	Name	RecordCount
2. Controls - Record Count (Value Configuration)	HTML Type	Field

Component Name	Property Name	Value
2. Controls - Record Count (Value Configuration)	Caption - String Reference	SBL_RECORD_COUNT-1004232336-5PK
Control User Property	Name	Count Function Pivot Field
Control User Property	Value	Id
3. Controls - LinkCtrl	Name	LinkCtrl
3. Controls - LinkCtrl	Method Invoked	GotoView
3. Controls - LinkCtrl	HTML Type	Link
Control User Property	1. Name: View	Value: FIN Contact Service View
Control User Property	2. Name: OverrideDisplayTextWithData	Value: Record Count
Applet Web Template	Name	Tile
Applet Web Template	Type	Base
Applet Web Template	Web Template	CCAppletInfoletTitle
Applet Web Template	Name	Control
Applet Web Template Items	Applet Title	AppletTitle Item Identifier: 184
Applet Web Template Items	LinkCtrl	LinkCtrl Item Identifier: 501

Note: The timeline pop-up is mapped to Action Copy business component and the end (Planned Completion) date for the record created using Action Copy has no post default value compared to Action business component which has the post default value as System timestamp. Any activity created using the timeline component functionality will not have any end (Planned Completion) date. Hence, such records will not display dates in the timeline component and needs to be updated manually as in the FINS dashboard – timeline data map we have mapped the display date to Planned Completion date.

Siebel High Tech and Industrial Manufacturing Dashboard

This topic provides information about the Siebel High Tech and Industrial Manufacturing dashboard (also known as the HTIM dashboard). This is an agent dashboard and provides a 360-degree view of information relevant to an agent. It can be used by any sales or relationship partner. This topic is part of *About Sample Industry Dashboards*.

For more information about Siebel High Tech and Industrial Manufacturing, see *Siebel High Tech and Industrial Manufacturing Guide* .

> **Note:** In addition to the configuration information in this topic, see also the topics about configuring each supported data visualization component in *Configuring Data Visualization Components*.

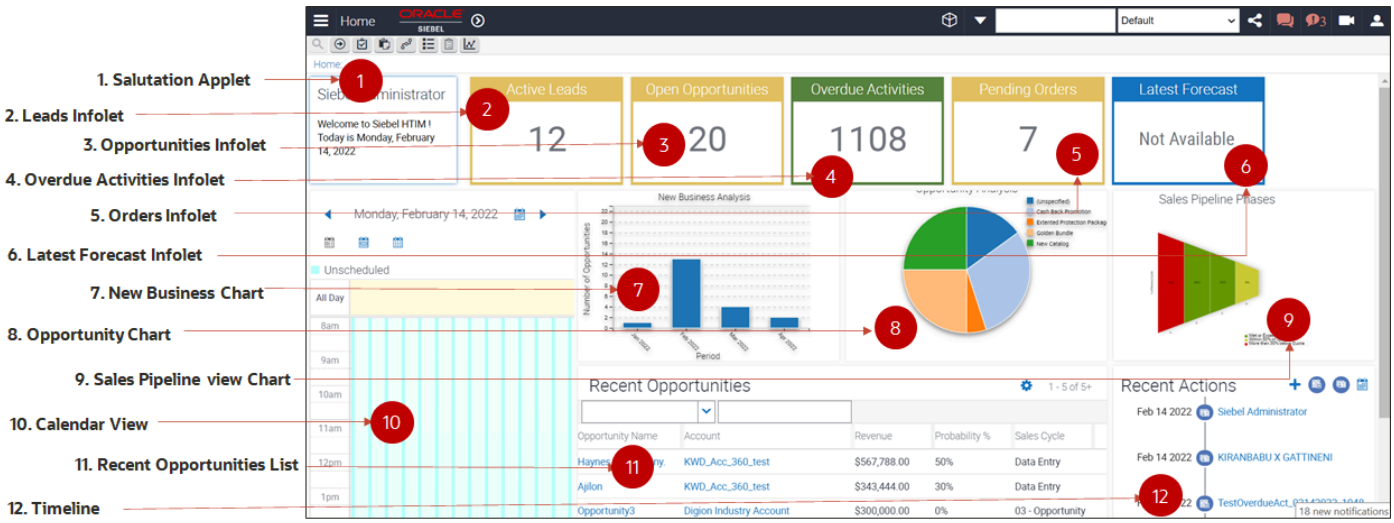
Navigating to the Siebel High Tech and Industrial Manufacturing Dashboard

To navigate to the Siebel High Tech and Industrial Manufacturing dashboard:

1. Navigate to the Site Map.
2. Select Dashboard.

Siebel High Tech and Industrial Manufacturing Dashboard

The following image shows the Siebel High Tech and Industrial Manufacturing dashboard. The following table lists the primary configurable components used to build the Siebel High Tech and Industrial Manufacturing dashboard. All of the components listed are part of the WebCallCenter Home business object. The key numbers in the following table correspond to callouts in the following image.



Key	Control Type	Component Name	Description	Business Component
NA	Screen	HTIM Dashboard Screen	The HTIM Dashboard screen.	Not applicable

Key	Control Type	Component Name	Description	Business Component
NA	View	HTIM Dashboard View	The HTIM Dashboard view having the sample HTIM dashboard.	Not applicable
1	Applet	Contact Salutation Applet	Displays the agent name, a welcome message, and the date.	Salutation (eApps)
2	Aggregate Infolet	Active Leads Infolet	Active leads under the agent.	Lead
3	Aggregate Infolet	Opportunities Count Infolet	Open opportunities count.	Opportunity
4	Aggregate Infolet	Activities Infolet	Overdue activities count.	Activity
5	Aggregate Infolet	Orders Infolet	Number of pending orders count.	Order
6	BC Field Infolet	Latest Forecast Infolet	Latest forecast value.	Forecast
7	Chart	New Business Chart	Number of opportunities across months.	Opportunity
8	Chart	Opportunity Chart	Product opportunity analysis chart.	Opportunity
9	Chart	Sales Pipeline Chart	Sales pipeline information.	Opportunity
10	Calendar	Calendar View	A calendar view with day, week, and month view.	Action
11	List View	Opportunities List	A list of recent opportunities.	Opportunity
12	Timeline applet	Contact Timeline	Timeline of recent activities and contacts updated or added.	Timeline VBC
13	Timeline Activity Pop-up Applet invoked from + icon on Timeline.	Timeline New Activity Creation	Used to create a new Activity	Action Copy

Configuration Example for a Calendar Applet: HTIM Calendar Daily Applet Home Page

The following table shows the configuration for the calendar applet named HTIM Calendar Daily Applet Home Page, which is part of the sample configuration for the Siebel High Tech and Industrial Manufacturing dashboard.

Property Name	Value	Comment
Name	HTIM Calendar Daily Applet Home Page	
Project	eCalendar	
Business Component	Action	
Class	CSSSWEFrameActHICalGrid	
Title	Daily Calendar	
Type	Standard	By default, the type is Standard.
Upgrade Behavior	Admin	This property can be anything, as per the customer's scenario. This is generic applet functionality.

The following tables show how to configure the controls for the calendar applet HTIM Calendar Daily Applet Home Page. The properties are listed in two tables.

Property	Name	Caption	HTML Height	Method Invoked
Value	CalCustomCtrl	N/A	650	N?A
Value	DrillDown	Drill Down	N/A	N/A
Value	QueryAssistant	Query Assistant	30	QueryAssistant

Continuing from the previous table, the following are additional property values for the same controls (listed in the Name column).

Name	Field	Field Retrieval Type	Class	HTML Width
CalCustomCtrl			CSSCoCalCustomCtrl	100
DrillDown	Id	Field Data		
QueryAssistant				

The following table shows how to configure the control user properties for these controls.

Control	Name	Value
CalCustomCtrl	PercentWidth	Y
QueryAssistant	Mode	Edit
QueryAssistant	Popup	Query Assistant

The following table shows how to configure the drilldown objects for the calendar applet HTIM Calendar Daily Applet Home Page.

Property	Name	Hyperlink Field	View	Business Component	Source Field	Destination Field
Value	Action - Detail	Id	eCalendar Detail View	Action	Id	Id
Value	Training Drilldown	Id	eTraining My Class Details View	Training Class Registration	Registration Id	Id

The following table shows how to configure the drilldown destination for the Action - Detail drilldown object.

Drilldown Object	Name	Value	Field	Destination Drilldown Object	Sequence
Action - Detail	Training Drilldown	Y	Is Training Type	Training Drilldown	1

The following table shows how to configure the applet web templates for the calendar applet HTIM Calendar Daily Applet Home Page.

Property	Value
Name	Base
Type	Base
Web Template	Calendar Applet
Upgrade Behavior	Admin

The following table shows the configuration for the applet web template items.

Control Name	Property	Value	Description
Salutation text	Name	CalCustomCtrl	
Salutation text	Control	CalCustomCtrl	
Salutation text	Item Identifier	501	
Salutation text	Type	Control	

The following table shows applet user properties to configure for the calendar applet HTIM Calendar Daily Applet Home Page.

Name	Value
Calendar Type	Siebel
Day Summary Tooltip Template	[Planned] - [Planned Completion] [Type]: [Description]
Description.Tooltip Fields	Type, Description, Planned, Planned Completion, MeetingLocation, Comment
Disable Buscomp Hierarchy	TRUE
Display Field Drilldown Object Names	Action - Detail
Display Fields	Description
Drilldown Object Name	Action - Detail
Enable Date Picker	Y
Enable New Button	N
Enable Print Button	N
Enable Quick Add	N
Enable Timezone Picker	N
Enable Today Button	N
End Date Field	Planned Completion

Name	Value
Home Page Mode	Y
Include All Employee View In Owner Picklist	Y
Owner Id Field	Primary Owner Id
Owner Login Field	Primary Owned By
Private Field	Private
Repeat Expires Field	Repeat Expires
Repeating Field	Repeating
Start Date Field	Planned
Update Field	Description

Configuration Example for a Chart Applet: HTIM Dashboard Chart Applet - Opportunity Analysis

The following is the configuration of the chart applet named HTIM Dashboard Chart Applet - Opportunity Analysis, which is part of the sample configuration for the Siebel High Tech and Industrial Manufacturing dashboard.

Property Name	Value	Comment
Name	HTIM Dashboard Chart Applet - Opportunity Analysis	
Project	HTIM Design Project	
Business Component	Opportunity	
Class	CSSFrameChart	
Title	Opportunity	
Type	Standard	By default, the type is Standard.
Upgrade Behavior	Preserve	This property can be anything, as per the customer's

Property Name	Value	Comment
		scenario. This is generic applet functionality.

The following is the configuration of the controls for the chart applet HTIM Dashboard Chart Applet - Opportunity Analysis.

Property	Name	Caption	HTML Type	Method Invoked	HTML Height	HTML Width
Value	Chart	Chart			250	250

The following is the configuration of a chart for the chart applet HTIM Dashboard Chart Applet - Opportunity Analysis.

Property	Name	Category Captions	Category Field	Data Function	Data Point Field	Type
Value	Created	Product	Product	Count	Name	2dPie

The following is the configuration of a chart element for this chart.

Property	Name	Text	Type
Value	Title	Opportunity Analysis	Title

The following is the configuration of the applet web templates for the chart applet HTIM Dashboard Chart Applet - Opportunity Analysis.

Property	Value
Name	Base
Type	Base
Web Template	Applet Chart Responsive
Upgrade Behavior	Preserve

The following is the configuration of the applet web template items.

Control Name	Property	Value	Description
Chart	Name	Chart	
Chart	Control	Chart	
Chart	Item Identifier	599	
Chart	Type	Control	

Configuration Example for a Chart Applet: HTIM Dashboard Oppty Chart Applet - New Business

The following is the configuration of the chart applet named HTIM Dashboard Oppty Chart Applet - New Business, which is part of the sample configuration for the Siebel High Tech and Industrial Manufacturing dashboard.

Property Name	Value	Comment
Name	HTIM Dashboard Oppty Chart Applet - New Business	
Project	HTIM Design Project	
Business Component	Opportunity	
Class	CSSFrameChart	
Title	New Business	
Type	Standard	By default, the type is Standard.
Upgrade Behavior	Preserve	This property can be anything, as per the customer's scenario. This is generic applet functionality.

The following is the configuration of the control for the chart applet HTIM Dashboard Oppty Chart Applet - New Business.

Property	Name	Caption	HTML Height	HTML Width
Value	Chart	Chart	250	250

The following is the configuration of a chart for the chart applet HTIM Dashboard Oppty Chart Applet - New Business. The properties are listed in two tables.

Property	Name	Category Captions	Category Field	Data Function	Data Point Field
Value	Revenue	Created	Created	Count	Revenue

Continuing from the previous table, the following are additional property values for the same chart (listed in the Name column).

Name	Data Point Captions	Period	Picklist Functions	Picklist Functions	Type
Revenue	Revenue	Month (Calendar)	Number of Opportunities	Number of Opportunities	2dBar

The following is the configuration of the chart elements for this chart. The chart elements are listed in two tables.

Property	Name	Axis ID	Coordinates	Display Format
Value	AxisTitle	XAxis		
Value	Title			

Continuing from the previous table, the following are additional properties for the same chart elements (listed in the Name column).

Name	Divisions	Text	Type	Vertical
Revenue		Period	AxisTitle	
Value		Opportunity Analysis	Title	

The following is the configuration of the applet web templates for the chart applet HTIM Dashboard Oppty Chart Applet - New Business.

Property	Value
Name	Base
Type	Base

Property	Value
Web Template	Applet Chart Responsive
Upgrade Behavior	Preserve

The following is the configuration of the applet web template items.

Control Name	Property	Value	Description
Chart	Name	Chart	
Chart	Control	Chart	
Chart	Item Identifier	599	
Chart	Type	Control	

Configuration Example for a Funnel Chart Applet: HTIM Home Page Funnel Graph Applet

The following is the configuration of the funnel chart applet named HTIM Home Page Funnel Graph Applet, which is part of the sample configuration for the Siebel High Tech and Industrial Manufacturing dashboard.

Property Name	Value	Comment
Name	HTIM Home Page Funnel Graph Applet	
Project	Visualization	
Business Component	Opportunity	
Class	CSSFrameFunnelChart	
Title	Sales Pipeline Phases	
Search Specification	[Close Date] > (Today ())	
Type	Standard	By default, the type is Standard.

Property Name	Value	Comment
Upgrade Behavior	Admin	This property can be anything, as per the customer's scenario. This is generic applet functionality.

The following is the configuration of the controls for the funnel chart applet HTIM Home Page Funnel Graph Applet. The controls are listed in two tables.

Property	Name	Caption	HTML Type	Method Invoked
Value	AppletTitle	Sales Pipeline Phases		
Value	Error1	There is no data for the chart (SBL-CHT-00102)		
Value	FunnelChart		JavaApplet	

Continuing from the previous table, the following are additional property values for the same controls (listed in the Name column).

Name	HTML Height	HTML Width	Class
AppletTitle			
Error1			
FunnelChart	350	300	CSSJavaFunnelChart

The following is the configuration of a chart for the funnel chart applet HTIM Home Page Funnel Graph Applet. The chart is listed in two tables.

Property	Name	Category Captions	Category Field	Data Function	Data Point Field
Value	Sales Stage	Sales Stage	Sales Stage	Sum	Revenue

Continuing from the previous table, the following are additional properties of the same chart (listed in the Name column).

Name	Picklist Function	Picklist Function Captions	Series Field	Series Captions
Sales Stage	% of Count Quota,% of Revenue Quota	% of Count Quota,% of Revenue Quota	Sales Method	Sales Method

The following is the configuration of the chart elements for this chart.

Property	Name	Axis ID	List of Values	Coordinates	Text	Type
Value	Title				Sales Pipeline Phases	Title
	XAxisTitle	XAxis	List Of Values, Value,Type = 'SALES_STAGE_PHASE_TYPE', Order By			AxisLabel
	XAxisTitle	XAxis			Phase	AxisTitle
	ZAxisLabel	ZAxis		70100	Met or Exceeded Quota, Within 30% of Quota, More than 30% below Quota	AxisLabel

The following is the configuration of the chart element locales for this chart.

Chart Element	Language Code	Translate
Title	ENU	Yes
XAxisTitle	ENU	Yes

The following is the configuration of the applet web templates for the funnel chart applet HTIM Home Page Funnel Graph Applet.

Property	Value
Name	Base
Type	Base
Web Template	Applet Chart Responsive With Title

Property	Value
Upgrade Behavior	Admin

The following is the configuration of the applet web template items.

Control Name	Property	Value	Description
AppletTitle	Name	AppletTitle	
AppletTitle	Control	AppletTitle	
AppletTitle	Item Identifier	90	
AppletTitle	Type	Control	
FunnelChart	Name	FunnelChart	
FunnelChart	Control	FunnelChart	
FunnelChart	Item Identifier	599	
FunnelChart	Type	Control	
Error1	Name	Error1	
Error1	Control	Error1	
Error1	Item Identifier	598	
Error1	Type	Control	

To map the physical renderer for the funnel chart applet HTIM Home Page Funnel Graph Applet:

1. Navigate to Administration - Application screen and then to the Manifest Administration view.

   You must register the new applet (such as HTIM Home Page Funnel Graph Applet) with physical renderer settings. This task is needed for automation support and for supporting a custom CSS style class.
2. In the UI Objects list, create a new record with values like the following:

Type	Usage Type	Name
Applet	Physical Renderer	<AppletName>

Type	Usage Type	Name

3. In the Object Expression list, create a new record with the following value:

Level
1

4. In the Files list, create new records with the following values:

Level	Name
1	siebel/salespipelinerenderer.js
1	siebel/chartspmodel.js

Configuration Example for a List Applet: HTIM Opportunity List Applet - No Buttons

The following is the configuration of the list applet named HTIM Opportunity List Applet - No Buttons, which is part of the sample configuration for the Siebel High Tech and Industrial Manufacturing dashboard.

Property Name	Value	Comment
Name	HTIM Opportunity List Applet - No Buttons	
Project	HTIM Design Project	
Business Component	Opportunity	
Class	CSSSWEFrameList	
Title	Opportunities	
Type	Standard	By default, the type is Standard.
Upgrade Behavior	Preserve	This property can be anything, as per the customer's scenario. This is generic applet functionality.
Search Specification	[Close Date] >= (Today ())	

Property Name	Value	Comment
Associate Applet	Opportunity Assoc Applet	
HTML Number of Rows	5	

The following is the configuration of the controls for the list applet HTIM Opportunity List Applet - No Buttons. The controls are listed in two tables.

Property	Name	Caption	HTML Type	Method Invoked
Value	GotoNextSet		RecNavNxt	GotoNextSet
Value	GotoPreviousSet		RecNavPrv	GotoPreviousSet
Value	PositionOnRow		PositionOnRow	PositionOnRow

Continuing from the previous table, the following are additional properties of the same controls (listed in the Name column).

Name	HTML Disabled Bitmap	HTML Bitmap	HTML Icon Map
GotoNextSet	RECNAV _NEXTSET_OFF	RECNAV_ NEXTSET_ON	
GotoPreviousSet	RECNAV_PREVSET_ OFF	RECNAV_PREVSET_ ON	
PositionOnRow	ROW_ON	ROW_OFF	

The following is the configuration of the drilldown objects for the list applet HTIM Opportunity List Applet - No Buttons. The drilldown objects are listed in two tables.

Property	Name	Hyperlink Field	View	Business Component
Value	Line of Business	Name	Opportunity Detail - Contacts View	Opportunity
Value	Primary Account	Account	Opportunity Detail - Contacts View	Account

Continuing from the previous table, this table lists additional properties of the same drilldown objects (listed in the Name column).

Name	Menu Text	Sequence	Source Field	Destination Field
Line of Business	Go To Products	2		Id
Primary Account		14	Account Id	

The following is the configuration of the applet web templates for the list applet HTIM Opportunity List Applet - No Buttons.

Property	Value
Name	Base
Type	Edit List
Web Template	Applet List (Base/EditList)
Upgrade Behavior	Preserve

The following is the configuration of the applet web template items.

Control Name	Control	Type	Item Identifier	Expression
Account	Account	List Item	503	
GotoNextSet	GotoNextSet	Control	123	
GotoPreviousSet	GotoPreviousSet	Control	122	
Name	Name	List Item	502	
PositionOnRow	PositionOnRow	Control	144	
Primary Revenue Amount	Primary Revenue Amount	List Item	506	
Primary Revenue Win Probability	Primary Revenue Win Probability	List Item	507	
Sales Cycle	Sales Cycle	List Item	508	

Control Name	Control	Type	Item Identifier	Expression

Configuration Example for a Timeline Applet: HTIM Dashboard TimeLine VBC Applet

The following is the configuration of the timeline applet named HTIM Dashboard TimeLine VBC Applet, which is part of the sample configuration for the Siebel High Tech and Industrial Manufacturing dashboard.

Property Name	Value	Comment
Name	HTIM Dashboard TimeLine VBC Applet	
Project	HTIM Design Project	
Business Component	Timeline VBC	
Class	CSSSWEFrameList	
Title	Recent Actions	
Type	Standard	By default, the type is Standard.
Upgrade Behavior	Preserve	This property can be anything, as per the customer's scenario. This is generic applet functionality.
HTML Number of Rows	12	

The following is the configuration of the controls for the timeline applet HTIM Dashboard TimeLine VBC Applet.

Property	Name	Caption	HTML Type	Method Invoked	Runtime	Show Popup
Value	ActionFilter	Activity	MiniButton	Filter|Action Copy		
Value	AppletTitle		Text			
Value	ContactFilter	Contact	MiniButton	Filter|Contact		
Value	DateFilterButton	Date	MiniButton	ShowPopup	yes	yes
Value	List		Text			

Property	Name	Caption	HTML Type	Method Invoked	Runtime	Show Popup
Value	New Button	New	MiniButtonNew	ShowPopup	yes	yes

The following is the configuration of the control user properties for these controls.

Control	Name	Value
ActionFilter	ClientPMUserProp	Filter Image
ActionFilter	Filter Image	siebui-icon-activities_icon
AppletTitle	KeepContext	TRUE
AppletTitle	View	Account Detail - Orders View
ContactFilter	ClientPMUserProp	Filter Image
ContactFilter	Filter Image	siebui-icon-contacts_icon
DateFilterButton	Mode	New
DateFilterButton	Popup	Dashboard Timeline Date Filter PopUp Applet
NewButton	Mode	New
NewButton	Popup	Dashboard Timeline Activity PopUp Applet

The following is the configuration of the drilldown objects for the timeline applet HTIM Dashboard TimeLine VBC Applet.

Property	Name	Hyperlink Field	View	Business Component	Source Field	Destination Field
Value	GotoActivities	Description	Activity Attachment View	Action	Artifact Id	Activity Id
Value	GotoContact	Description	Contact Detail View	Contact	Artifact Id	Id

The following is the configuration of the drilldown destinations for these drilldown objects.

Drilldown Object	Name	Value	Field	Destination Drilldown Object	Sequence
GotoActivities	GotoContact	Contact	Type	GotoContact	1

The following is the configuration of the list and list map (in List Columns).

Name	Field	HTML Type	Display Name
Artifact Id	Artifact Id	Text	Id
Date	Date	PlainText	Date
Description	Description	Text	Description
Type	Type	Text	Type
Type Image	Type Image	Text	Type Image

The following is the configuration of the applet web templates for the timeline applet HTIM Dashboard TimeLine VBC Applet.

Property	Value
Name	Edit List
Type	Edit List
Web Template	Applet Vertical TimeLine
Upgrade Behavior	Preserve

The following is the configuration of the applet web template items.

Control Name	Control	Type	Item Identifier
ActionButton	ActionFilter	Control	133
ContactButton	ContactFilter	Control	134
Date	Date	List Item	501
DateFilterButton	DateFilterButton	Control	156

Control Name	Control	Type	Item Identifier
Description	Description	List Item	503
NewButton	NewButton	Control	155
Type	Type Image	List Item	502

The following is the configuration of the applet user properties for the timeline applet HTIM Dashboard TimeLine VBC Applet.

Name	Value
CanInvokeMethod: NewQuery	N
Data Map	HTIM Dashboard TimeLine DataMap

To map the physical renderer for the timeline applet HTIM Dashboard TimeLine VBC Applet:

1. Navigate to Administration - Application screen and then to the Manifest Administration view.

   You must register the new applet (such as HTIM Dashboard TimeLine VBC Applet) with physical renderer settings. This task is needed for automation support and for supporting a custom CSS style class.
2. In the UI Objects list, create a new record with values like the following:

Type	Usage Type	Name
Applet	Physical Renderer	<AppletName>

3. In the Object Expression list, create a new record with the following value:

Level
1

4. In the Files list, create a new record with the following values:

Level	Name
1	siebel/timelinerenderer.js

Navigate to the Administration - Application screen and then to the Data Map Administration view. Configure the data map for the timeline applet HTIM Dashboard TimeLine VBC Applet.

Note: Data Maps can be Workspace enabled in your development environment. If they have been Workspace enabled in your development environment and you are working in that environment, then you can only modify them in an editable Workspace. You do not need an editable Workspace to create and edit Data Maps in your Production environment.

Name	Source Business Object	Destination Business Object
HTIM Dashboard TimeLine DataMap	WebCallCenter Home	WebCallCenter Home

The following is the configuration of the data map components for this data map.

Name	Source Business Object	Destination Business Object	Advanced Options
Action	Action Copy	Timeline VBC	1. Merge To Destination = Y 2. Source Search Specification = ([Updated By] = LoginId() or [Created By] = LoginId()) AND [Updated] > (Today() - 7)
Contact	Contact	Timeline VBC	1. Merge To Destination = Y 2. Source Search Specification = ([Updated By] = LoginId() or [Created By] = LoginId()) AND [Updated] > (Today() - 7)

The following is the configuration of the data map fields for Action.

Source Type	Source	Destination Type	Destination
Expression	"Action"	Field	Type
Expression	"siebui-icon-activities_icon"	Field	Type Image
Field	Activity Id	Field	Artifact Id
Expression	IIF([Description] IS NOT NULL, [Description],[Id])	Field	Description
Field	Updated	Field	Date

The following is the configuration of the data map fields for Contact.

Source Type	Source	Destination Type	Destination
Expression	"Contact"	Field	Type
Expression	"siebui-icon-contacts_icon"	Field	Type Image
Field	Full Name	Field	Description
Field	Id	Field	Artifact Id
Field	Updated	Field	Date

Note: Due to Siebel limitation of having the same business component on 2 independent applets in the same view, we have used Action Copy for Timeline and Action for Calendar in HTIM dashboard. In the sample dashboard, timeline applet maps to Action Copy business component and calendar maps to Action business component due to which the activity added using the timeline applet will not be reflected in the calendar.

Configuration Example for a Salutation Applet: Salutation Applet HTIM

The following is the configuration of the salutation applet named Salutation Applet HTIM, which is part of the sample configuration for the Siebel High Tech and Industrial Manufacturing dashboard.

Property Name	Value	Comment
Name	Salutation Applet HTIM	
Project	HTIM Design Project	
Business Component	Salutation (eApps)	
Class	CSSFrameSalutation	
Title	Salutation	
Type	Standard	By default, the type is Standard.
Upgrade Behavior	Preserve	This property can be anything, as per the customer's scenario. This is generic applet functionality.

The following is the configuration of the applet web templates for the salutation applet Salutation Applet HTIM.

Property	Value
Name	Base
Type	Base
Web Template	Applet App Homepage Banner
Upgrade Behavior	Preserve (this will automatically come once the applet is refreshed)

The following is the configuration of the applet web template items.

Control Name	Property	Value	Description
Salutation text	Name	Salutation text	
Salutation text	Control	Explorer	
Salutation text	Item Identifier	90	
Salutation text	Type	Control	

The following is the configuration of the controls for the salutation applet Salutation Applet HTIM.

Property	Name	Caption	HTML Height	HTML Type	Method Invoked
Value	AppletGraphics	Salutation		Caption	GotoView
Value	Explorer			Text	

To map the physical renderer for the salutation applet Salutation Applet HTIM:

1. Navigate to Administration - Application screen and then to the Manifest Administration view.

   You must register the new applet (such as Salutation Applet HTIM) with physical renderer settings. This task is needed for automation support and for supporting a custom CSS style class.
2. In the UI Objects list, create a new record with values like the following:

Type	Usage Type	Name
Applet	Physical Renderer	<AppletName>

Type	Usage Type	Name

3. In the Object Expression list, create a new record with the following value:

Level
1

4. In the Files list, create new records with the following values:

Level	Name
1	siebel/salutationpr.js

The following is the configuration of a rule set and rules for the salutation applet Salutation Applet HTIM. Do this configuration in the Administration - Personalization screen, then the Applets view. First create rule set Employee HTIM Salutation, then create individual rules.

The following is the configuration of Rule1 through Rule5.

Rule1	Value
Name	NamePrefix1
Rule Type	Expressions
Sequence	1
Active	Yes
Conditional Expression	(GetProfileAttr ('Full Name') IS NOT NULL)
Include Expression	" " + GetProfileAttr ("Me.M/M") + GetProfileAttr ("Me.First Name") + " " + GetProfileAttr ("Me.Last Name") + ""

Rule2	Value
Name	NamePrefix2
Rule Type	Expressions

Rule2	Value
Sequence	2
Active	Yes
Include Expression	" " + LookupValue("SALUTATION","SIEBEL_HTIM") + " "+"!"

Rule3	Value
Name	DatePrefix
Rule Type	Expressions
Sequence	3
Active	Yes
Include Expression	" " + LookupValue("SALUTATION","DATE_PREFIX") + " "

Rule4	Value
Name	Date
Rule Type	Invoke Method
Sequence	4
Active	Yes
Method Name	LongDate

Rule5	Value
Name	DateSuffix
Rule Type	Expressions
Sequence	5

Rule5	Value
Active	Yes
Include Expression	" "

Note: After adding rules, click Menu in the applet and choose Reload Personalization Rules.

The following is the configuration of lists of values. Do this configuration in the Administration - Data screen, then the Lists of Values view.

Lists of Values	Value
Type	SALUTATION
Display Value	Welcome to HTIM Dashboard
Language- Independent Code	SIEBEL_HTIM
Sequence	19
Active	Yes
Translate	Yes

Note: After adding lists of values, click Clear Cache.

Configuration Example for an Infolet Applet: HTIM Lead Infolet Applet

The following is the configuration of the infolet applet named HTIM Lead Infolet Applet, which is part of the sample configuration for the Siebel High Tech and Industrial Manufacturing dashboard.

Property Name	Value	Comment
Name	HTIM Lead Infolet Applet	
Project	HTIM Design Project	
Business Component	PUB Lead	
Class	CSSSWEFrameInfolet	The infolet framework class, which invokes object manager aggregation methods and

Property Name	Value	Comment
		constructs the CSS class based on the business component mapped to the applet.
Title	Active Leads	For example, the title might be Active Leads. This is not the title that is shown in the infolet.
Type	Standard	By default, the type is Standard.
Upgrade Behavior	Preserve	This property can be anything, as per the customer's scenario. This is generic applet functionality.

The following is the configuration of the applet web templates for the infolet applet HTIM Lead Infolet Applet. (Use this information for all other infolets described in this topic.)

Property	Value
Name	Base
Type	Base
Web Template	CCAppletInfoletTile
Upgrade Behavior	Preserve (this will automatically come once the applet is refreshed)

The following is the configuration of the applet web template items.

Control Name	Property	Value	Description
AppletTitle	Name	AppletTitle	
AppletTitle	Control	AppletTitle	
AppletTitle	Item Identifier	184	184 is the item identifier specified in the infolet web template (infolet framework) for mapping the applet title.
AppletTitle	Type	Control	
LinkCtrl	Name	LinkCtrl	

Control Name	Property	Value	Description
LinkCtrl	Control	LinkCtrl	
LinkCtrl	Item Identifier	501	501 is the item identifier specified in the infolet web template (infolet framework) for mapping the value within the infolet.
LinkCtrl	Type	Control	

The following is the configuration of the controls for the infolet applet HTIM Lead Infolet Applet.

Property	Name	Caption	HTML Type	Method Invoked	HTML Height	HTML Width
Value	AppletTitle	Active Leads	Caption			
Value	LinkCtrl		Link	GotoView		
Value	Record Count	Record Count	Field		30	30

The following is the configuration of the control user properties for these controls.

Control	Name	Value
LinkCtrl	OverrideDisplayTextWithData	Record Count
LinkCtrl	View	PUB Leads List View - Read Only
Record Count	Count Function Pivot Field	Id

Configuration Example for an Infolet Applet: HTIM Opportunity Count Infolet Applet

The following is the configuration of the infolet applet named HTIM Opportunity Count Infolet Applet, which is part of the sample configuration for the Siebel High Tech and Industrial Manufacturing dashboard.

Property Name	Value	Comment
Name	HTIM Opportunity Count Infolet Applet	

Property Name	Value	Comment
Project	HTIM Design Project	
Business Component	Opportunity	The business component from where the data is aggregated. For example, Opportunity.
Class	CSSSWEFrameInfolet	The infolet framework class, which invokes object manager aggregation methods and constructs the CSS class based on the business component mapped to the applet.
Title	Open Opportunities	For example, the title might be Open Opportunities. This is not the title that is shown in the infolet.
Search Specification	[Close Date] >= (Today ())	The search specification of the applet that is used for the aggregate function.
Type	Standard	By default, the type is Standard.
Upgrade Behavior	Preserve	This property can be anything, as per the customer's scenario. This is generic applet functionality.

Review the configuration for the applet web templates and applet web template items for the infolet applet HTIM Opportunity Count Infolet Applet. (For details, see the information for the infolet applet HTIM Lead Infolet Applet, earlier in this topic.)

The following is the configuration of the controls for the infolet applet HTIM Opportunity Count Infolet Applet.

Property	Name	Caption	HTML Type	Method Invoked	HTML Height	HTML Width
Value	AppletTitle	Open Opportunities	Caption			
Value	LinkCtrl		Link	GotoView		
Value	Record Count	Record Count	Field		30	30

The following is the configuration of the control user properties for these controls.

Control	Name	Value
LinkCtrl	OverrideDisplayTextWithData	Record Count
LinkCtrl	View	Opportunity List View
Record Count	Count Function Pivot Field	Id

Configuration Example for an Infolet Applet: HTIM Overdue Activity Infolet Applet

The following is the configuration of the infolet applet named HTIM Overdue Activity Infolet Applet, which is part of the sample configuration for the Siebel High Tech and Industrial Manufacturing dashboard.

Property Name	Value	Comment
Name	HTIM Overdue Activity Infolet Applet	
Project	HTIM Design Project	
Business Component	Action	The business component from where the data is aggregated. For example, Action.
Class	CSSSWEFrameInfolet	The infolet framework class, which invokes object manager aggregation methods and constructs the CSS class based on the business component mapped to the applet.
Title	Overdue Activities	For example, the title might be Overdue Activities. This is not the title that is shown in the infolet.
Search Specification	[Planned] <= Today () AND EXISTS ([Owned By] = LoginName())	The search specification of the applet that is used for the aggregate function.
Type	Standard	By default, the type is Standard.
Upgrade Behavior	Preserve	This property can be anything, as per the customer's scenario. This is generic applet functionality.

Review the configuration for the applet web templates and applet web template items for the infolet applet HTIM Overdue Activity Infolet Applet. (For details, see the information for the infolet applet HTIM Lead Infolet Applet, earlier in this topic.)

The following is the configuration of the controls for the infolet applet HTIM Overdue Activity Infolet Applet.

Property	Name	Caption	HTML Type	Method Invoked	HTML Height	HTML Width
Value	AppletTitle	Overdue Activities	Caption			
Value	LinkCtrl		Link	GotoView		
Value	Record Count	Record Count	Field		30	30

The following is the configuration of the control user properties for these controls.

Control	Name	Value
LinkCtrl	OverrideDisplayTextWithData	Record Count
LinkCtrl	View	Activity List View
Record Count	Count Function Pivot Field	Id

Configuration Example for an Infolet Applet: HTIM Pending Orders Infolet Applet

The following is the configuration of the infolet applet named HTIM Pending Orders Infolet Applet, which is part of the sample configuration for the Siebel High Tech and Industrial Manufacturing dashboard.

Property Name	Value	Comment
Name	HTIM Pending Orders Infolet Applet	
Project	HTIM Design Project	
Business Component	Order Entry - Orders	The business component from where the data is aggregated. For example, Order Entry - Orders.
Class	CSSSWEFrameInfolet	The infolet framework class, which invokes object manager aggregation methods and constructs the CSS class based

Property Name	Value	Comment
		on the business component mapped to the applet.
Title	Pending Orders	For example, the title might be Pending Orders. This is not the title that is shown in the infolet.
Search Specification	[Status] <> LookupValue (FS_ORDER_STATUS, Closed) AND [Status] <> LookupValue (FS_ORDER_STATUS, Cancelled) AND [Order Type] = LookupValue (FS_ORDER_TYPE, "Sales Order") AND [Order Date]<=Today()	The search specification of the applet that is used for the aggregate function.
Type	Standard	By default, the type is Standard.
Upgrade Behavior	Preserve	This property can be anything, as per the customer's scenario. This is generic applet functionality.

Review the configuration for the applet web templates and applet web template items for the infolet applet HTIM Pending Orders Infolet Applet. (For details, see the information for the infolet applet HTIM Lead Infolet Applet, earlier in this topic.)

The following is the configuration of the controls for the infolet applet HTIM Pending Orders Infolet Applet.

Property	Name	Caption	HTML Type	Method Invoked	HTML Height	HTML Width
Value	AppletTitle	Pending Orders	Caption			
Value	LinkCtrl		Link	GotoView		
Value	Record Count	Record Count	Field		30	30

The following is the configuration of the control user properties for these controls.

Control	Name	Value
LinkCtrl	OverrideDisplayTextWithData	Record Count
LinkCtrl	View	Order Entry - All Orders View (Sales)

Control	Name	Value
Record Count	Count Function Pivot Field	Id

Configuration Example for an Infolet Applet: HTIM Forecast Latest Infolet Applet

The following is the configuration of the infolet applet named HTIM Forecast Latest Infolet Applet, which is part of the sample configuration for the Siebel High Tech and Industrial Manufacturing dashboard.

Property Name	Value	Comment
Name	HTIM Forecast Latest Infolet Applet	
Project	HTIM Design Project	
Business Component	Forecast 2000 -- Forecast	The business component from where the data is aggregated. For example, Action.
Class	CSSSWEFrameInfolet	The infolet framework class, which invokes object manager aggregation methods and constructs the CSS class based on the business component mapped to the applet.
Title	Latest Forecast	For example, the title might be Latest Forecast. This is not the title that is shown in the infolet.
Search Specification	[Owner Login] = LoginName() AND [Forecast View Mode] <> LookupValue('FCST_VISIBILITY_ 'My Revenues - Indirect Sales')	The search specification of the applet that is used for the aggregate function.
Type	Standard	By default, the type is Standard.
Upgrade Behavior	Preserve	This property can be anything, as per the customer's scenario. This is generic applet functionality.

Review the configuration for the applet web templates and applet web template items for the infolet applet HTIM Forecast Latest Infolet Applet. (For details, see the information for the infolet applet HTIM Lead Infolet Applet, earlier in this topic.)

The following is the configuration of the controls for the infolet applet HTIM Forecast Latest Infolet Applet.

Property	Name	Caption	HTML Type	Method Invoked	HTML Height	HTML Width
Value	AppletTitle	Latest Forecast	Caption			
Value	LinkCtrl		Link	GotoView		
Value	Latest Forecast	Latest Forecast	Field		30	30

The following table shows how to configure the control user properties for these controls.

Control	Name	Value
LinkCtrl	OverrideDisplayTextWithData	Latest Forecast
LinkCtrl	View	Forecast 2000 -- Forecast View

Siebel Telecommunications Dashboard

This topic provides information about the Siebel Telecommunications dashboard. This dashboard provides a 360-degree view of information about an account. This topic is part of *About Sample Industry Dashboards*.

For more information about Siebel Telecommunications, see *Siebel Communications Guide* .

Note: In addition to the configuration information in this topic, see also the topics about configuring each supported data visualization component in *Configuring Data Visualization Components*.

Navigating to the Siebel Telecommunications Dashboard

To navigate to the Siebel Telecommunications dashboard:

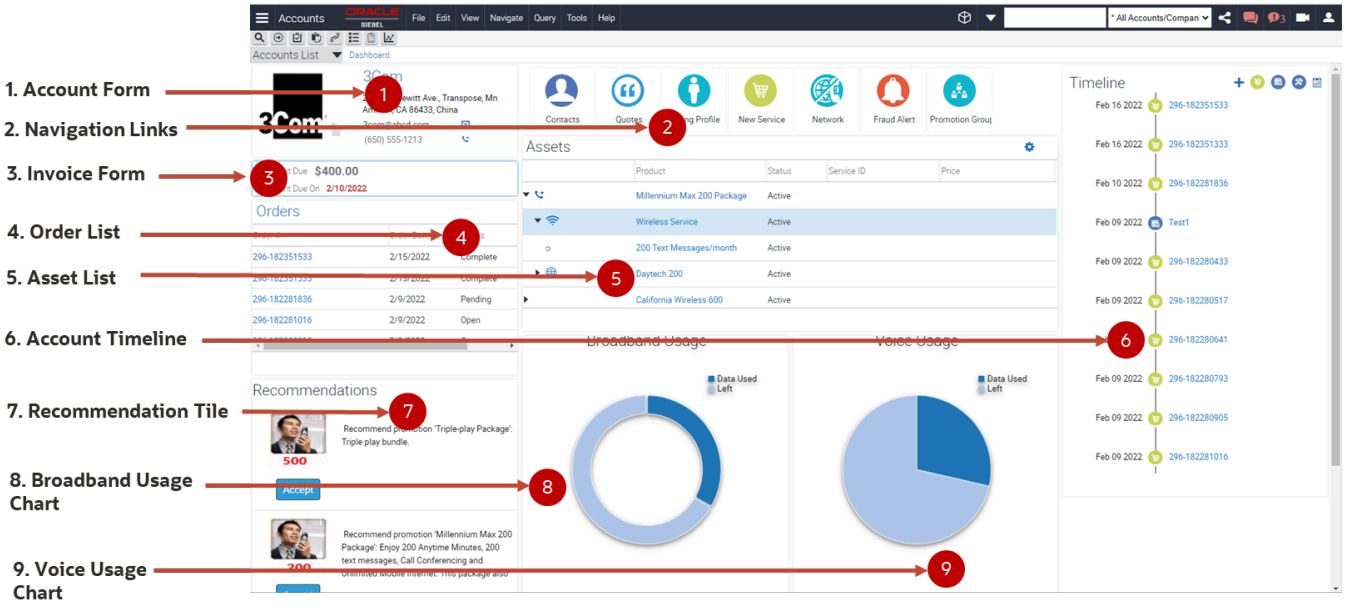
1. Drill down on a customer account.
2. Select Dashboard from the third-level menu.

or

1. Select Customer Directory in the application-level menu.
2. Query for the customer account and drill down on the account number.

Siebel Telecommunications Dashboard

The following image illustrates the Siebel Telecommunications dashboard. The following table lists the primary configurable components used to build the Siebel Telecommunications dashboard. All of the components listed are part of the Account business object. The key numbers in the following table correspond to callouts in the following image.



Key	Control Type	Component Name	Description	Business Component
1	Form	Account Details	Account Name, Address, Contract.	Account
2	Navigation Link	Navigation Link	Contact, Quote, Billing Profile, New Service, Network, Fraud Alert, Promotion Groups.	Navigation Links Runtime
3	Timeline Applet	Account Timeline	Timeline view of sales orders, activities, and service requests.	Timeline VBC
4	Form	Invoice	Last open invoice details.	FS Invoice
5	List	Asset List	Installed asset hierarchy.	TOUI Dashboard Installed Asset
6	List	Order List	List of active orders.	Order Entry - Orders
7	Tile	Recommendation	Recommendations for an account.	UMF Passive Message Virtual BusComp
8	Chart	Broadband Usage	Chart representation of the broadband usage.	TOUI Dashboard Broadband Usage VBC
9	Chart	Voice Usage	Chart representation of the broadband usage.	TOUI Dashboard Voice Usage VBC

Key	Control Type	Component Name	Description	Business Component
10	Timeline Activity Pop-up Applet invoked from + icon on Timeline.	Timeline New Activity Creation	Used to create a new Activity	Action Copy

Additional Configurable Components in the Siebel Telecommunications Dashboard

The following is the view configuration for the Siebel Telecommunications dashboard.

Name	Value	Comment
View Name	TOUI Dashboard UI View	
Business Object	Account	
View Template	TOUI Dashboard UI Alt Mobile View Template	
View Presentation Mode	siebel/appletsliderviewpm.js	Manifest entry for View PM
View Physical Renderer	siebel/appletsliderviewpr.js List	Manifest entry for View PR

The following is the view user property configuration for the Siebel Telecommunications dashboard.

Name	Value	Comment
ClientPMUserProp	UISliderControl 1,UISliderControl 2, UISliderControl 3, UISliderControl 4	
UISliderControl 2	ShowCart:showcart:icon:siebui-icon-controls:target:.siebui-db-section-3:oper:toggleClass:siebui-visible	For Order applet as icon in lower resolutions
UISliderControl 3	ShowTimeline:showtimeline:icon:siebui-icon-cart-alt:target:.siebui-db-telco .siebui-db-section-1 .siebui-db-subsection-3:oper:toggleClass:siebui-visible	For Time Line applet as icon in lower resolutions
UISliderControl 4	ShowBill:showbill:icon:siebui-icon-bill:target:.siebui-db-telco .siebui-db-section-2 .siebui-db-subsection-2 .siebui-db-subsection-2-col3:oper:toggleClass:siebui-visible	For Bill applet as icon in lower resolutions

Configuring Charts for the Siebel Telecommunications Dashboard

In order to add charts to the Siebel Telecommunications dashboard, perform the following steps:

1. In the Siebel CRM application, navigate to the Administration - Configuration screen.
2. Click Parameters in the tab drop-down.
3. Click Advanced, then query for *Siebel File System*.
4. Make a note of the current value for the Siebel File System parameter, such as C:\fs.
5. Copy the following XML files to the Siebel File System location indicated by the parameter value:
    - BroadbandResponse.xml
    - VoiceResponse.xml

    Note: The contents of BroadbandResponse.xml and VoiceResponse.xml are described later this topic.
6. Note the current account's Row Id.
7. Modify BroadbandResponse.xml. Change the ServiceAccountId in both of the TOUIDashboardBroadbandUsageVBC records to the current account's Row Id.
8. To display Broad Band Usage Chart for an additional account, add two more TOUIDashboardBroadbandUsageVBC records, then modify the ServiceAccountId field with another account's Row Id.
9. Modify VoiceReponse.xml. Change the ServiceAccountId in both of the TOUIDashboardVoiceUsageVBC records to the current account's Row Id.
10. To display Voice Usage Chart for an additional account, add two more TOUIDashboardVoiceUsageVBC records, then modify the ServiceAccountId field with another account's Row Id.

Note: This procedure is used to simulate or demonstrate how the charts framework sources data from external sources. Modify these steps as appropriate for your requirements and to get the charts from a location applicable to your environment.

BroadbandResponse.xml

The following is the default content of BroadbandResponse.xml:

```
<?xml version="1.0" encoding="UTF-8"?><SiebelMessage
 MessageId=""
 IntObjectName="TOUI Dashboard Broadband Response IO"
 MessageType="Integration Object"
 IntObjectFormat="Siebel Hierarchical"
><ListOfTOUIBroadbandResponseIO
><TOUIDashboardBroadbandUsageVBC
><Type
>Data Used</Type
><AccountName
>JS_TEST_1</AccountName
><AccountId
>88-1W5M4J</AccountId
><AmountDue
></AmountDue
><BillNumber
></BillNumber
><BillPOID
></BillPOID
><BillPayment
>36</BillPayment
><BillPeriod
></BillPeriod
```

```
><BillingProfileId
></BillingProfileId
><BillingProfileName
></BillingProfileName
><CurrencyCode
></CurrencyCode
><DueDate
></DueDate
><Value
>100</Value
><ServiceAccountId
>88-26CND</ServiceAccountId
></TOUIDashboardBroadbandUsageVBC
><TOUIDashboardBroadbandUsageVBC
><Type
>Left</Type
><AccountName
>JS_TEST_1</AccountName
><AccountId
>88-1W5M4J</AccountId
><AmountDue
></AmountDue
><BillNumber
></BillNumber
><BillPOID
></BillPOID
><BillPayment
></BillPayment
><BillPeriod
></BillPeriod
><BillingProfileId
></BillingProfileId
><BillingProfileName
></BillingProfileName
><CurrencyCode
></CurrencyCode
><DueDate
></DueDate
><Value
>80</Value
><ServiceAccountId
>88-26CND</ServiceAccountId
></TOUIDashboardBroadbandUsageVBC>
</ListOfTOUIBroadbandResponseIO
></SiebelMessage>
```

VoiceResponse.xml

The following is the default content of VoiceResponse.xml:

```
<?xml version="1.0" encoding="UTF-8"?><SiebelMessage
 MessageId=""
 IntObjectName="TOUI Dashboard Voice Response IO"
 MessageType="Integration Object"
 IntObjectFormat="Siebel Hierarchical"
><ListOfTOUIVoiceResponseIO
><TOUIDashboardVoiceUsageVBC
><Type
>Data Used</Type
><AccountName
>JS_TEST_1</AccountName
><AccountId
>88-1W5M4J</AccountId
><AmountDue
></AmountDue
```

```
><BillNumber
></BillNumber
><BillPOID
></BillPOID
><BillPayment
></BillPayment
><BillPeriod
></BillPeriod
><BillingProfileId
></BillingProfileId
><BillingProfileName
></BillingProfileName
><CurrencyCode
></CurrencyCode
><DueDate
></DueDate
><Value
>800</Value
><ServiceAccountId
>88-26CND</ServiceAccountId
></TOUIDashboardVoiceUsageVBC
><TOUIDashboardVoiceUsageVBC
><Type
>Left</Type
><AccountName
>JS_TEST_1</AccountName
><AccountId
>88-1W5M4J</AccountId
><AmountDue
></AmountDue
><BillNumber
></BillNumber
><BillPOID
></BillPOID
><BillPayment
>36</BillPayment
><BillPeriod
></BillPeriod
><BillingProfileId
></BillingProfileId
><BillingProfileName
></BillingProfileName
><CurrencyCode
></CurrencyCode
><DueDate
></DueDate
><Value
>2000</Value
><ServiceAccountId
>88-26CND</ServiceAccountId
></TOUIDashboardVoiceUsageVBC
></ListOfTOUIVoiceResponseIO
></SiebelMessage
>
```

Configuring Data Visualization Components

This topic provides information about the configuration tasks for configuring data visualization components. It contains the following information:

- Configuring infolet components:

- *Configuring an Infolet to Display a Value for a Field*
- *Configuring an Infolet to Display an Aggregate Field Value*
- *Configuring a Form Infolet to Display Values for Multiple Fields*
- *Configuring a List Infolet to Display a List of Records*
- *Additional Configuration Tasks for All Infolets*

- *Configuring Timeline Components*
- *Configuring Hierarchy Components*

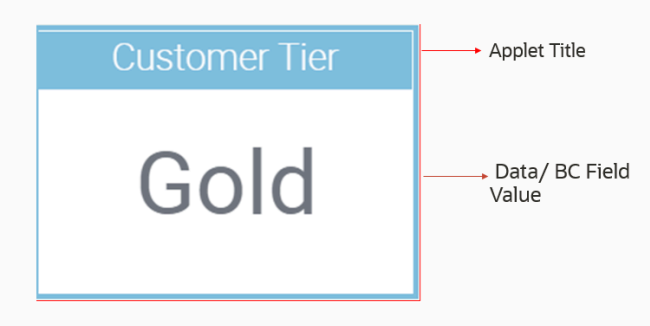
Related Topics

About Data Visualization Components

About Sample Industry Dashboards

Configuring an Infolet to Display a Value for a Field

This topic describes how to configure an infolet to display a value for a business component field – for example, the following image shows the Customer Tier infolet with the value Gold. This topic is part of *Configuring Data Visualization Components*.



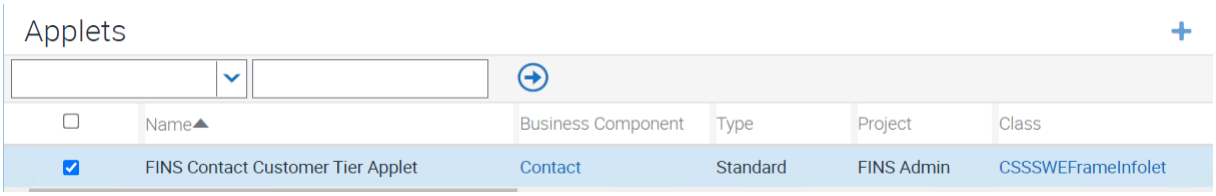
To create a new applet and configure properties

1. Create a new workspace.
2. Create a new applet.
3. Configure the applet properties as shown in the following table – note that this is the generic applet configuration.

Property Name	Value	Comment
Name	<AppletName>	The name of the applet. In this example, the applet name is FINS Contact Customer Tier Applet.
Business Component	<BusComp>	The name of the business component whose data is to be displayed. The business component must be defined in

Property Name	Value	Comment
		the business object specified for any view that uses this applet.
Type	Standard	By default, the type is Standard.
Class	CSSSWEFrameInfolet	The name of a C++ class used to manage the applet. This is the infolet framework class for displaying a single field value. The framework constructs the CSS class based on the business component mapped to the applet.

The following image shows this sample configuration for the FINS Contact Customer Tier Applet.



4. For this new applet, navigate to Controls, then create and configure a control for the applet title configuration. In this example, the caption is Customer Tier.

> **Note:** In this configuration example, you create a total of three controls: AppletTitle, Customer Tier, and LinkCtrl.

Property	Value	Comment
Name	<Title Control Name>	In the sample applet, the control name is AppletTitle.
Caption - String Reference	<Symbolic String of Applet Title>	The infolet title to be displayed. The Caption property is populated based on this mapping. In this example, the caption is Customer Tier.
HTML Type	Caption	The HTML type for the AppletTitle control. This must be Caption.

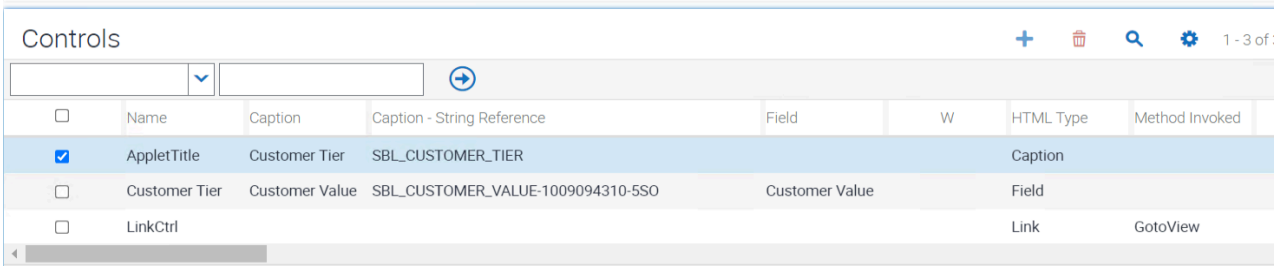
5. For this applet, create and configure the Customer Tier with the properties shown in the following table.

Property	Value	Comment
Name	<Control Name>	The name of the control. In the sample applet, the control name is Customer Tier. This control name is used in the control user property OverrideDisplayTextWithData.
Field	<BC Field Name>	The business component field, which is mapped at the applet level. This field value is exposed in the infolet. In the sample applet, the Customer Value field is mapped to show Customer Tier information of a Contact.
HTML Type	Field	The HTML type for the Customer Tier control. This must be Field.

6. For this applet, create and configure the LinkCtrl control for the drilldown configuration with the properties shown in the following table.

Property	Value	Comment
Name	LinkCtrl	The name of the control.
HTML Type	Link	The HTML type for the LinkCtrl control. This must be Link.
Method Invoked	GotoView	The method invoked when the link is clicked.

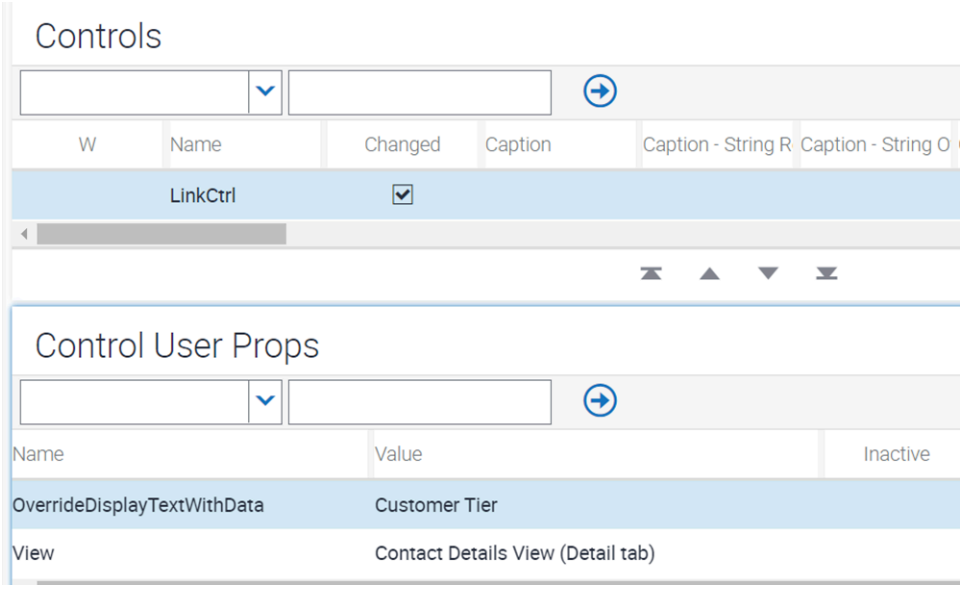
The following image shows this sample configuration for controls for the FINS Contact Customer Tier Applet.



7. Configure user properties for the LinkCtrl control with the properties shown in the following table.

Property	Value	Comment
View	<Destination View Name>	User is navigated to this destination view on drilldown on the value in the infolet.
OverrideDisplayTextWithData	<Control Name>	The control that is configured with the business component field value.

The following image shows this sample configuration for the LinkCtrl control and its user properties



To map the applet web template and web template items

1. For this applet (FINS Contact Customer Tier Applet), create a new applet web template.
2. Configure the applet web template properties as shown in the following table – note that this is the generic applet web template configuration for this type of infolet.

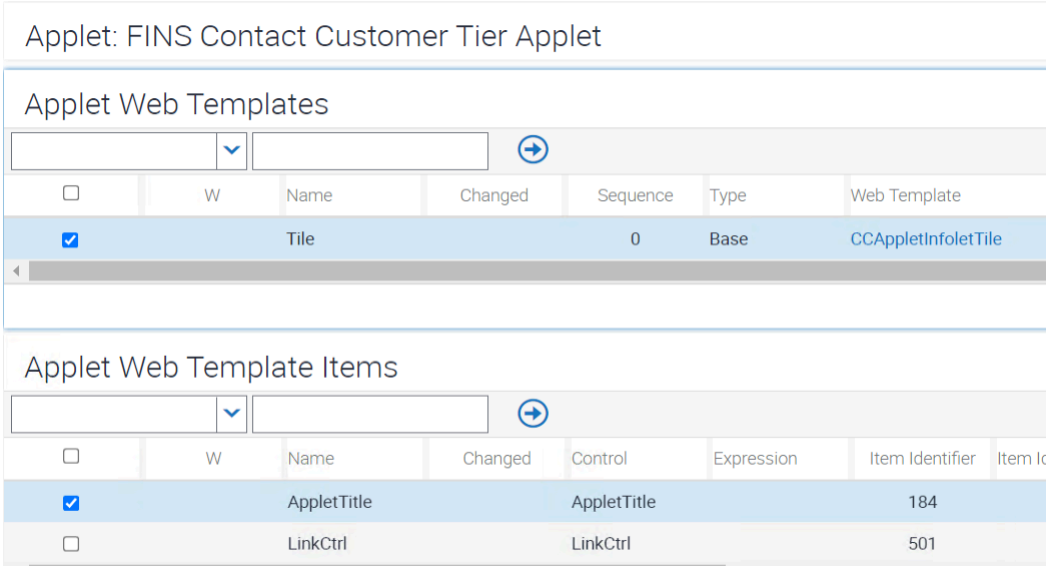
Property	Value	Comment
Name	<Name>	The name of the applet web template.
Type	Base	The type of applet web template. In this case, the Type must be Base.
Web Template	CCAppletInfoletTile	The type of web template for this applet web template. It must be CCAppletInfoletTile.

Property	Value	Comment

3. For the new applet web template, navigate to Applet Web Template Item, then create applet web template items with the values shown in the following table.

Name	Control	Item Identifier	Description
<Web Template Item 1>	<Title Control Name>	184	<Title Control Name> is the control created for showing the applet title. 184 is the item identifier specified in an infolet web template for mapping the applet title.
<Web Template Item 2>	LinkCtrl	501	You configured the LinkCtrl control in a prior procedure. 501 is the item identifier specified in an infolet web template for mapping the value within the infolet.

The following image shows this sample configuration for the applet web template and web template items, for the FINS Contact Customer Tier infolet.



To map the presentation model

1. Navigate to Administration - Application screen, then the Manifest Administration view.
   You must register the new applet (such as FINS Contact Customer Tier Applet) with presentation model settings. This task is needed for automation support and for supporting a custom CSS style class for infolets.
2. In the UI Objects list, create a new record with values shown in the following table.

Type	Usage Type	Name
Applet	Presentation Model	<AppletName>

3. In the Object Expression list, create a new record with the value shown in the following table.

Level
1

4. In the Files list, create a new record with the values shown in the following table.

Level	Name
1	siebel/infoletpm.js

To map the physical renderer

1. Navigate to Administration - Application screen, then to the Manifest Administration view.
   You must register the new applet (such as FINS Contact Customer Tier Applet) with physical renderer settings. This task is needed for automation support and for supporting a custom CSS style class for infolets.
2. In the UI Objects list, create a new record with values shown in the following table.

Type	Usage Type	Name
Applet	Physical Renderer	<AppletName>

3. In the Object Expression list, create a new record with the value shown in the following table.

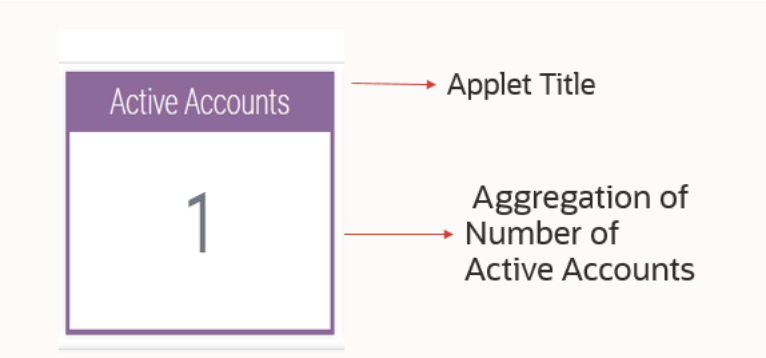
Level
1

4. In the Files list, create a new record with the values shown in the following table.

Level	Name
1	siebel/infoletpr.js

Configuring an Infolet to Display an Aggregate Field Value

This topic describes how to configure an infolet to display an aggregate field value – for example, the following image shows the Active Accounts infolet with the value 1 (which is the aggregate number of active accounts). This topic is part of *Configuring Data Visualization Components*.



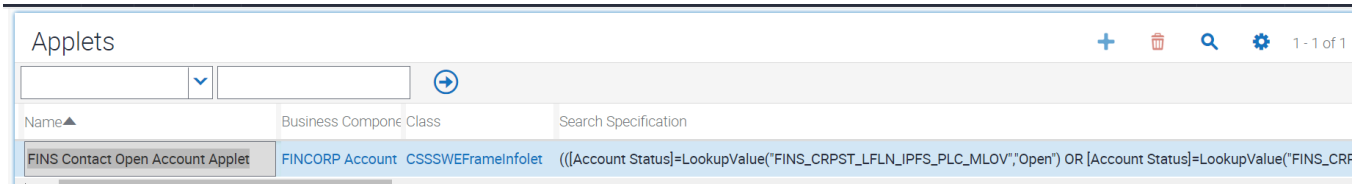
To configure an infolet to display an aggregate field value

1. Create a new workspace.
2. Create a new applet.
3. Configure the applet properties as shown in the following table – note that this is the generic applet configuration:

Property Name	Value	Comment
Name	<AppletName>	The name of the applet. In this example, the applet name is FINS Contact Open Account Applet.
Business Component	<BusComp>	The name of the business component (for example FINCORP Account whose data is to be displayed. The business component must be defined in the business object specified for any view that uses this applet.
Class	CSSSWEFrameInfolet	The name of a C++ class used to manage the applet. This is the infolet framework

Property Name	Value	Comment
		class for displaying a single field value (aggregate in this case). The framework invokes the Object Manager aggregation methods and constructs the CSS class based on the business component mapped to the applet.
Search Specification	<Search Spec>	The search specification to be applied on the infolet.
Type	Standard	By default, the type is Standard.

The following image shows this sample configuration for the FINS Contact Open Account Applet. In this sample, the search specification considers only open accounts.



4. For this new applet, navigate to Controls, then create and configure a control for the applet title configuration. In this example, the caption is Active Accounts. For details, see *Configuring an Infolet to Display a Value for a Field*.

> **Note:** In this configuration example, you create a total of three controls: AppletTitle, Record Count, and LinkCtrl.

5. For this applet, create and configure the Record Count control (for example) as shown in the following table.

Property	Value	Comment
Name	<Record Count>	The name of the control. In the sample applet, the control name is Record Count. This control name is used in the control user property OverrideDisplayTextWithData.
HTML Type	Field	The HTML type for the Record Count control. This must be Field.

6. Configure the user properties for the Record Count control as shown in the following table.

Control User Property	Value	Comment
Name	Count Function Pivot Field	This aggregate function is used to get the record count, such as the number of accounts. Multiple aggregate functions are mentioned later in this topic.
Value	Id	This is the field on which the aggregation function is applied.

7. For this applet, create and configure the LinkCtrl control as shown in the following table for the drilldown configuration.

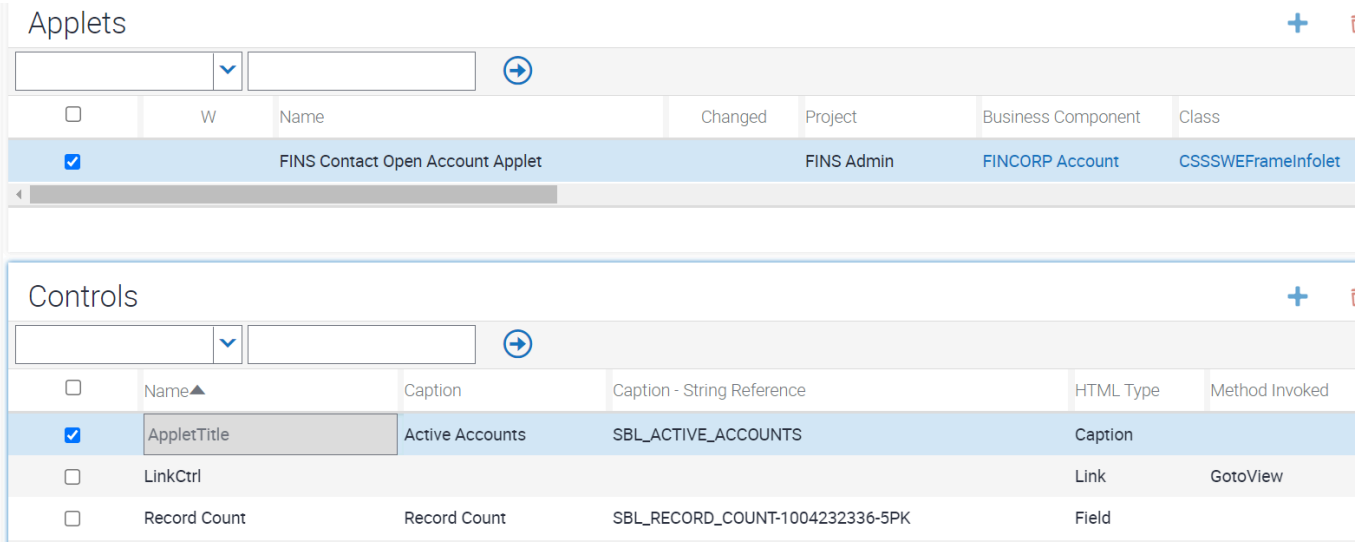
Property	Value	Comment
Name	LinkCtrl	The name of the control. This must be LinkCtrl.
HTML Type	Link	The HTML type for the LinkCtrl control. This must be Link.
Method Invoked	GotoView	The method invoked when the link is clicked.

8. Configure user properties for the LinkCtrl control as shown in the following table.

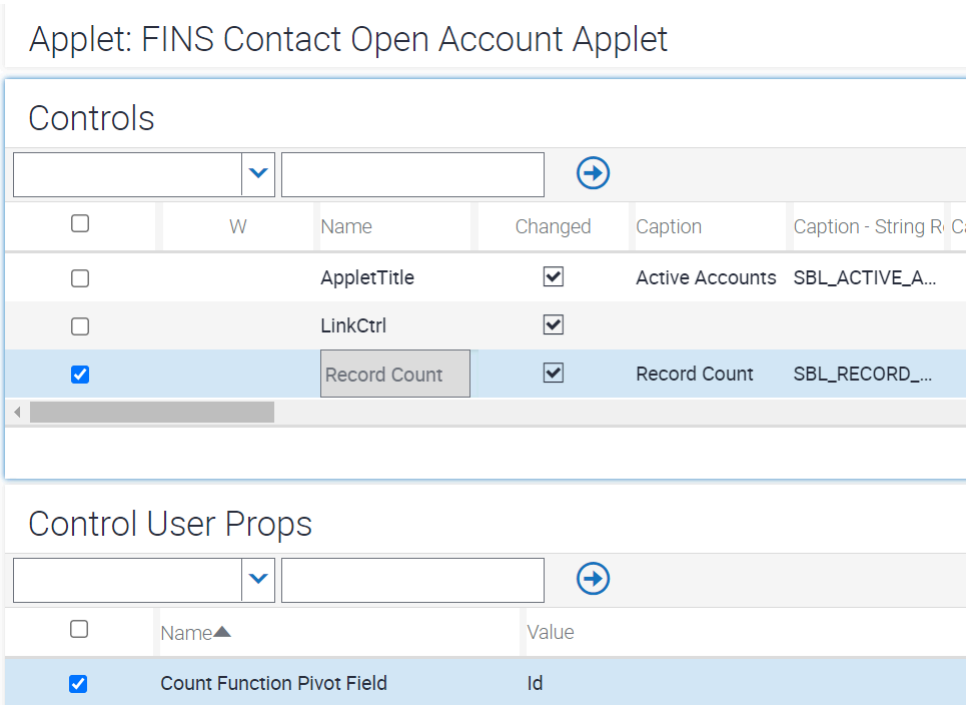
Property	Value	Comment
View	<Destination View Name>	User is navigated to this destination view on drilldown on the value in the infolet.
OverrideDisplayTextWithData	<Record Count>	The control (such as Record Count) that is configured with the business component field value.

Property	Value	Comment

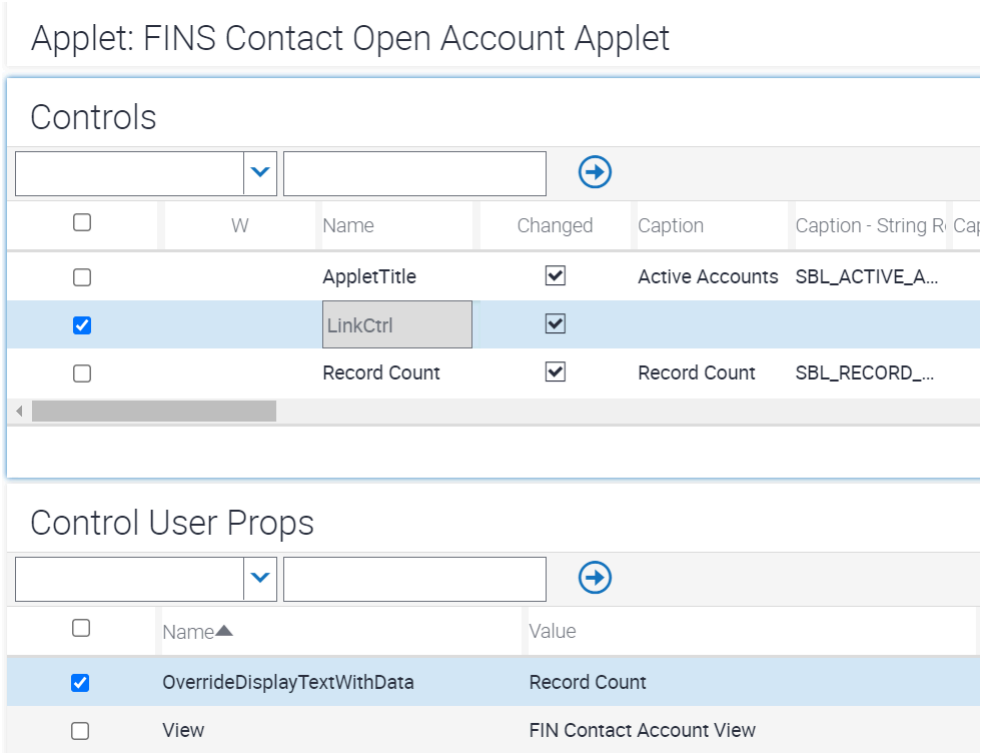
The following image shows the control configured (AppletTitle) for the FINS Contact Open Account Applet.



The following image shows the configuration for the Record Count control and its user property:



The following is a sample configuration for the LinkCtrl control and its user properties:



9. Map the applet web template and web template items.

   For details, see the relevant procedure in *Configuring an Infolet to Display a Value for a Field*.
10. Map the presentation model and the physical renderer.

    You must register the new applet (such as FINS Contact Open Account Applet) with presentation model and physical renderer settings. These tasks are needed for automation support and for supporting a custom CSS style class for infolets. For details, see the relevant procedures in *Configuring an Infolet to Display a Value for a Field*.

About Aggregate Functions Supported for Infolets

The following table lists the aggregate functions that are supported for infolets. Set the appropriate user property on the relevant applet control. In the sample configuration, the Count Function Pivot Field property is set on the Record Count control for FINS Contact Open Account Applet. Note that database aggregate functions are not supported on calculated fields.

Aggregate Function	Control User Property Name	Value
Average	Avg Function Pivot Field	Field name that requires the Avg aggregate function.
Sum	Sum Function Pivot Field	Field name that requires the Sum aggregate function.
Min	Min Function Pivot Field	Field name that requires the Min aggregate function.

Aggregate Function	Control User Property Name	Value
Max	Max Function Pivot Field	Field name that requires the Max aggregate function.
Count	Count Function Pivot Field	Field name that requires the Count aggregate function.

About Aggregate Values and Search Specifications

Also note the following about configuring an infolet to display an aggregate value:

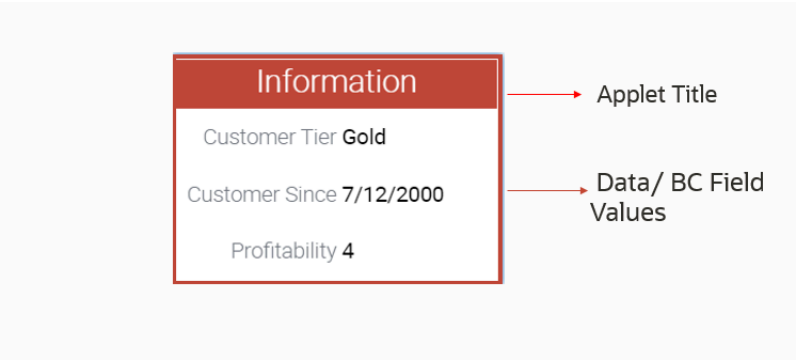
- If there is any discrepancy in the aggregate value on the infolet, then review the search specification that is applied on the infolet business component. For example, a source business object predefined query (PDQ) is applied along with the search specification on the infolet.
- When you drill down on the aggregate value link, and if the aggregate value on the infolet does not match the count of records displayed in the destination applet, then review the destination search specification that is applied on the infolet search specification.

If you encounter either of these cases, then refer to the following equations to reconcile the value in the infolet and the record list in the destination applet after a drilldown:

- **Source business object PDQ + infolet search specification.** The aggregate value on the infolet.
- **Infolet search specification + destination search specification.** The list of records displayed in the destination applet after drilldown.

Configuring a Form Infolet to Display Values for Multiple Fields

This topic describes how to configure a form infolet to display values for multiple fields for a business component – for example, the following image shows the Information infolet with the following values: Customer Tier: Gold, Customer Since: 7/12/2020, Profitability: 4. This topic is part of *Configuring Data Visualization Components*. In this example, one control is created for the applet title and three fields (the maximum) are configured for display in the form infolet.



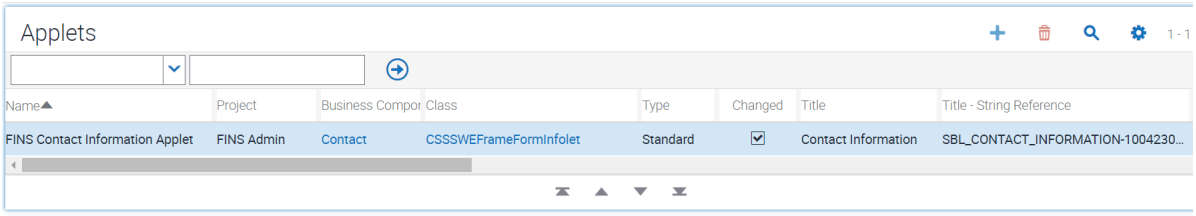
To configure a form infolet to display three field values

1. Create a new workspace.
2. Create a new applet.

3. Configure the applet properties as shown in the following table – note that this is the generic applet configuration:

Property Name	Value	Comment
Name	<AppletName>	The name of the applet. In this example, the applet name is FINS Contact Information Applet.
Business Component	<BusComp>	The name of the business component containing the fields whose data is to be displayed. The business component must be defined in the business object specified for any view that uses this applet.
Class	CSSSWEFrameFormInfolet	The name of a C++ class used to manage the applet. This is the infolet framework class for showing multiple field values. The framework constructs the CSS class based on the business component mapped to the applet.
Type	Standard	By default, the type is Standard.

The following image shows this sample configuration for the FINS Contact Information Applet:



4. For this new applet, navigate to Controls, then create and configure a control for the applet title configuration. In this example, the caption is Information. For details, see *Configuring an Infolet to Display a Value for a Field*.

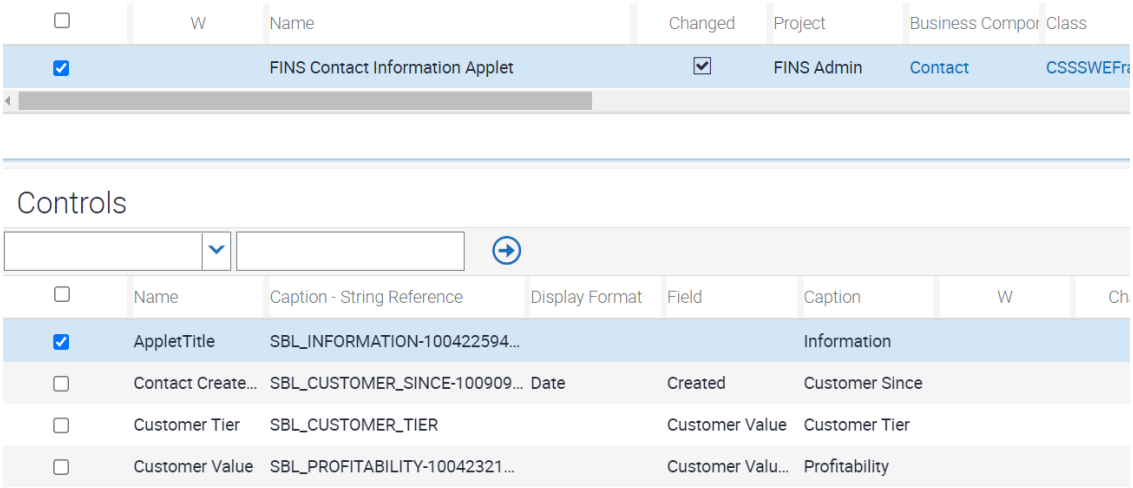
> **Note:** In this configuration example, you create a total of four controls: AppletTitle and three controls that each specify a business component field.

5. For this new applet, create and configure three controls (that will specify the business component fields) with the properties shown in the following table.

Property	Value	Comment
Name	<Control Name>	The name of the control.

Property	Value	Comment
Field	<BC Field Name>	The business component field, which is mapped at the applet level. This field value is exposed in the infolet. In this example, the three controls are configured to display data from these business component fields: o Customer Tier o Created o Customer Value
Field Type	BC Field	The field type, which must be BC Field to display the value of a business component field.
Caption	<Caption>	The field label for the field. In this example, the three controls are configured to use these captions: o Customer Tier o Customer Since o Profitability
HTML Type	PlainText	The HTML type for the controls you create. This must be PlainText.

The following image shows this sample configuration for the controls for the applet FINS Contact Information Applet:



To map the applet web template and web template items

1. For this applet (FINS Contact Information Applet), create a new applet web template.
2. Configure the properties for the new applet web template as shown in the following table – note that this is the generic applet web template configuration for this type of infolet.

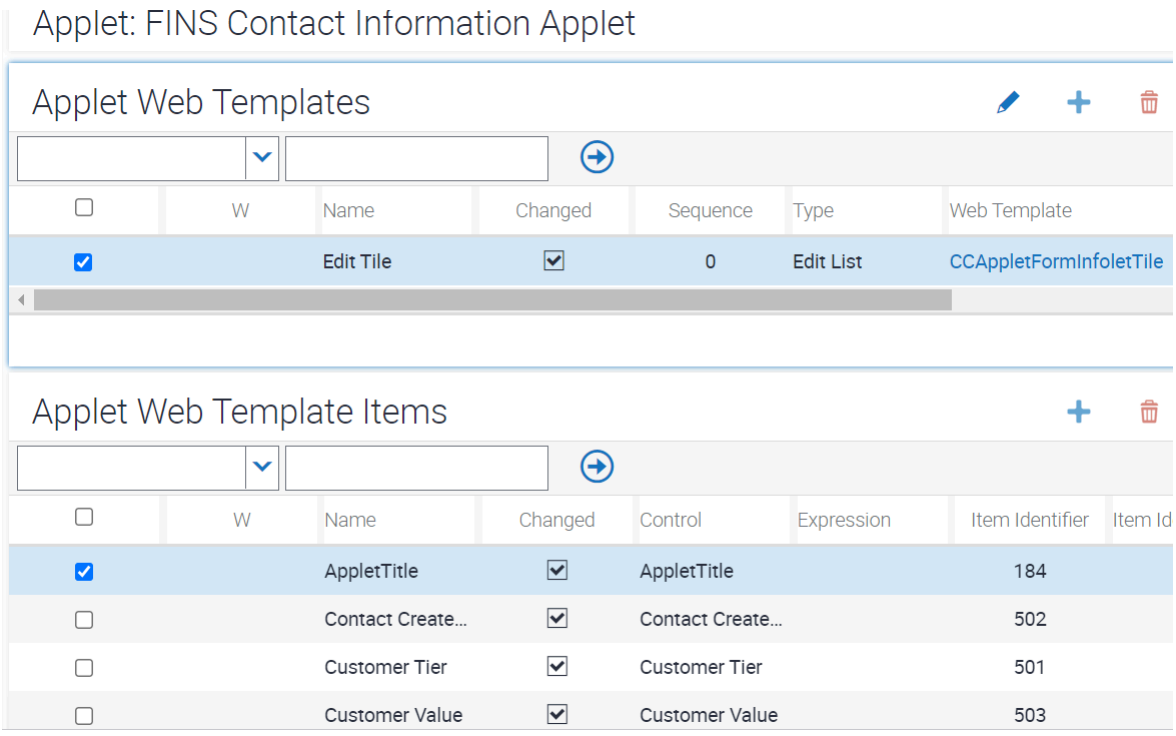
Property	Value	Comment
Name	<Name>	The name of the applet web template.
Type	Edit List	The type of applet web template. In this case, the Type must be Edit List.
Web Template	CCAppletFormInfoletTile	The type of web template for this applet web template. The Web Template must be CCAppletFormInfoletTile.

3. For the new applet web template, navigate to Applet Web Template Item, then create applet web template items with the values shown in the following table.

Name	Control	Item Identifier	Description
<Web Template Item 1>	<Title Control Name>	184	<Title Control Name> is the control created for showing the applet title. 184 is the item identifier specified in an infolet web template for mapping the applet title.
<Web Template Item 2>	<Control 2>	501	The web template item for the first control configured for field value display in the form infolet. 501 is the item identifier specified in an infolet web template for mapping the first value within the infolet.
<Web Template Item 3>	<Control 3>	502	The web template item for the second control configured for field value display in the form infolet.

Name	Control	Item Identifier	Description
			502 is the item identifier specified in an infolet web template for mapping the second value within the infolet.
<Web Template Item 4>	<Control 4>	503	The web template item for the third control configured for field value display in the form infolet. 503 is the item identifier specified in an infolet web template for mapping the third value within the infolet.

The following image shows the sample configuration of the applet web template and web template items, for the FINS Contact Information Applet infolet.



To map the presentation model

1. Navigate to Administration - Application screen, then the Manifest Administration view.

   You must register the new applet (such as FINS Contact Information Applet) with presentation model settings. This task is needed for automation support and for supporting a custom CSS style class for infolets. This

procedure is the same as the one in *Configuring an Infolet to Display a Value for a Field*, but in this case the Files record you create specifies the file siebel/forminfoletpm.js.

2. In the UI Objects list, create a new record with the values shown in the following table.

Type	Usage Type	Name
Applet	Presentation Model	<AppletName>

3. In the Object Expression list, create a new record with the value shown in the following table.

Level
1

4. In the Files list, create a new record with the values shown in the following table.

Level	Name
1	siebel/forminfoletpm.js

To map the physical renderer

1. Navigate to Administration - Application screen, then the Manifest Administration view.

   You must register the new applet (such as FINS Contact Information Applet) with physical renderer settings. This task is needed for automation support and for supporting a custom CSS style class for infolets. This procedure is the same as the one in *Configuring an Infolet to Display a Value for a Field*, but in this case the Files record you create specifies the file siebel/forminfoletpr.js.

2. In the UI Objects list, create a new record with the values shown in the following table.

Type	Usage Type	Name
Applet	Physical Renderer	<AppletName>

3. In the Object Expression list, create a new record with the value shown in the following table.

Level
1

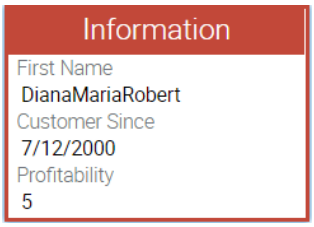
4. In the Files list, create a new record with the values shown in the following table.

Level	Name
1	siebel/forminfoletpr.js

About Vertical Styling for Form Applets

The framework for form infolets supports automatic vertical styling of the elements. If the data coming from the business component field is too large to fit into the infolet layout, then the framework automatically changes the layout to "Vertical" and the data is shown in a vertical (stacked) fashion as follows, and shown in the following image.

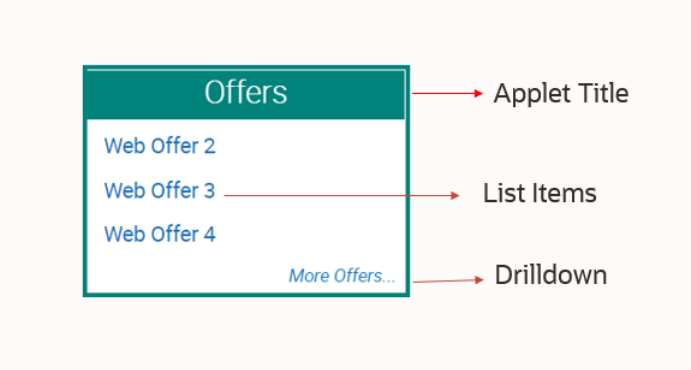
```
First Name
 DianaMariaRobert
Customer Since
 7/12/2020
Profitability
 5
```



Vertical styling is handled by the framework itself and requires no custom configuration.

Configuring a List Infolet to Display a List of Records

This topic describes how to configure a list infolet to display a list of records from a business component – for example, the following image show the Offers infolet with the following list items: Web Offer 2, Web Offer 3, Web Offer 4. This topic is part of *Configuring Data Visualization Components*.



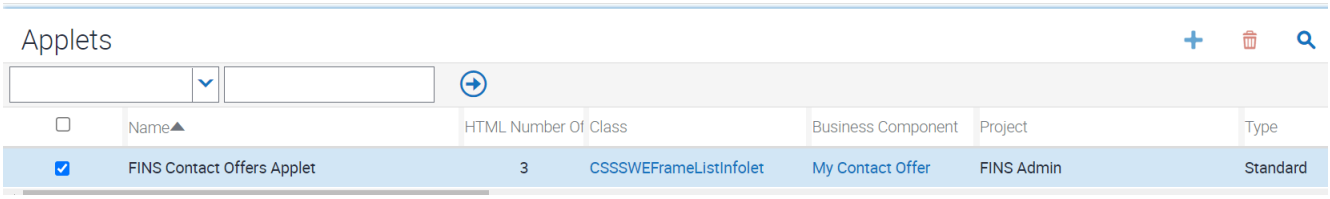
To configure a list infolet to display a list of records

1. Create a new workspace.

2. Create a new applet.
3. Configure the applet properties as shown in the following table – note that this is the generic applet configuration:

Property Name	Value	Comment
Name	<AppletName>	The name of the applet. In this example, the applet name is FINS Contact Offers Applet.
HTML Number of Rows	3	The number of rows to display in the infolet.
Class	CSSSWEFrameListInfolet	The name of a C++ class used to manage the applet. This is the infolet framework class for displaying a list of records. The framework constructs the CSS class based on the business component mapped to the applet.
Business Component	<BusComp>	The name of the business component whose data is to be displayed. The business component must be defined in the business object specified for any view that uses this applet.
Type	Standard	By default, the type is Standard.

The following image shows this sample configuration for the FINS Contact Offers Applet.



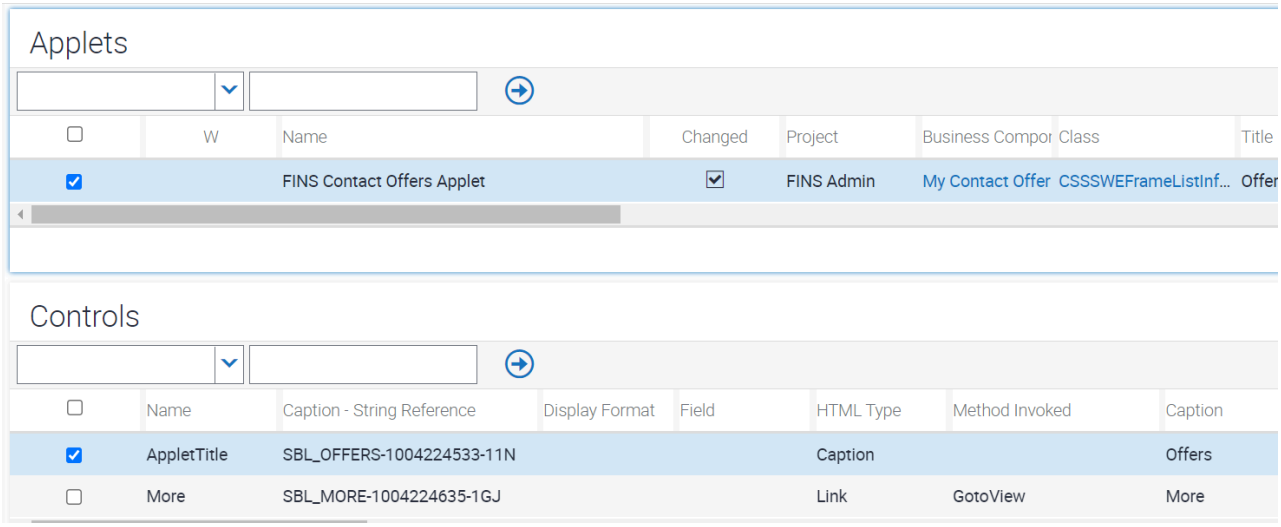
4. For this new applet, navigate to Controls. Create and configure a control for the applet title configuration. In this example, the caption is Offers. For details, see *Configuring an Infolet to Display a Value for a Field*.

> **Note:** In this configuration example, you create a total of two controls: AppletTitle and More.

5. For this applet, create and configure the More control, for More link configuration, as follows:

Property	Value	Comment
Name	More	In the sample applet, the control name is More. Do not change this value.
HTML Type	Link	Type of HTML display of field label and value. Do not change this value.
Method Invoked	GotoView	The method invoked when the More link is clicked.
Caption	More	Prefix used in drilldown text displayed. In this case, the value is set to More.

The following image shows this sample configuration for the controls for the applet FINS Contact Offers Applet.

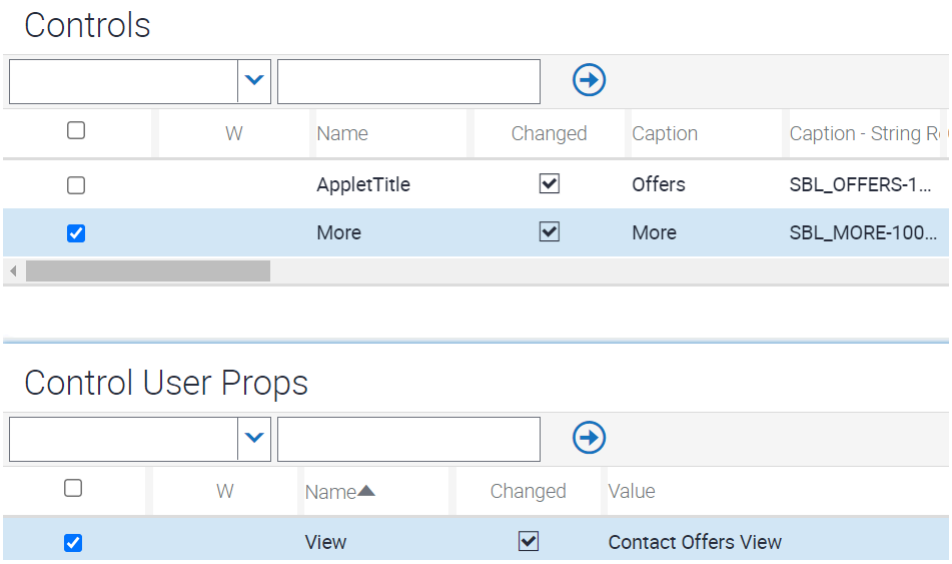


6. Configure the user properties for the More control as shown in the following table.

Property	Value	Comment
Name	View	Do not change this value.
Value	<View Name to be displayed on the link event>	The user is navigated to this destination view on clicking the link on the value in the infolet. In this example, the view name is Contact Offers View.

Property	Value	Comment

The following image shows this sample configuration for the More control and its user properties.



7. For the FINS Contact Offers Applet, navigate to List, then create a new list record named List with the properties shown in the the following table.

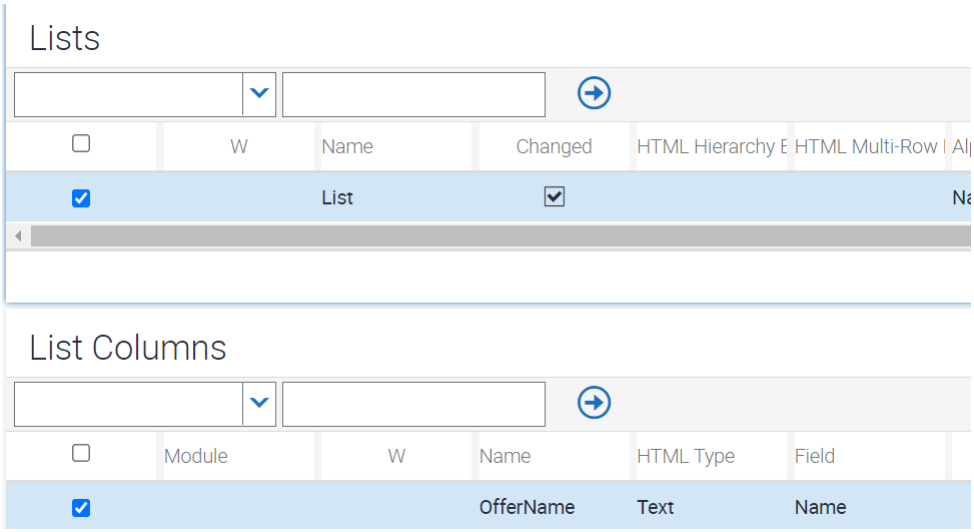
Property	Value	Comment
Name	List	Do not change this value.

8. For this list, create and configure a list column named OfferName with the properties shown in the following table.

Property	Value	Comment
HTML Type	Text	The HTML type for the list column you create. This value must be Text.
Name	<Name>	In the sample configuration, the list column name is OfferName.
HTML Row Sensitivity	True	
HTML List Edit	True	

Property	Value	Comment
HTML Display Mode	EncodeData	
Field	<BC Field>	The name of the field whose field value is shown in the applet, such as Name in case of FINS Contact Offers Applet.
Display Name	<Display Name>	The display name of the list column. In the sample configuration, the display name is Offer Name.

The following image shows the sample configuration for the List list and its OfferName list column.

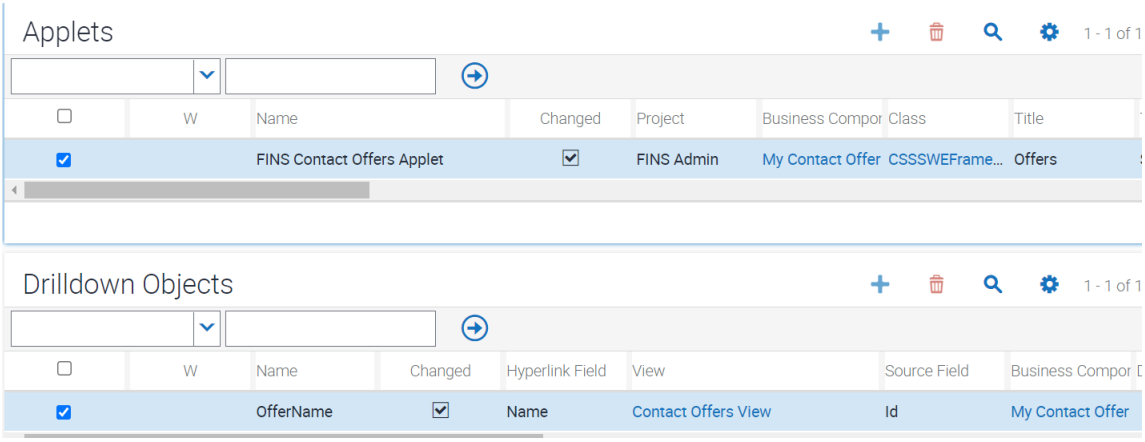


9. For this applet (FINS Contact Offers Applet), navigate to Drilldown Objects. then create and configure a drilldown object with the properties shown in the following table.

Property	Value	Comment
Name	<Name>	In the sample configuration, the drilldown object name is OfferName.
Hyperlink Field	<BC Field>	The name of the field that is mapped in the list column.
View	<Destination View Name>	The name of the view to navigate to.
Source Field	<Source Field>	The name of the source field.

Property	Value	Comment
Business Component	<BC Name>	The name of the business component, which is mapped at the applet level.

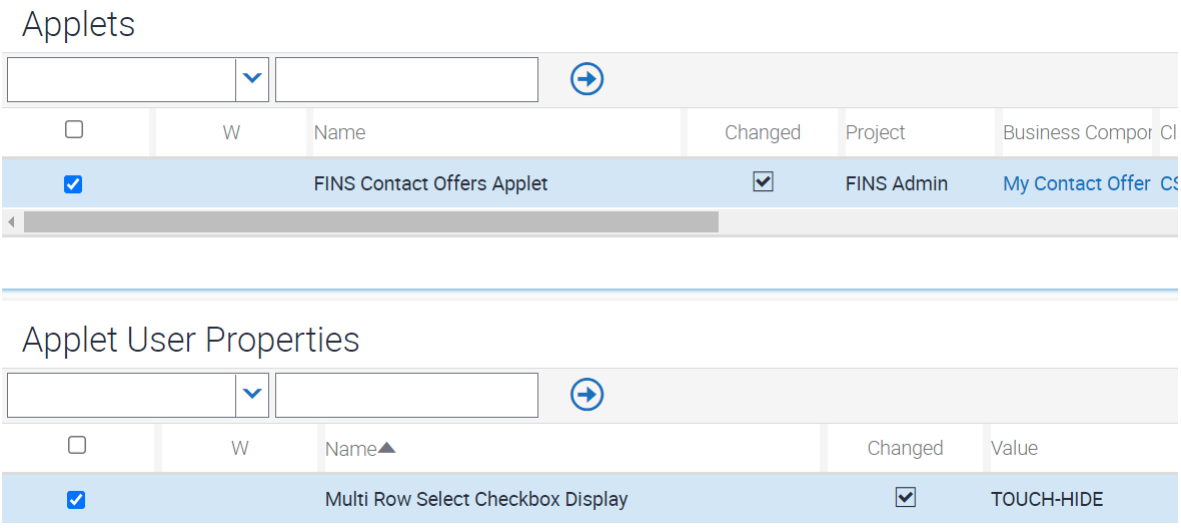
The following image shows the sample configuration for the OfferName drilldown object:



10. For this applet (FINS Contact Offers Applet), navigate to Applet User Properties, then create and configure a user property named Multi Row Select Checkbox Display with the value TOUCH-HIDE.

> **Note:** You must configure this property in order to hide the checkboxes that are shown by default in the list infolet in touch-screen enabled devices.

The following image shows the sample configuration for the applet user property Multi Row Select Checkbox Display.



To map the applet web template and web template items

1. For this applet (FINS Contact Offers Applet), create a new applet web template.
2. Configure the properties for the applet web template as shown in the following table – note that this is the generic applet web template configuration for this type of infolet:

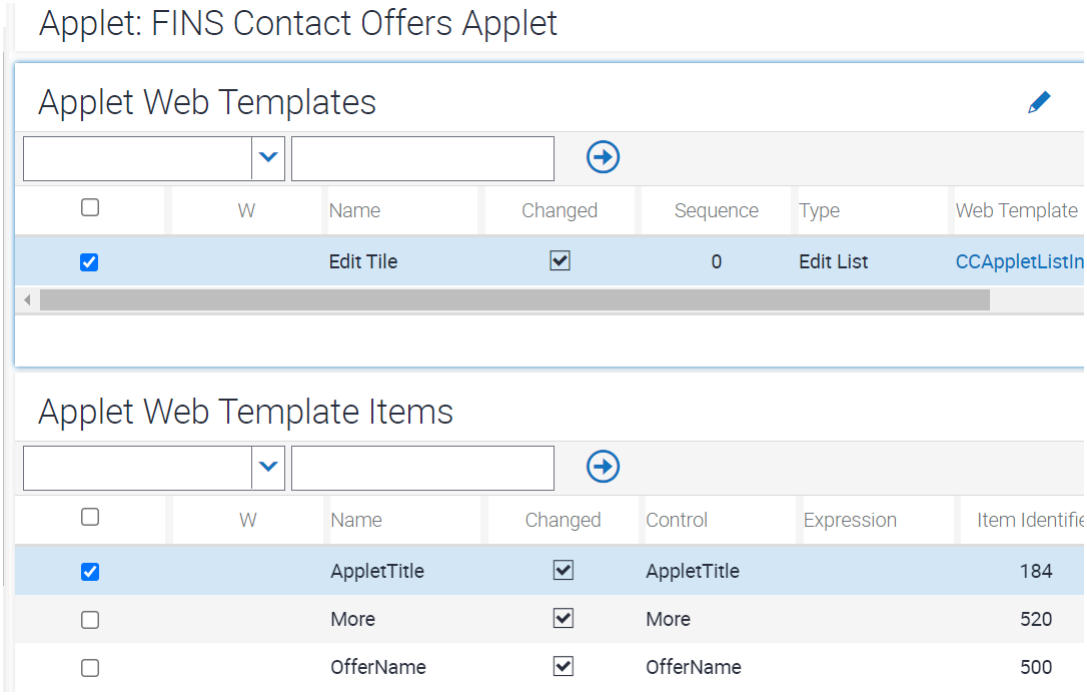
Property	Value	Comment
Name	<Name>	The name of the applet web template.
Type	Edit List	The type of applet web template. In this case, the Type must be Edit List.
Web Template	CCAppletListInfolet	The type of web template for this applet web template. The Web Template must be CCAppletListInfolet.

3. For the new applet web template, navigate to Applet Web Template Item, then create applet web template items with the values shown in the following table.

Name	Control	Item Identifier	Description
<Web Template Item 1>	<Title Control Name>	184	<Title Control Name> is the control created for showing the applet title. 184 is the item identifier specified in a list infolet web template for mapping the applet title.
<Web Template Item 2>	<Name>	500	You configured the list and list column in a prior procedure. 500 is the item identifier specified in a list infolet web template for mapping the list column <Name> that shows the data in the list infolet.
<Web Template Item 3>	More	520	You configured the list and list column in a prior procedure. 520 is the item identifier specified in a list infolet web

Name	Control	Item Identifier	Description
			template for mapping the More link within the infolet.

The following image shows a sample configuration of the applet web template and web template items, for the FINS Contact Offers Applet infolet:



To map the presentation model

1. Navigate to Administration - Application screen, then the Manifest Administration view.

   You must register the new applet (such as FINS Contact Offers Applet) with presentation model settings. This task is needed for automation support and for supporting a custom CSS style class for infolets. This procedure is the same as the one in *Configuring an Infolet to Display a Value for a Field*, but in this case the Files record you create specifies the file siebel/listinfoletpm.js.
2. In the UI Objects list, create a new record with values shown in the following table.

Type	Usage Type	Name
Applet	Presentation Model	<AppletName>

3. In the Object Expression list, create a new record with the value shown in following table.

Level
1

4. In the Files list, create a new record with the values shown in the following table.

Level	Name
1	siebel/listinfoletpm.js

To map the physical renderer

1. Navigate to Administration - Application screen, then to the Manifest Administration view.

   You must register the new applet (such as FINS Contact Offers Applet) with presentation model settings. This task is needed for automation support and for supporting a custom CSS style class for infolets. This procedure is the same as the one in *Configuring an Infolet to Display a Value for a Field*, but in this case the Files record you create specifies the file siebel/listinfoletpr.js.
2. In the UI Objects list, create a new record with values shown in the following table.

Type	Usage Type	Name
Applet	Physical Renderer	<AppletName>

3. In the Object Expression list, create a new record with value shown in the following table.

Level
1

4. In the Files list, create a new record with the values shown in the following table.

Level	Name
1	siebel/listinfoletpr.js

Additional Configuration Tasks for All Infolets

This topic summarizes some additional configuration tasks that apply for all infolets. This topic is part of *Configuring Data Visualization Components*.

You must consider the following details about configuring dashboards to display any type of infolets:

- **Mapping infolets to a view.** The template file for the view on which infolets are to be displayed must have placeholders for each infolet. As shown in the example for Siebel Financial Services dashboard, the HTML code must be updated as follows:
  - Apply the framework div class siebui-infolet-container in the view template file. For example, the wrapper class for all infolets, highlighted in the following image, is as follows:

    ```
    <div class="siebui-infolet-container siebbui-span-xl-12 siebui-span-lg-12
     siebui-span-md-12 siebui-span-sm-12">
    ```

  - Add placeholders for the required number of infolets. For example, 4 placeholders to map 4 infolets with id from 2 to 5, highlighted in the following image, is as follows:

    ```
    <div od-iterator="currentId" od-id="[2:5]">
     <div od-type="applet" od-id="od-attr-currentId" od-property="FormattedHtml"
     hintMapType="Applet" hintText="Applet" od-context="Parent"/>>
    <!--od section interator close->
    ```

Fins Dashboard View

```
1  <!--  Template Start: Fins Dashboard View -->
2  <div class="siebui-db-fins">
3    <div class="siebui-infolet-container siebui-span-xl-12 siebui-span-lg-12 siebui-span-md-12 siebui-span-sm-12">
4      <div class="siebui-contact-container">
5        <div od-type="applet" od-id="1" od-property="FormattedHtml" hintText="Contact Applet"/>
6      </div>
7      <div od-iterator="currentId" od-id="[2:5]">
8        <div od-type="applet" od-id="od-attr-currentId" od-property="FormattedHtml" hintMapType="Applet" hintText="Applet" od-context="Parent"/>
9      <!--od section iterator close-->
10     </div>
11   </div>
12   <div class="siebui-hierarchy-container siebui-span-xl-8 siebui-span-lg-8 siebui-span-md-8 siebui-span-sm-12">
13     <div od-type="applet" od-id="201" od-property="FormattedHtml" hintMapType="Applet" hintText="Applet" od-context="Parent"/>
14   </div>
15   <div class="siebui-timeline-container siebui-span-xl-4 siebui-span-lg-4 siebui-span-md-4 siebui-span-sm-12">
16     <div od-type="applet" od-id="204" od-property="FormattedHtml" hintMapType="Applet" hintText="Applet" od-context="Parent"/>
17   </div>
18 </div>
```

Wrapper class for all Infolets

4 placeholders to map 4 infolets with id from 2 to 5

- **Configuring view web template items for infolets.** View web template items used to display infolets must use the correct applet mode for the mapped applet (infolet) to be displayed. The correct mode is shown for the applet web template in each of the infolet configuration topics.

  - View template items for infolets displaying a single field value or an aggregate value are mapped using Base mode.
  - View template items for form infolets and list infolets are mapped using Edit List mode.

  The four infolets that are shown in the sample FINS Dashboard view are mapped with item identifiers 2, 3, 4, and 5, respectively.

View Web Template Item	Applet Name	Infolet Type	Applet Mode
Item Identifier 2	FINS Contact Information Applet	Form Infolet with 3 field values	Edit List
Item Identifier 3	FINS Contact Open Account Applet	Infolet showing Aggregation	Base
Item Identifier 4	FINS Contact Open Request Applet	Infolet showing Aggregation	Base
Item Identifier 5	FINS Contact Offers Applet	Infolet showing list of 3 records	Edit List

- **Configuring custom CSS style class for infolets.** You can override all styles for an infolet based on the CSS class that is applied by the framework based on the entity to which it is mapped.

  If the infolet is based on the Contact business component, then the server framework injects a class by name siebui-contact (siebui-<BC Name>). The server framework also takes care of any special characters in the business component name. The following are the some of the rules for converting the business component name to the CSS class name:

  - If the business component name has a space or a special character, then it is converted to "-" (dash).
  - If there are multiple special characters or spaces, one after the other, then all special characters and spaces are removed and only a single dash is substituted.
  - The entire name is converted to lower-case letters.

  For example, assume a business component named Comm Package Item.Sequence Number (Sequence). The CSS styling implementation for this business component can be done as follows:

```
&.siebui-comm-package-item-sequence-number-sequence- {
border: 5px solid @opportunity-color;
.siebui-applet-header {
background-color: #35ADC8;
}
}
```

- **Adding a new CSS class through configuration.** If two infolets are based on the same business component, then you can apply or override the styling on the applets by providing a new CSS class in an applet user

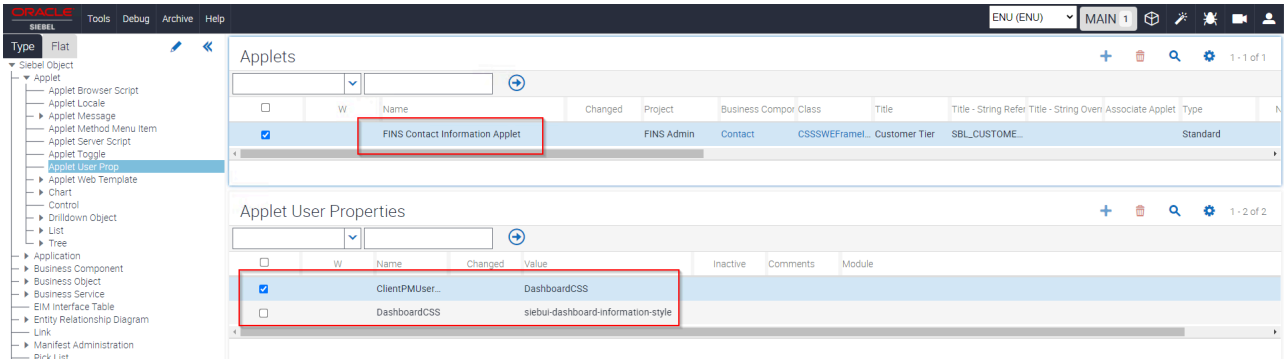
property. For example, the following applet user properties must be configured for the infolet applet. This configuration is common for any type of infolet.

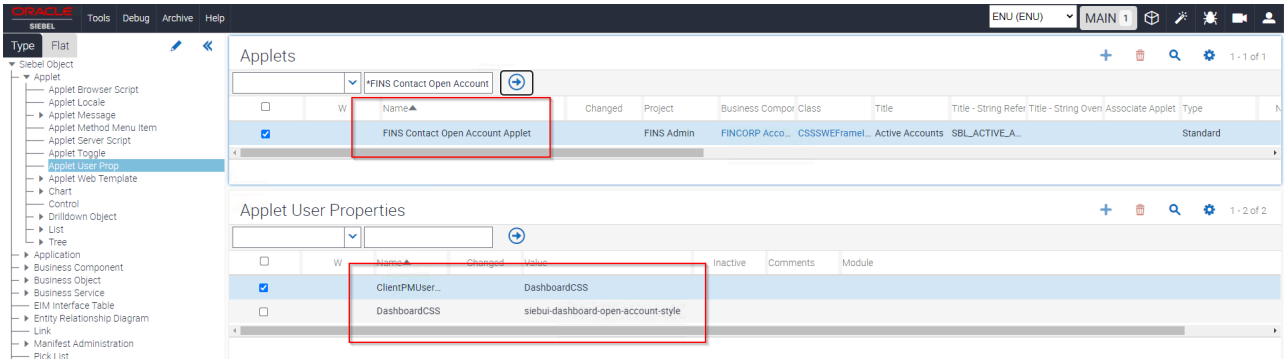
On a newly created applet, navigate to Applet User Prop. Create the following new records with the respective CSS class.

Name	Value
ClientPMUserProp	DashboardCSS
DashboardCSS	<Custom CSS class to be added>

Name	Value

The examples shown in the following images highlight the configuration required for adding a new CSS class (2).





The following image shows that the new CSS class (2) was added on top of the existing style class.



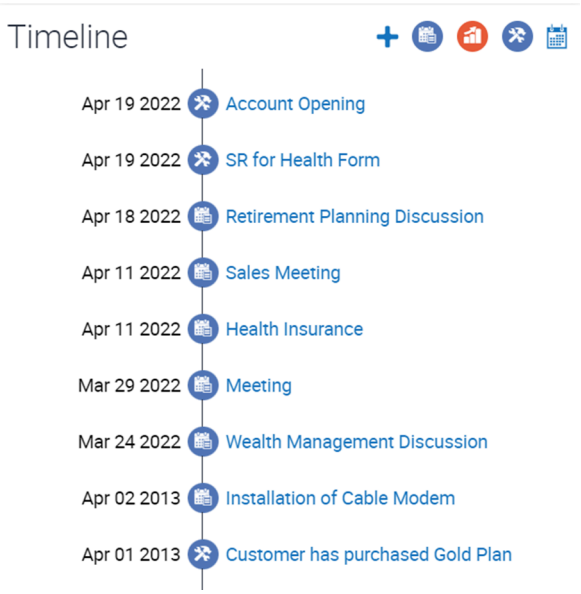
Configuring Timeline Components

This topic provides information about configuring timeline components. This topic is part of *Configuring Data Visualization Components*. This topic contains the following information:

- About Configuring Timeline Components
- About the Timeline VBC Virtual Business Component
- Mapping Business Components
- Configuring the Data Map
- Configuring the Timeline Applet
- Configuring the Applet User Property for Specifying the Data Map
- Creating Controls
- Configuring Drilldown Objects
- Creating the List and List Columns
- Mapping Applet Web Template and Web Template Items
- Mapping the Physical Renderer to the Timeline Applet
- Mapping the Timeline Applet to a View Web Template

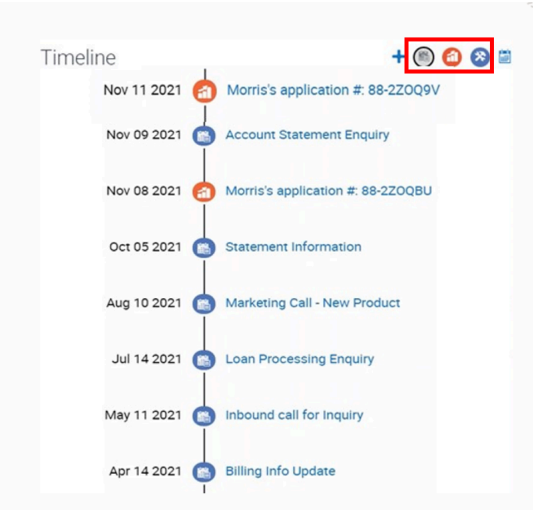
About Configuring Timeline Components

As noted in *Timeline Components*, a timeline applet can display various entities (like activities, opportunities, sales orders, and service requests) for a business object (such as Account or Contact). The entities are displayed in reverse chronological order, with the most recent record at the top. The example data used throughout is from the timeline applet FINS Contact Dashboard Timeline List Applet, which is part of the Siebel Financial Services dashboard. Each type of entity is displayed using a different icon. Icons are configurable. The following image shows an example of a timeline component.

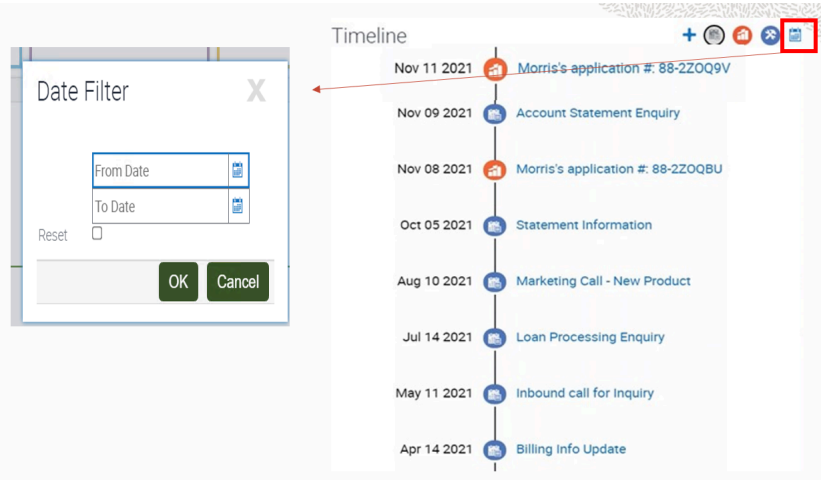


The following are some of the features of the timeline component:

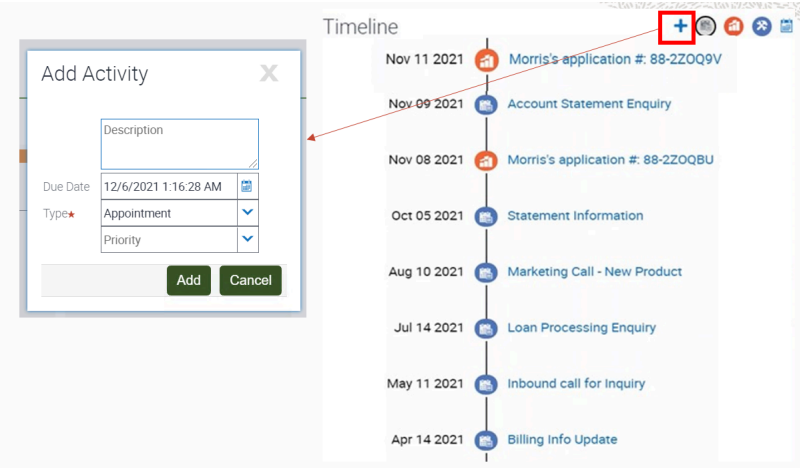
- **Entity filtering.** You can filter the data based on the type of entity. The entity filter buttons, highlighted in the following image, are toggle buttons. Clicking a button hides data for that entity from the timeline and the button dims. Clicking the dimmed button displays that entity data on the timeline again. In the following image, the available entity filters (activities, opportunities, service requests) are highlighted in the example timeline. (The timeline on the Siebel Telecommunications dashboard includes sales orders, as seen in *Siebel Telecommunications Dashboard*.)



- **Date filtering.** Click the Calendar button, highlighted in the following image, to filter the data based on a date range. Clicking the Calendar button opens the Date Filter pop-up dialog (also shown in the following image) where you can use the Reset option to reset the most recently used date filter.



- **Adding a new activity.** Click Add Activity (the plus (+) icon), highlighted in the following image, to enter a new activity. Clicking the plus (+) icon in the timeline opens the Add Activity pop-up. Once an activity is added, it appears in the timeline.



- **Navigating to detail view.** Click the description of a timeline item to drill down and navigate to the detail view. For example, clicking an opportunity item navigates to the Opportunity Details view, while clicking a sales order navigates to the Sales Order Details view.

About the Timeline VBC Virtual Business Component

Timeline VBC is a new C++ framework virtual business component created to support the timeline functionality. This VBC executes the data map to fetch information from multiple entity business components mapped to the VBC in the data map. Timeline VBC must be added to the business object with a link to the primary business component.

For sample data for the Siebel Financial Services dashboard, the VBC executes the data map FINS Contact Dashboard TimeLine DataMap and retrieves data from these entities: Activities, Opportunities, Sales Orders, and Service Requests. Data fields for these entities are mapped to the following fields in Timeline VBC:

Timeline VBC Fields (Destination Fields in the Data Map)	Description
Artifact Id	Required for drilldown functionality. Maps to the row ID field of the source business component.
Date	The pivot field used for sorting functionality in timeline. The data type must be DATETIME.
Description	The description shown for a timeline item. By default, the drilldown is configured on this field.
Type	The data map retrieves data from multiple business components. The Type field is used to distinguish between these business components. You must associate each unique type with the appropriate business component in the data map.
Type Image	The CSS class name associated to each type to render the icon image while showing a list of records in the timeline applet.

Mapping Business Components

Before you configure the timeline applet, you must map the business components and configure the data map. First, do the following:

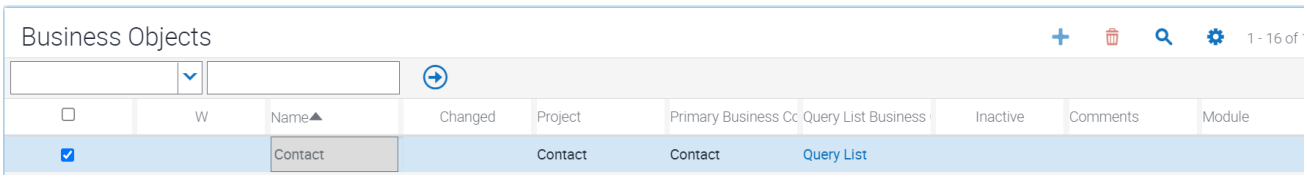
- Identify all source business components that need to be part of the timeline data. Make sure that business components for multiple entities are part of a single business object, such as Contact.
- Make sure the business component search specification is constrained to get the appropriate data. For example, for the Siebel Financial Services dashboard, the Opportunities business component is constrained by the Contact Id field to get opportunities for a particular contact.

In Siebel Web Tools, select Business Object. For the business object you are using (such as Contact), map the following required framework business object components (business components). Also map all of the business components for which data is to display on the timeline.

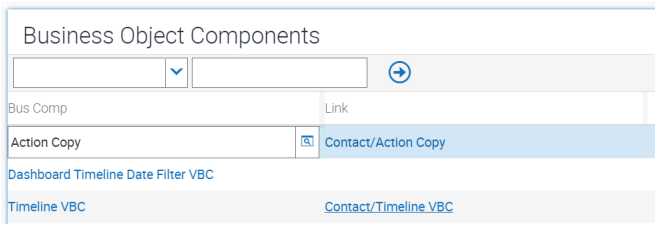
Business Component	Link	Comment
Dashboard Timeline Date Filter VBC		Supports date filtering in the timeline.
Timeline VBC	<Primary BC> / Timeline VBC	Consolidates data from multiple entities using the data map. In this example, the primary business component is Contact.
Action Copy	<Primary Business Component>/Action Copy	This BC is mapped to "Dashboard Timeline Activity PopUp Applet".

Note: As the Timeline pop-up Applet which is used to create a new Activity ,User has to make sure to add the Business Component mapped in the popup applet into the Business Object with required Link between parent and Action Copy BC as needed.

The following image shows the Contact business object.



The following image shows the business object components (Dashboard Timeline Date Filter VBC and Timeline VBC) for the Contact business object.



Next, configure the data map for the timeline.

Configuring the Data Map

Configuring the data map specifies source and destination business component mapping and merges source records from multiple entities to a single destination VBC (Timeline VBC). When creating the data map, you also define field mapping of source and destination fields. For examples, refer to the sample data map, FINS Contact Dashboard TimeLine DataMap.

Data map. To create the data map, navigate to Administration - Application, then Data Map Administration. Create a new record that specifies the names of the source business object (containing the source business components) and the destination business object (containing Timeline VBC). The following table shows fields for creating a data map record.

> **Note:** Data Maps can be Workspace enabled in your development environment. If they have been Workspace enabled in your development environment and you are working in that environment, then you can only modify them in an editable Workspace. You do not need an editable Workspace to create and edit Data Maps in your Production environment.

> **Note:** In FINS Contact Dashboard TimeLine DataMap, Contact is both the source and destination business object.

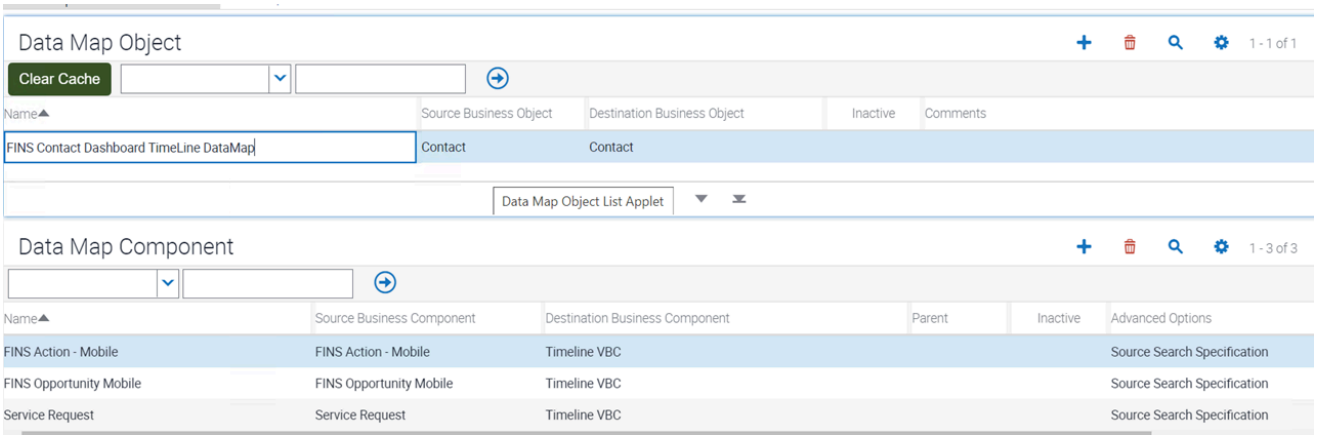
Name	Source Business Object	Destination Business Object
<Data Map Name>	<Source Business Object Name>	<Destination Business Object Name>

Data map components. Next, specify the data map components. Each data map component specifies a business component in the source business object and in the destination business object. The example timeline applet, FINS Contact SDashboard Timeline List Applet, displays timeline data from Activities, Opportunities, Sales Orders, and Service Requests.

To support activities, opportunities, and service requests, FINS Contact Dashboard TimeLine DataMap includes the following source business components:

- FINS Action - Mobile
- FINS Opportunity Mobile
- Service Request

The following image shows that these data map components are associated with the following data map object: Fins Contact Dashboard Timeline DataMap.



Data map components in FINS Contact Dashboard TimeLine DataMap also support advanced options, which are used for search specifications, as follows:

Name	Source Business Component	Destination Business Component	Advanced Options
FINS Action - Mobile	FINS Action - Mobile	Timeline VBC	Source Search Specification Value: [Contact Id]=[&Id]
FINS Opportunity Mobile	FINS Opportunity Mobile	Timeline VBC	Source Search Specification Value: [Key Contact Id]=[&Id]
Service Request	Service Request	Timeline VBC	Source Search Specification Value: [Contact Id]=[&Id]

Data map fields. Finally, in the Data Map Field list of the Data Map Administration view, you specify the field mapping from each source business component to the destination business component (Timeline VBC). Create one record for each destination field in Timeline VBC, as follows. Examples follow for data map components for Activities, Opportunities, and Service Requests. If you later determine you do not need a particular field to display in the timeline, then you can set it to be inactive.

Source Type	Source	Destination Type	Destination
Field	Id	Field	Artifact Id
Field	<Date Type Field from Source Business Component>	Field	Date

Source Type	Source	Destination Type	Destination
Expression	<Expression to Expose Value of Source Field Name in UI>	Field	Description
Expression	"<Source Business Component Name>"	Field	Type
Expression	"<Entity Icon Class>"	Field	Type Image

The following table lists examples of field mapping for the data map component FINS Action - Mobile (for Activities):

Source Type	Source	Destination Type	Destination
Field	Id	Field	Artifact Id
Field	Planned Completion	Field	Date
Expression	IIF([Description] IS NOT NULL, [Description],[Id])	Field	Description
Expression	"FINS Action - Mobile"	Field	Type
Expression	"siebui-icon-activities_icon"	Field	Type Image

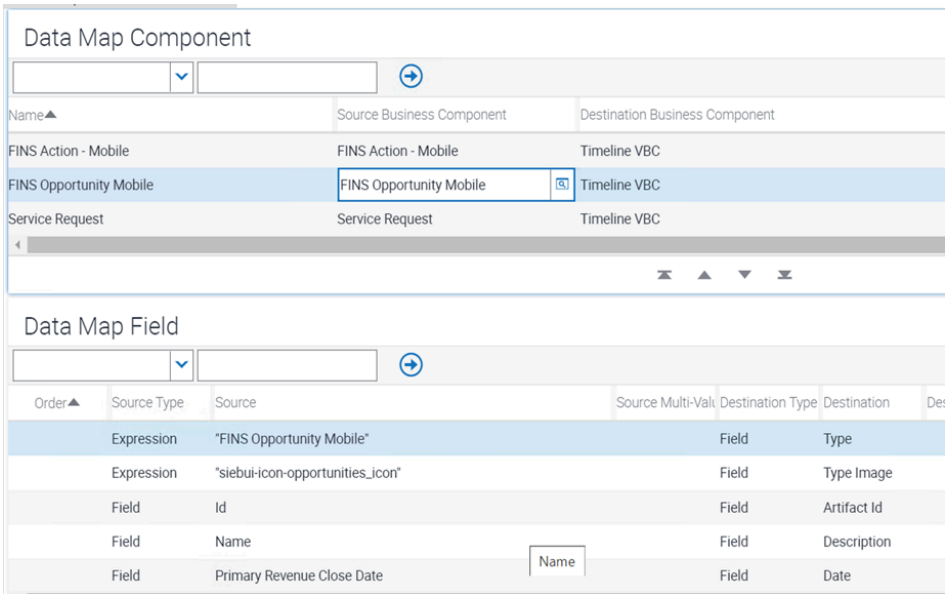
The following table lists examples of field mapping for the data map component FINS Opportunity Mobile (for Opportunities):

Source Type	Source	Destination Type	Destination
Field	Id	Field	Artifact Id
Field	Primary Revenue Close Date	Field	Date
Expression	Name	Field	Description
Expression	"FINS Opportunity Mobile"	Field	Type
Expression	"siebui-icon-opportunities_icon"	Field	Type Image

The following table lists examples of field mapping for the data map component Service Request (for Service Requests):

Source Type	Source	Destination Type	Destination
Field	Id	'Field'	Artifact Id
Field	Updated	'Field'	Date
Expression	IIF([Abstract] IS NOT NULL, [Abstract],[SR Number])	'Field'	Description
Expression	"Service Request"	'Field'	Type
Expression	"siebui-icon-service_requests_icon"	'Field'	Type Image

The following image shows a sample configuration for specifying data map fields (it shows the data map fields associated with the following data map component: FINS Opportunity Mobile.



Note: After completing the configuration of the data map for the timeline, clear the cache data using the Clear Cache button in the Data Map Objects list.

Configuring the Timeline Applet

This step shows you how to create and configure a new timeline applet. You must create a new workspace and create a new applet with the following properties. The example data in this topic is for the FINS Contact Dashboard Timeline List Applet.

Property Name	Value	Comment
Name	<AppletName>	The name of the applet. In this example, the applet name is FINS Contact Dashboard Timeline List Applet.
Class	CSSSWEFrameList	The name of a C++ class used to manage the applet. This is the timeline applet framework class. The framework constructs the CSS class based on the business component mapped to the applet.
Type	Standard	By default, the type is Standard.
Business Component	Timeline VBC	The name of the virtual business component for timeline applets. The Timeline VBC business component must be defined in the business object specified for any view that uses this applet.
Applet Title	<AppletTitle>	The applet title is autopopulated based on the value of the property Title - String Reference.

Configuring the Applet User Property for Specifying the Data Map

For the timeline applet (FINS Contact Dashboard Timeline List Applet in this example), create a Data Map user property, as follows. By setting this user property, you associate the data map you created with the timeline applet. This data map is used by the timeline framework to fetch the corresponding timeline items.

Property	Value	Comment
Data Map	<DataMapName>	The name of the data map, such as FINS Contact Dashboard TimeLine DataMap, that you are associating to this timeline applet.

Creating Controls

Filter type controls. For the timeline applet (FINS Contact Dashboard Timeline List Applet in this example), create a filter type control for each entity filter supported for the timeline, as follows:

Property	Value	Comment
Name	<EntityFilterName>	The name of the control, such as the following: Activities: ActionFilter

Property	Value	Comment
		Opportunities: OpptyFilter Service Requests: SRFilter
HTML Type	MiniButton	The HTML type for the control you create. This must be MiniButton.
Method Invoked	Filter|<BC Name of the Entity>	The method invoked when the entity filter is clicked.
Caption - String Reference	<Caption>	The caption property is populated based on this mapping.

Next, configure control user properties for each filter type control, as follows:

Property	Value	Comment
ClientPMUserProp	Filter Image	This value cannot be changed.
Filter Image	<Icon Name>	The icon for the entity filter. The following icon names are used: Activities: siebui-icon-activities_icon. Opportunities: siebui-icon-opportunities_icon. Service Requests: siebui-icon-service_requests_icon.

NewButton control. For the timeline applet (FINS Contact Dashboard Timeline List Applet in this example), create the NewButton control for the timeline, as follows. This button invokes the Activity Popup Applet.

Property	Value	Comment
Name	NewButton	The name of the control, such as NewButton.
HTML Type	MiniButtonNew	The HTML type for the control you create. This must be MiniButtonNew.
Method Invoked	ShowPopup	The method invoked when the new button is clicked.
Caption - String Reference	<Caption>	The caption property is populated based on this mapping. The value might be SBL_NEW-1004235437-602.

Next, configure control user properties for the NewButton control, as follows:

Property	Value	Comment
Mode	New	This value cannot be changed.
Popup	Dashboard Timeline Activity PopUp Applet	The name of the pop-up applet to display.

DateFilter control. For the timeline applet (FINS Contact Dashboard Timeline List Applet in this example), create the DateFilter control for the timeline, as follows. This button invokes the Dashboard Timeline Date Filter PopUp Applet.

Property	Value	Comment
Name	DateFilterButton	The name of the control, such as DateFilterButton.
HTML Type	MiniButtonNew	The HTML type for the control you create. This must be MiniButtonNew.
Method Invoked	ShowPopup	The method invoked when the new button is clicked.
Caption - String Reference	<Caption>	The caption property is populated based on this mapping. The value might be SBL_NEW-1004235437-602.

Next, configure control user properties for the DateFilter control, as follows:

Property	Value	Comment
Mode	New	This value cannot be changed.
Popup	Dashboard Timeline Date Filter PopUp Applet	The name of the pop-up applet to display.

Configuring Drilldown Objects

To support drilldown functions for the timeline, you must configure drilldown objects. For this applet (FINS Contact Dashboard Timeline List Applet in this example), navigate to Drilldown Objects. Then create and configure drilldown objects, as follows. Example values are shown for activities, opportunities, and service requests.

Name	View	Source Field	Business Component	Destination Field
<Drilldown Name>	<Destination View>	<Field from node level BC>	<Destination BC>	<Field from Destination BC>

Name	View	Source Field	Business Component	Destination Field
Example records follow.				
GotoActivities	Activity Attachment View	Artifact Id	Action	Activity Id
GoToOppty	Opportunity Details	Artifact Id	Opportunity	Row Id
GotoServiceRequests	Service Request Detail View	Artifact Id	Service Request	Id

When multiple drilldown objects are configured for the applet, then, for each drilldown object after the first one, you must create a record under Dynamic Drilldown Destinations, as follows:

Name	Field	Value	Destination Drilldown Object	Sequence	Upgrade Behavior
<Dynamic Drilldown Destination Name>	<Type> Specifies the Type field in the business component that the applet references.	<View matched with the Type field>	<Name of Drilldown Object in BC>	<Search order for children of a drilldown object> Search from lowest to highest according to the integer value that this property contains.	<Upgrade Behavior> Specifies the upgrade behavior.
GoToOppty	Type	FINS Opportunity Mobile	GoToOppty	1	Preserve
GotoServiceRequests	Type	Service Request	GotoServiceRequests	2	Preserve

Creating the List and List Columns

For this applet (FINS Contact Dashboard Timeline List Applet), navigate to List. Create a new list record named List, with the following properties:

Property	Value	Comment
Name	List	Do not change this value.

For this list, create and configure the following list columns:

Name	Field	HTML Type	Display Name
Artifact Id	Artifact Id	Text	Id

Name	Field	HTML Type	Display Name
Date	Date	PlainText	Date
Description	Description	Text	Description
Type	Type	Text	Type
Type Image	Type Image	Text	Type Image

Mapping Applet Web Template and Web Template Items

For this applet (FINS Contact Dashboard Timeline List Applet in this example), create a new applet web template, as follows:

Property	Value	Comment
Name	<Edit List>	The name of the applet web template.
Type	Edit List	The type of applet web template. In this case, the Type must be Edit List.
Web Template	Applet Vertical TimeLine	The type of web template for this applet web template. It must be Applet Vertical TimeLine.

Create an applet web template item for this applet web template, as follows:

Name	Control	Item Identifier	Type
<Entity Filter>	<Control Name>	<Item Identifier> Must be in range of 10 to 154, inclusive.	<Control Type>
<NewButton>	NewButton	155	Control
<DateFilterButton>	DateFilterButton	156	Control
Date	Date	501	List Item
Type	Type Image	502	List Item
Description	Description	503	List Item

Mapping the Physical Renderer to the Timeline Applet

You must register the new timeline applet (such as FINS Contact Dashboard Timeline List Applet in this example) with physical renderer settings. In this task, the Files record you create specifies the file siebel/timelinerenderer.js (for physical renderer). This task is needed for automation support and for supporting a custom CSS style class.

1. Navigate to Administration - Application screen and then to the Manifest Administration view.

   You must register the new applet (such as FINS Contact Dashboard Timeline List Applet) with new physical renderer settings.
2. In the UI Objects list, create a new record with the properties shown in the following table.

Type	Usage Type	Name
Applet	Physical Renderer	<AppletName>

3. In the Object Expression list, create a new record with the property shown in the following table.

Level
1

4. In the Files list, create a new record with the properties shown in the following table.

Level	Name
1	siebel/timelinerenderer.js

Mapping the Timeline Applet to a View Web Template

You must also map the timeline applet (FINS Contact Dashboard Timeline List Applet in this example) to the view that is to display the timeline applet. Typically, this is the view representing a particular dashboard into which you are mapping multiple applets for different data visualization components.

The view web template for the view FINS Contact Dashboard View (for the Siebel Financial Services dashboard), for example, has a placeholder for the timeline applet. For example, review the following content (which is highlighted in the following image) for the view web template Fins Dashboard View:

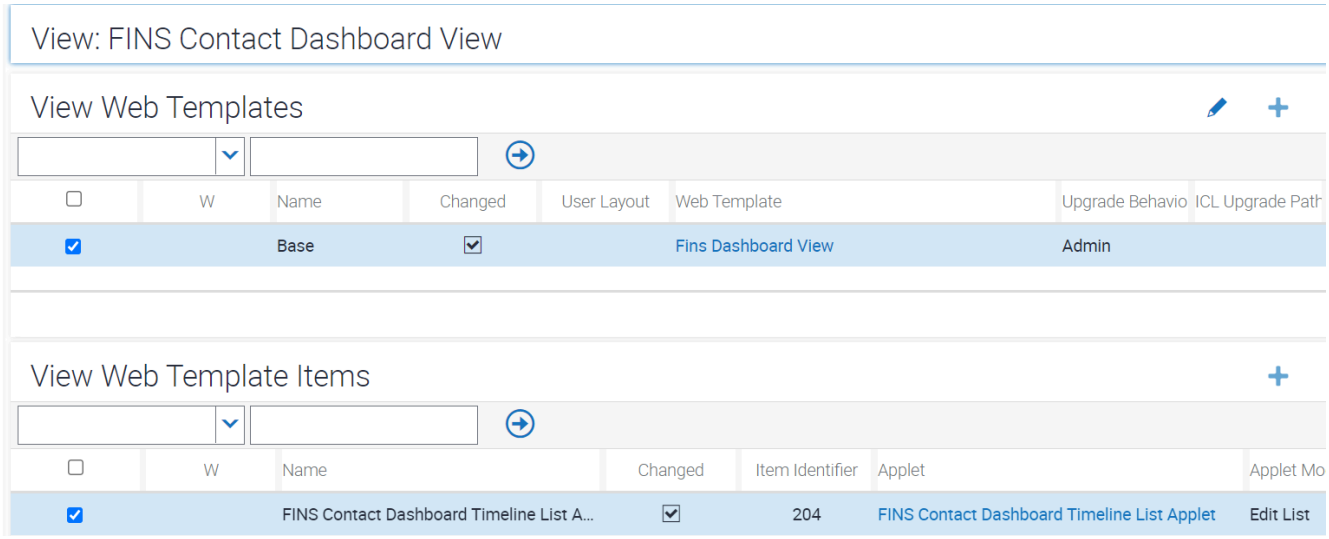
```
<div class="siebui-timeline-container siebui-span-x1-4 siebui-span-1g-4 siebui-span-md-4 siebui-span-sm-12">
 <div od-type="applet" od-id="204" od-property="FormattedHtml" hintMapType="Applet" hintText="Applet" od-
context="Parent"/>
```



In any new view to display a timeline applet, this timeline applet must be mapped to the view. You must also add the necessary classes to the view web template to achieve the required height and width. You can customize or override the styling on the existing CSS classes of the timeline applet. The view web template item must specify the applet mode Edit List for a timeline applet that you map. For example, create a view web template item as follows:

Name	Applet	Applet Mode	Item Identifier
<Applet Name>	<Timeline Applet Name>	Edit List	<Item Identifier for Timeline>

The following image shows a sample configuration of the view web template and view web template items, for this example (FINS Contact Dashboard View).



Configuring Hierarchy Components

This topic provides information about configuring hierarchy components, which are also called relationship hierarchy components. This topic is part of *Configuring Data Visualization Components*.

This topic contains the following information:

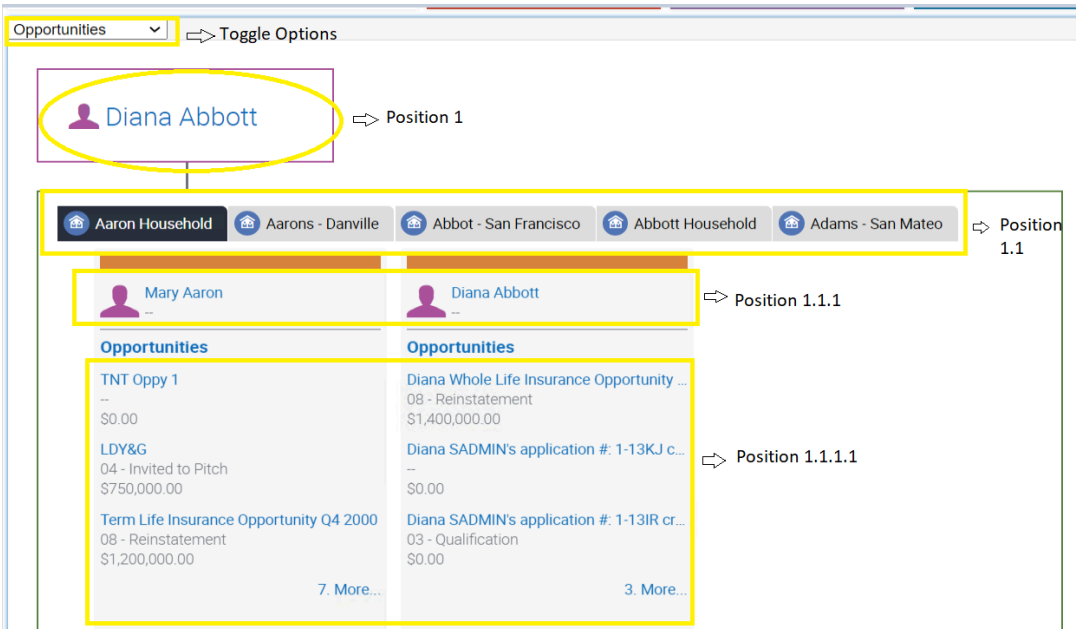
- About Configuring Hierarchy Components
- Mapping Business Components to the Business Object
- Configuring and Mapping Business Component Fields
- Configuring the Hierarchy Applet
- Configuring the Tree Control
- Mapping the Applet Web Template and Web Template Items
- Configuring the Applet Tree and Tree Nodes
- Configuring Drilldown Objects
- Configuring Applet User Properties Related to Drilldown and Images
- Configuring Multiple Hierarchy Applets
- Mapping the Presentation Model and Physical Renderer
- Mapping the Hierarchy Applet to a View Web Template

About Configuring Hierarchy Components

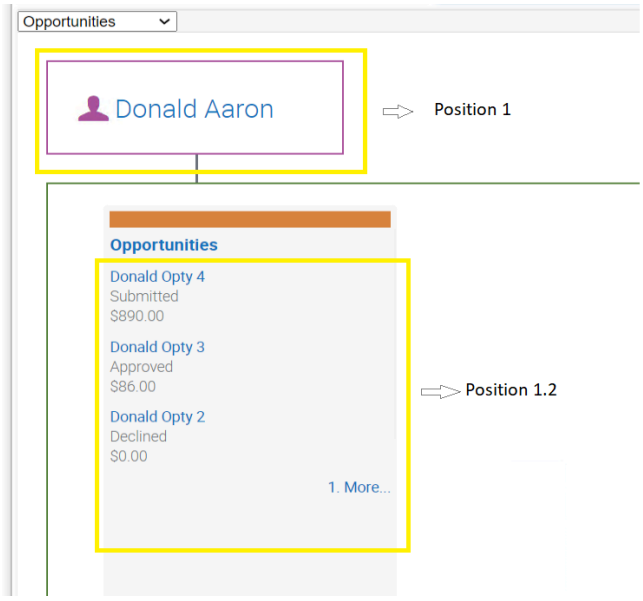
As noted in *Hierarchy Components*, a hierarchy applet displays linked data in an ordered four-level hierarchy. Hierarchy data is organized within a tree model framework and rendered using a custom tile model. Conceptually, each level represents a position, such as the following. The example data used throughout is from the hierarchy applet FINS Contact Hierarchy Applet, which is part of the Siebel Financial Services dashboard.

- **Position 1 (level 1).** In this example, this level represents a single contact, such as Diana Abbott. The remaining levels represent aggregated data associated with this contact.
- **Position 1.1 (level 2).** This level represents a single household of the parent contact. Multiple households (up to five) are displayed using tabs, with a single tab currently selected.
- **Position 1.1.1 (level 3).** This level represents multiple contacts of the parent household. Each contact is shown in a tile below the parent household.
- **Position 1.1.1.1 (level 4).** This level represents the opportunities of each parent household contact. The opportunities are shown in a tile below the parent household contact. The user can also access different level-4 data using an applet toggle drop-down menu.

The hierarchy shown in the following image is based on the level-4 entity, opportunities. An applet toggle is available at the top of the hierarchy applet, from which, in this example, the user might be able to select other entities, such as activities, financial accounts, or service requests. You can configure each selectable option in the applet toggle and what it displays.



If no data is found representing position 1.1 (that is, households of parent contact in this example), then the hierarchy applet will display alternate data, referred to as position 1.2. For example, position 1.2 can be configured as opportunities of the parent contact. In this example, if data from position 1.2 is displayed (as shown in the following image), then no tabs are shown. The position 1.2 data (opportunities) displays in tiles, the same as in the four-level case previously described, because Opportunities is selected in the applet toggle drop-down menu.



Mapping Business Components to the Business Object

The relationship hierarchy framework can be reused by any business scenario matching up to four levels of hierarchy. To adapt this framework for a new business scenario, first identify the business object and the four levels of business components for which to show hierarchical data. The business components representing the four levels must belong to a single business object and must have a hierarchical relationship within the business object.

If you need to create the hierarchical relationship of the four levels of business components, then you can use the following table as a general guide for this purpose. Also examine the repository data used in this example, which is configured in the business object Contact.

Business Component	Links (Source BC / Destination BC)	Description
<Position 1 BC> Example: FINS DB Hierarchy Root Contact	<Primary BC> / <Position 1 BC> Example: Contact/FINS DB Hierarchy Root Contact	Root contact "Diana Abbott"
< Position 1.1 BC> Example: FINS DB Hierarchy Household	<Position 1 BC> / <Position 1.1 BC> Example: FINS DB Hierarchy Root Contact/FINS DB Hierarchy Household	Relation between root contact and all its households.
< Position 1.1.1 BC> Example: FINS DB Hierarchy Contact	<Position 1.1 BC> / <Position 1.1.1 BC> Example: FINS DB Hierarchy Household/FINS DB Hierarchy Contact	Relation between a household and all its contacts.
<Entity 1 Position 1.1.1.1 BC> Example: FINS DB Hierarchy Opportunity 2	<Position 1.1.1 BC> / <Position 1.1.1.1 BC> Example: FINS DB Hierarchy Contact/FINS DB Hierarchy Opportunity 2	Relation between a household contact and its opportunities entity.
<Entity 1 Position 1.2 BC> Example: FINS DB Hierarchy Opportunity 1	<Position 1 BC> / <Position 1.2 BC> Example: FINS DB Hierarchy Root Contact/FINS DB Hierarchy Opportunity 1	Relation between root contact and its opportunities entity.

To support hierarchy applets in dashboards, the business component relationships described in the table have been implemented for the Contact business object. These repository updates included creating new business components and links.

According to your requirements for using hierarchy applets, you might choose to modify or extend these configuration changes or configure new hierarchies. You might, for example, want to reuse the existing hierarchical relationships defined for positions 1, 1.1, and 1.1.1 but, for positions 1.1.1.1 and 1.2, make configuration changes that substitute opportunities with activities, financial accounts, or service requests. For these entities, the following business components are configured and mapped in the Contact business object to show entity details at positions 1.1.1.1 and 1.2.

In these tables, Entity 1 is Opportunity, Entity 2 is Activity, Entity 3 is Financial Account, and Entity 4 is Service Request.

Business Component	Links (Source BC / Destination BC)	Position
<Entity 2 Position 1.1.1.1 BC> Example: FINS DB Hierarchy Activity 1	FINS DB Hierarchy Contact/FINS DB Hierarchy Activity 1	1.1.1.1
<Entity 2 Position 1.2 BC> Example: FINS DB Hierarchy Activity 2	FINS DB Hierarchy Root Contact/FINS DB Hierarchy Activity 2	1.2
<Entity 3 Position 1.1.1.1 BC> Example: FINS DB Hierarchy FA 1	FINS DB Hierarchy Contact/FINS DB Hierarchy FA 1	1.1.1.1
<Entity 3 Position 1.2 BC> Example: FINS DB Hierarchy FA 2	FINS DB Hierarchy Root Contact/FINS DB Hierarchy FA 2	1.2
<Entity 4 Position 1.1.1.1 BC> Example: FINS DB Hierarchy SR 1	FINS DB Hierarchy Contact/FINS DB Hierarchy SR 1	1.1.1.1
<Entity 4 Position 1.2 BC> Example: FINS DB Hierarchy SR 2	FINS DB Hierarchy Root Contact/FINS DB Hierarchy SR 2	1.2

Configuring and Mapping Business Component Fields

As part of configuring hierarchy applets, you must also configure and map business component fields. These business component fields are mapped in the tree node configuration, which is part of the applet configuration. For some of the fields in the current example, calculated fields were created (as detailed in the following paragraph) allowing values from multiple fields to be combined. Where more than one field value is required, you must create a calculated field (concatenation of multiple fields separated by | symbols).

In the example in the following table, each position can be configured to display field data in a way consistent with its location in the hierarchy:

- Positions 1 and 1.1 each support the display of a single field value (single attribute).
- Position 1.1.1 uses a calculated field value to obtain and display two field values (two attributes).
- Positions 1.1.1.1 and 1.2 each use a calculated field to obtain and display four field values (four attributes).

Position	Business Component	Field	Number of Attributes Shown	Description
Position 1	FINS DB Hierarchy Root Contact	Full Name	1	The field Full Name is mapped in the applet tree control Node Position 1 to render the Contact FullName value.
Position 1.1	FINS DB Hierarchy Household	Household Name	1	The field Household Name is mapped in the applet tree control Node Position 1.1 to render the Household value.
Position 1.1.1	FINS DB Hierarchy Contact	'Rel Hierarchy Calc Field'	2	This calculated field has the following value: [Full Name]+"| "+[Relation to Household]
Position 1.1.1.1	FINS DB Hierarchy Opportunity 2	'Rel Hierarchy Calc Field'	4	This calculated field has the following value: [Name]+"| "+[Sales Stage]+"| "+"currency"+"| "+[Currency Code]+"| "+[Primary Revenue Amount]
Position 2	FINS DB Hierarchy Opportunity 1	'Rel Hierarchy Calc Field'	4	This calculated field has the following value: [Name]+"| "+[Sales Stage]+"| "+"currency"+"| "+[Currency Code]+"| "+[Primary Revenue Amount]

You can reuse existing fields or create new calculated fields, depending on your use case.

Configuring the Hierarchy Applet

This step shows you how to create and configure a new hierarchy applet. You must create a new workspace and create a new applet with the following properties. The example data in this topic is for the FINS Contact Hierarchy Applet.

Property Name	Value	Comment
Name	<AppletName>	The name of the applet. In this example, the applet name is FINS Contact Hierarchy Applet.
Class	CSSSWETreeRelHier	The name of a C++ class used to manage the applet. This is the hierarchy applet framework class. The framework constructs the CSS class based on the business component mapped to the applet.

Property Name	Value	Comment
Type	Standard	By default, the type is Standard.
Business Component	<BusComp>	The name of the primary business component for this hierarchy applet. The business component must be defined in the business object specified for any view that uses this applet.
Applet Title	<AppletTitle>	The applet title is autopopulated based on the value of the property Title - String reference.

Configuring the Tree Control

For the hierarchy applet (FINS Contact Hierarchy Applet in this example), create a Tree control, as follows.

Property	Value	Comment
Name	Tree	The name of the control.
HTML Type	Text	The HTML type for the control you create. This must be Text.
Caption - String Reference	Tree	The caption property is populated based on this mapping. In this example, the caption is Tree.

Mapping the Applet Web Template and Web Template Items

For this applet (FINS Contact Hierarchy Applet in this example), create a new applet web template with the properties shown in the following table.

Property	Value	Comment
Name	Base	The name of the applet web template.
Type	Base	The type of applet web template. In this case, the Type must be Base.
Web Template	Relationship Hierarchy Applet	The type of web template for this applet web template. It must be Relationship Hierarchy Applet.

Create an applet web template item for this applet web template with the properties shown in the following table.

Name	Control	Item Identifier
Tree	Tree	99,994

Configuring the Applet Tree and Tree Nodes

For this applet (FINS Contact Hierarchy Applet in this example), navigate to the Tree object and create a new record with the properties shown in the following table.

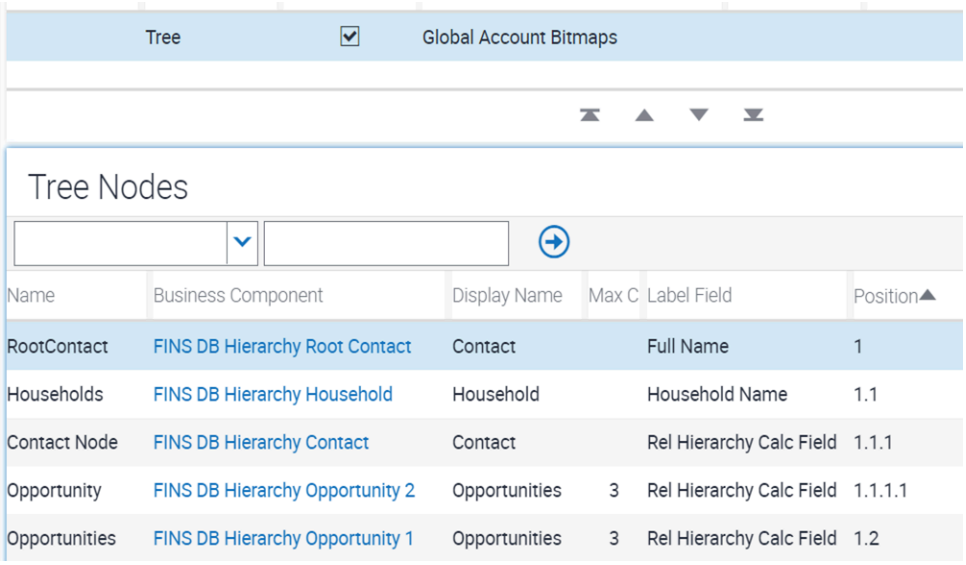
Name	HTML Hierarchy Bitmap
Tree	Global Account Bitmaps

Then navigate to the Tree Node object and create a new record with the properties shown in the following table. This configuration differs in each entity at only positions 1.1.1.1 and 1.2.

Name	Business Component	Display Name	Label Field	Max Child Items	Position
<Position 1>	<Position 1 BC>	<Title of Position 1>	<Position 1 BC Field>		1
<Position 1.1>	<Position 1.1 BC>	<Title of Position 1.1>	<Position 1.1 BC Field>		1.1
<Position 1.1.1>	<Position 1.1.1 BC>	<Title of Position 1.1.1>	<Position 1.1.1 BC Field>		1.1.1
<Position 1.1.1.1>	<Position 1 1.1.1.1 BC>	<Title of Position 1.1.1.1>	<Position 1.1.1.1 BC Field>	3	1.1.1.1
<Position 1.2>	<Position 1.2 BC>	<Title of Position 1.2>	<Position 1.2 BC Field>	3	1.2

Note: The Max Child Items property is set to 3 for the tree nodes for positions 1.1.1.1 and 1.2, meaning that up to three records can be displayed for those positions. If there are more than three records, then the hierarchy applet framework supports configuring a More link that allows the user to display additional records, as described later in this topic.

The following image shows an example of tree node configuration for a hierarchy applet.



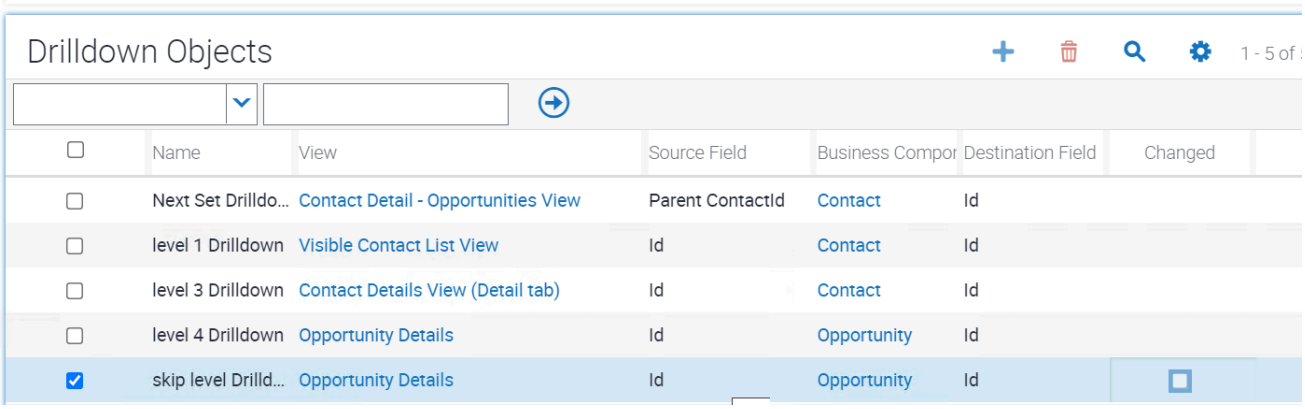
Configuring Drilldown Objects

To support drilldown functions for different levels in the hierarchy, you must configure drilldown objects. For this applet (FINS Contact Hierarchy Applet in this example), navigate to Drilldown Objects. Then create and configure five drilldown objects for different positions with the properties shown in the following table. Drilldown is supported at positions 1, 1.1.1, 1.1.1.1, and 1.2 and on More link. This configuration differs in each entity at only positions 1.1.1.1 and 1.2.

Note: The Source Field in the following table is mapped to an unbounded picklist. This picklist lists all fields of the business component that are mapped at the Applet level. However, for the hierarchy to map the fields of the relevant position business component, instead of using the drop-down directly, update the property with the respective field.

Name	View	Source Field	Business Component	Destination Field
<Drilldown Name> For example: 1. Next Set Drilldown 2. Level 1 Drilldown 3. Level 3 Drilldown 4. Level 4 Drilldown 5. Skip Level Drilldown	<Destination View> For example: 1. Contact Detail - Opps View 2. Visible Contact List View 3. Contact Details View (Detail tab) 4. Opportunity Details 5. Opportunity Details	<Field from node level BC> For example: 1. Parent ContactId 2. Id 3. Id 4. Id 5. Id	<Destination BC> For example: 1. Contact 2. Contact 3. Contact 4. Opportunity 5. Opportunity	<Field from Destination BC> For example: 1. Id 2. Id 3. Id 4. Id 5. Id

The following shows an example of the drilldown objects (as detailed in the previous table) for a hierarchy applet.



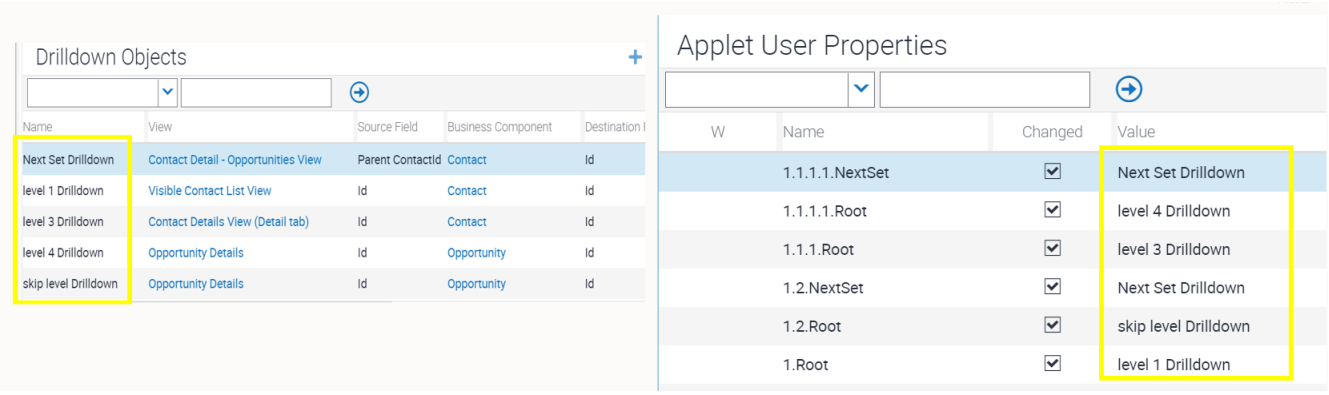
Configuring Applet User Properties Related to Drilldown and Images

In addition to creating the drilldown objects for the hierarchy applet, you must enable the drilldown for each applicable tree node position. To do this, first create an applet user property called Enable Tree DrillDown. Then create applet user properties for the tree node position and "Root" or "NextSet", as shown in the following table. Set the value to the name of the corresponding drilldown object, which you previously configured.

Then, for the new applet, navigate to Applet User Properties. Create new records with the following properties. For each property, set the value to the name of the corresponding drilldown object.

Name	Value
1.Root	<Position 1 drilldown name>
1.1.1.Root	<Position 1.1.1 drilldown name>
1.1.1.1.Root	<Position 1.1.1.1 drilldown name>
1.1.1.1.NextSet	<Position 1.1.1.1 More link drilldown name>
1.2.Root	<Position 1.2 drilldown name>
1.2.NextSet	<Position 1.2 More link drilldown name>

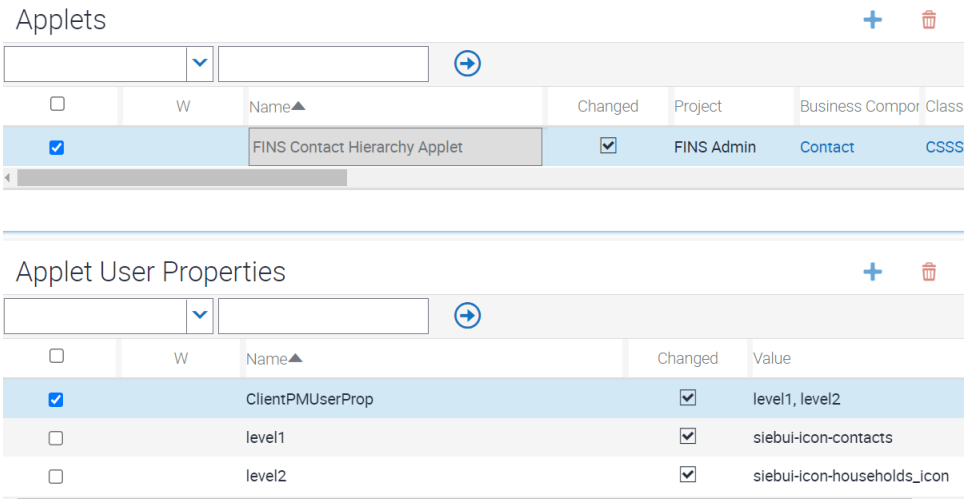
The following image shows an example of the drilldown objects (Next Set Drilldown, Level 1 Drilldown, Level 3 Drilldown, Level 4 Drilldown, Skip Level Drilldown) and applet user properties (as detailed in the previous table) to support drilldown for a hierarchy applet.



Next, configure applet user properties (as shown in the following table) to map images at positions 1, 1.1, and 1.1.1.

Name	Value	Comments
ClientPMUserProp	level1, level2	This is a generic framework user property to send data to the client. This property is processed at the client side to get the level1 icon value to map in the user interface at positions 1 and 1.1.1 and to get the level2 icon value to map at position 1.1. The property name and value cannot be changed for hierarchy applets.
level1	<Icon Class>	This property is used by the hierarchy applet client code to fetch the icon value for positions 1 and 1.1.1. The property name cannot be changed. In this example, the property specifies the value siebui-icon-contacts.
level2	<Icon Class>	This property is used by the hierarchy applet client code to fetch the icon value for position 1.1. The property name cannot be changed. In this example, the property specifies the value siebui-icon-households_icon.

The following image shows an example of the applet user properties (ClientPMUserProp, level1, level2) to support images for a hierarchy applet (FINS Contact Hierarchy Applet).



Configuring Multiple Hierarchy Applets

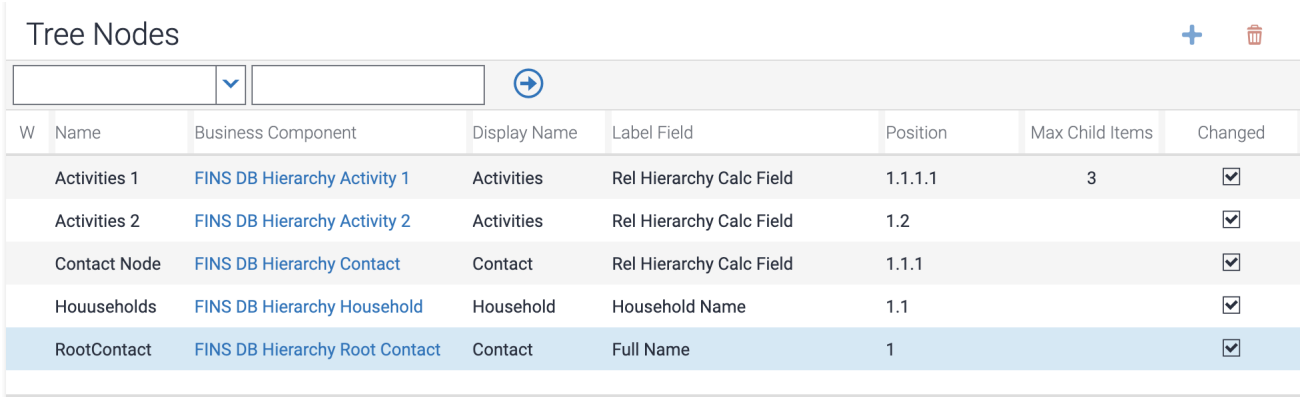
If you need to create a new hierarchy applet for a new entity to serve as a toggle applet for displaying level-4 data, then perform the following four steps:

- **Copy an existing hierarchy applet.** Make a copy of an existing configured hierarchy applet and then modify the copy for the new entity. By doing this, you save configuration steps compared to creating a new hierarchy applet from the beginning.
- **Modify the tree node configuration for the copied hierarchy applet.** For the newly copied hierarchy applet, change the tree node property values for position 1.1.1.1 and position 1.2, as detailed in the following table, to support the new entity.

Name	Business Component	Display Name	Label Field	Position
<Entity Name for Position 1.1.1.1>	<New Entity BC for Position 1.1.1.1>	<Entity Display Name for Position 1.1.1.1>	<Field from New Entity BC>	1.1.1.1
<Entity Name for Position 1.2>	<New Entity BC for Position 1.2>	<Entity Display Name for Position 1.2>	<Field from New Entity BC>	1.2

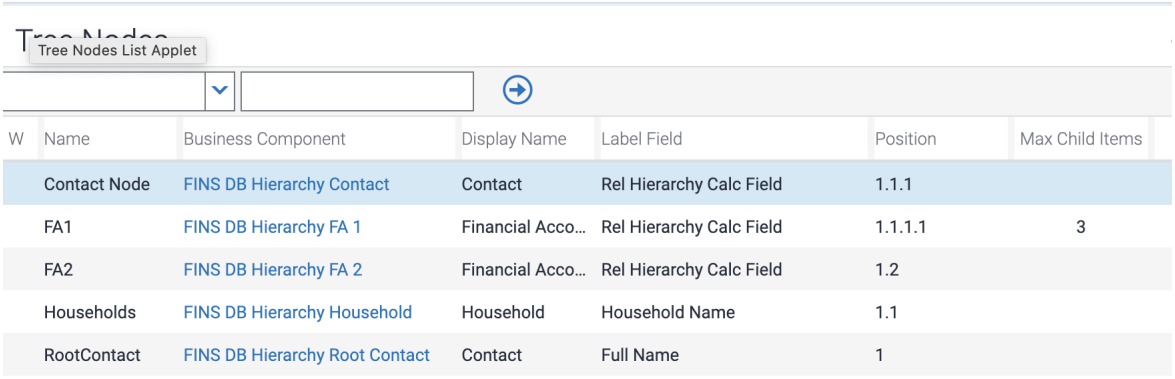
Name	Business Component	Display Name	Label Field	Position

The following image shows an example of the tree nodes to modify when creating an Activities hierarchy applet.



W	Name	Business Component	Display Name	Label Field	Position	Max Child Items	Changed
	Activities 1	FINS DB Hierarchy Activity 1	Activities	Rel Hierarchy Calc Field	1.1.1.1	3	☑
	Activities 2	FINS DB Hierarchy Activity 2	Activities	Rel Hierarchy Calc Field	1.2		☑
	Contact Node	FINS DB Hierarchy Contact	Contact	Rel Hierarchy Calc Field	1.1.1		☑
	Houuseholds	FINS DB Hierarchy Household	Household	Household Name	1.1		☑
	RootContact	FINS DB Hierarchy Root Contact	Contact	Full Name	1		☑

The following image shows an example of the tree nodes to modify when creating a Financial Accounts hierarchy applet.



W	Name	Business Component	Display Name	Label Field	Position	Max Child Items
	Contact Node	FINS DB Hierarchy Contact	Contact	Rel Hierarchy Calc Field	1.1.1	
	FA1	FINS DB Hierarchy FA 1	Financial Acco...	Rel Hierarchy Calc Field	1.1.1.1	3
	FA2	FINS DB Hierarchy FA 2	Financial Acco...	Rel Hierarchy Calc Field	1.2	
	Households	FINS DB Hierarchy Household	Household	Household Name	1.1	
	RootContact	FINS DB Hierarchy Root Contact	Contact	Full Name	1	

The following image shows an example of the tree nodes to modify when creating a Service Request hierarchy applet.

Tree Nodes

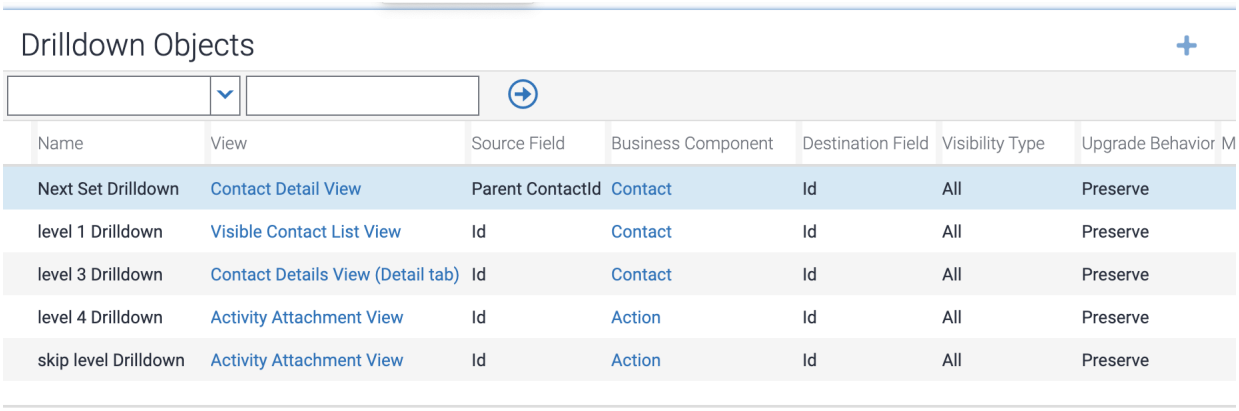
W	Name	Business Component	Display Name	Label Field	Position	Max Child Items
	Contact Node	FINS DB Hierarchy Contact	Contact	Rel Hierarchy Calc Field	1.1.1	
	Households	FINS DB Hierarchy Household	Household	Household Name	1.1	
	RootContact	FINS DB Hierarchy Root Contact	Contact	Full Name	1	
	SR	FINS DB Hierarchy SR 1	Service Requests	Rel Hierarchy Calc Field	1.1.1.1	3
	Service Requests	FINS DB Hierarchy SR 2	Service Requests	Rel Hierarchy Calc Field	1.2	

- **Modify the drilldown object configuration for the copied hierarchy applet.** For the newly copied hierarchy applet, change the property values for the drilldown objects shown in the following table. Modify the following properties as required to support the new entity: View, Source Field, Business Component, and Destination Field.

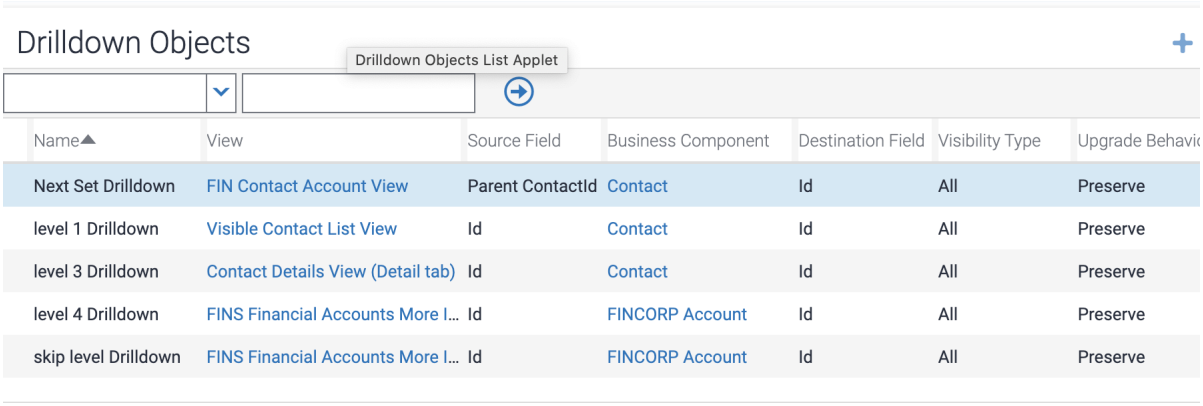
Name	View	Source Field	Business Component	Destination Field
Next Set Drilldown	<View Name>	<Source Field>	<Destination BC>	<Destination Field>
level 4 Drilldown	<View Name>	<Source Field>	<Destination BC>	<Destination Field>
skip Level Drilldown	<View Name>	<Source Field>	<Destination BC>	<Destination Field>

Name	View	Source Field	Business Component	Destination Field

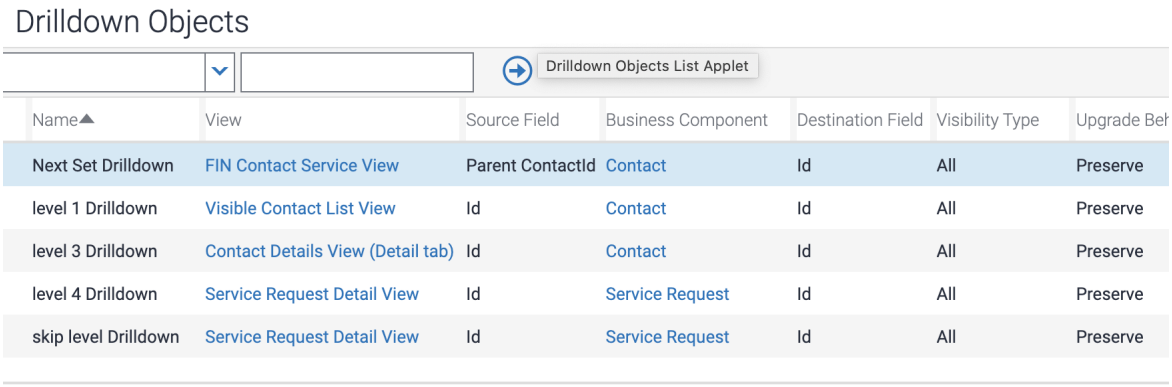
The following image shows an example of the drilldown objects to modify when creating an Activities hierarchy applet.



The following image shows an example of the drilldown objects to modify when creating a Financial Accounts hierarchy applet.



The following image shows an example of the drilldown objects to modify when creating a Service Requests Accounts hierarchy applet.



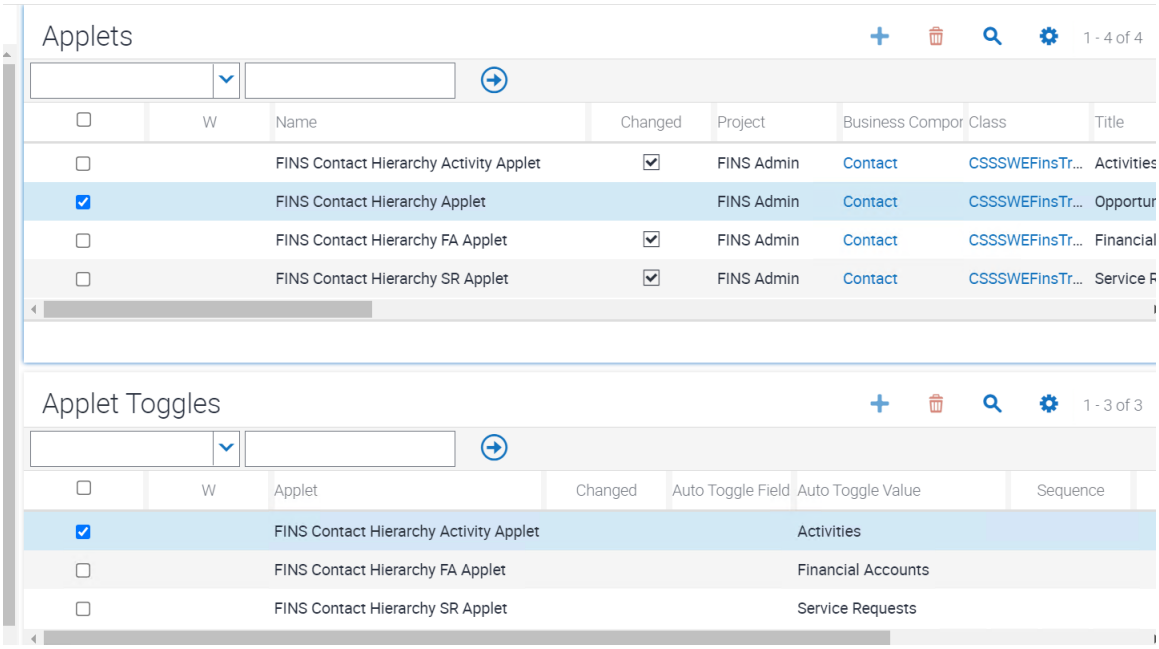
- **Map applet toggles.** For the main hierarchy applet (FINS Contact Hierarchy Applet in this example), you must configure the applet toggles that provide additional choices for level-4 data to display in the hierarchy applet. As described earlier, the default configured level-4 entity for this example is Opportunities, but alternative data such as Activities, Financial Accounts, or Service Requests can also be displayed at position 1.1.1.1 or position 1.2 in the hierarchy applet. Business components and links already configured for this purpose are listed in the information about mapping business components. Displaying these entities requires creating applet toggle records for the main hierarchy applet.

  For the main hierarchy applet (FINS Contact Hierarchy Applet in this example), navigate to Applet Toggles. Then create new records that specify each required applet toggle for each required entity. Also specify the Auto Toggle Value to indicate the entity name that is selectable in the applet toggle drop-down menu at the top of the hierarchy applet. The following table identifies some of this information to help you add applet toggles for FINS Contact Hierarchy Applet. Substitute entries for your use case as needed.

Entity	Applet	Auto Toggle Value
Activities	FINS Contact Hierarchy Activity Applet	Activities
Financial Accounts	FINS Contact Hierarchy FA Applet	Financial Accounts
Service Requests	FINS Contact Hierarchy SR Applet	Service Requests

Entity	Applet	Auto Toggle Value

The following image shows an example of the applet toggle records required to support Activities, Financial Accounts or Service Requests (for FINS Contact Hierarchy Applet).



Mapping the Presentation Model and Physical Renderer

You must register the new hierarchy applet (such as FINS Contact Hierarchy Applet in this example) and all related toggle hierarchy applets with presentation model and physical renderer settings. In these tasks, the Files records you create specify the files siebel/relationshiphierpm.js (for presentation model) and siebel/relationshiphierpr.js (for physical renderer). These tasks are needed for automation support and for supporting a custom CSS style class. Note the following:

- The client presentation model file relationshiphierpm.js retrieves data in a property set named "root" and is mapped to the client-side controls.
- The client physical renderer file relationshiphierpr.js renders data in tile format.

Also map the presentation model relationshiphierviewpm.js to the view displaying the hierarchy applet. Doing this is necessary to maintain the context of the selected position 1.1 tab when the user toggles between different toggle applets.

To map the presentation model to the hierarchy applet

1. Navigate to Administration - Application screen and then to the Manifest Administration view.

   You must register the new applet (such as FINS Contact Hierarchy Applet) with new presentation model settings.

2. In the UI Objects list, create a new record with properties shown in the following table.

Type	Usage Type	Name
Applet	Presentation Model	<AppletName>

3. In the Object Expression list, create a new record with property shown in the following table.

Level
1

4. In the Files list, create a new record with properties shown in the following table.

Level	Name
1	siebel/relationshiphierpm.js

To map the physical renderer to the hierarchy applet

1. Navigate to Administration - Application screen and then to the Manifest Administration view.

   You must register the new applet (such as FINS Contact Hierarchy Applet) with new physical renderer settings.
2. In the UI Objects list, create a new record with properties shown in the following table.

Type	Usage Type	Name
Applet	Physical Renderer	<AppletName>

3. In the Object Expression list, create a new record with property shown in the following table.

Level
1

4. In the Files list, create a new record with properties shown in the following table.

Level	Name
1	siebel/relationshiphierpr.js

To map the presentation model to the view

1. Navigate to Administration - Application screen and then to the Manifest Administration view.

   You must register the view displaying the hierarchy applet with new presentation model settings.
2. In the UI Objects list, create a new record with properties shown in the following table.

Type	Usage Type	Name
View	Presentation Model	<ViewName>

3. In the Object Expression list, create a new record with property shown in the following table.

Level
1

4. In the Files list, create a new record with properties shown in the following table.

Level	Name
1	siebel/relationshiphierviewpm.js

Mapping the Hierarchy Applet to a View Web Template

You must also map the main hierarchy applet (FINS Contact Hierarchy Applet in this example) to the view that is to display the hierarchy applet. Typically, this is the view representing a particular dashboard into which you are mapping multiple applets for different data visualization components.

The view web template for the view FINS Contact Dashboard View (for the Siebel Financial Services dashboard), for example, has a placeholder for the hierarchy applet. For example, review the following content (which is highlighted in the following image) for the view web template Fins Dashboard View:

```
<div class="siebui-timeline-container siebui-span-x1-8 siebui-span-1g-8 siebui-span-md-8 siebui-span-sm-12">
 <div od-type="applet" od-id="201" od-property="FormattedHtml" hintMapType="Applet" hintText="Applet" od-
context="Parent"/>
```

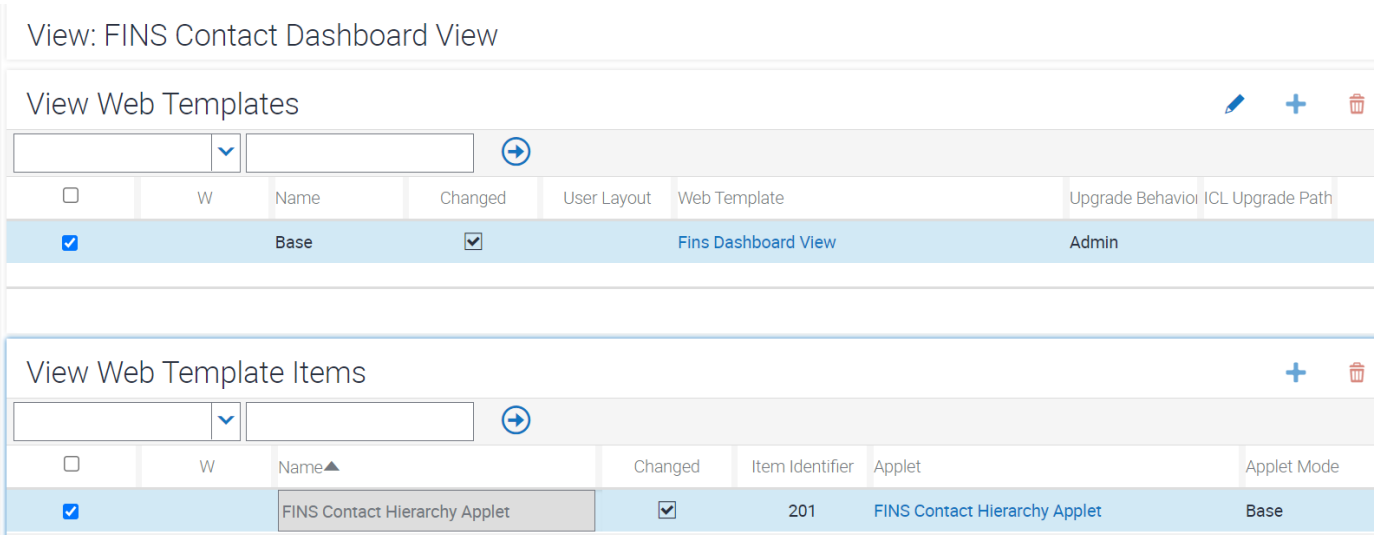
Fins Dashboard View

```
<!--  Template Start: Fins Dashboard View -->
<div class="siebui-db-fins">
  <div class="siebui-infolet-container siebui-span-xl-12 siebui-span-lg-12 siebui-span-md-12 siebui-span-sm-12">
    <div class="siebui-contact-container">
      <div od-type="applet" od-id="1" od-property="FormattedHtml" hintText="Contact Applet"/>
    </div>
    <div od-iterator="currentId" od-id="[2:5]">
      <div od-type="applet" od-id="od-attr-currentId" od-property="FormattedHtml" hintMapType="Applet" hintText="Applet" od-context
    <!--od section iterator close-->
    </div>
  </div>
  <div class="siebui-hierarchy-container siebui-span-xl-8 siebui-span-lg-8 siebui-span-md-8 siebui-span-sm-12">
    <div od-type="applet" od-id="201" od-property="FormattedHtml" hintMapType="Applet" hintText="Applet" od-context="Parent"/>
  </div>
  <div class="siebui-timeline-container siebui-span-xl-4 siebui-span-lg-4 siebui-span-md-4 siebui-span-sm-12">
     <div od-type="applet" od-id="204" od-property="FormattedHtml" hintMapType="Applet" hintText="Applet" od-context="Parent"/>
  </div>
</div>
```

Only the main hierarchy applet (FINS Contact Hierarchy Applet in this example) must be mapped in the view. In any new view to display a hierarchy applet, this hierarchy applet must be mapped to the view. You must also add the necessary classes to the view web template to achieve the required height and width. You can customize or override the styling on the existing CSS classes of the hierarchy applet. The view web template item must specify the applet mode Base for a hierarchy applet that you map. For example, create a view web template item with the properties shown in the following table.

Name	Applet	Applet Mode	Item Identifier
<Applet Name>	<Hierarchy Applet Name>	Base	<Item Identifier for Hierarchy>

The following image shows a sample configuration of the view web template and view web template items, for this example (FINS Contact Dashboard View).



Creating a New Data Visualization Dashboard

This topic provides summary information about how to create a new data visualization dashboard. Some of these tasks are detailed elsewhere in this chapter, such as in topics about configuring particular data visualization components. Some specific tasks vary according to the different data visualization components used.

To create a new data visualization dashboard

1. Create a dashboard view.
2. Create a Siebel web template as per your requirements. Provide placeholders for different data visualization components or applets in the view, as per your design.
3. Create applets such as a salutation applet, infolets, timeline, charts, and so on. These can be custom applets or standard applets. If you are using custom applets such as infolets, timeline, or hierarchy components, then you must create applet web templates.
4. Add each applet to the view web template.
5. Add any styling changes and presentation model or physical rendering mapping.
6. Map the view to the corresponding Siebel CRM application screen for display.
7. Map the relevant roles and responsibilities.

27 Configuring the Customer Dashboard

Configuring the Customer Dashboard

This chapter describes how to configure the Customer Dashboard. It includes the following topics:

- *Overview of the Customer Dashboard*
- *Enabling the Customer Dashboard*
- *Process of Configuring the Customer Dashboard*
- *Modifying the Appearance and Layout of the Customer Dashboard*
- *Options to Update the Customer Dashboard*

Overview of the Customer Dashboard

The *Customer Dashboard* is a feature that provides access to customer information, such as contact name and account number. It remains persistent as the user navigates through Siebel CRM. The Customer Dashboard is displayed as part of the Communications Panel in the Siebel client. For more information, see *Siebel Fundamentals* and *Siebel CTI Administration Guide* .

Note the following:

- Siebel CRM updates the Customer Dashboard based on user actions or software events. The user can either select Update Customer Dashboard in the View menu to open the dashboard with updated information, or open the Communications Panel manually and then click Update in the Customer Dashboard to update the dashboard. For more information, see *Options to Update the Customer Dashboard*.
- This information remains in the Customer Dashboard until Siebel CRM runs the Clear Dashboard command.
- During a session, Siebel CRM saves all of the data that it displays in the Customer Dashboard. The user can use the Forward and Backward buttons to display this stored data.
- You can configure a button on an applet that updates the Customer Dashboard with information from the currently chosen row in an applet. For more information, see *Configuring a Button to Update the Customer Dashboard*.
- You can configure the Customer Dashboard to display data from any business component.
- For the Customer Dashboard, the following label names are predefined: Label 1, Label 2, Label 3, Label 4, Label 5, Label 6, and Label Time. The following field names are predefined: Field 1, Field 2, Field 3, Field 4, Field 5, Field 12, and Field Time. Only these labels and fields can be modified. To map new fields, you must modify the Siebel web template, AppletDashboard. See also *Mapping a Business Component Field to a Customer Dashboard Field*.

Object Types That the Customer Dashboard Uses

The following table describes the object types that the Customer Dashboard uses. All object types use Persistent Customer Dashboard in the Name property except the Business Service Method object type. For example, the Customer

Dashboard references the Persistent Customer Dashboard virtual business component. This business component references the Persistent Customer Dashboard business object.

Object Type	Description
Business Object	Groups together business components that can update the Customer Dashboard.
Business Component	A virtual business component.
Business Service	Controls functionality of the Customer Dashboard.
Applet	Displays data in the Siebel client.
View	Displays applets in the Siebel client.
Business Service Method	Updates the Customer Dashboard. Upon receiving the arguments, the methods evaluate the set of fields to display, gets the data, and then enters the data into the Customer Dashboard.

Enabling the Customer Dashboard

Siebel CRM enables the Customer Dashboard for Siebel Call Center, Siebel Sales, and Siebel Service, by default. You can enable the Customer Dashboard for other applications.

To enable the Customer Dashboard

1. In Siebel Tools, display the Business Service User Prop object type, which is a child of the Business Service object type.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. In the Object Explorer, click Business Service.
3. In the Business Services list, locate the Persistent Customer Dashboard business service.
4. Verify that the Inactive property does not contain a check mark, which is the default setting.
5. In the Object Explorer, expand the Business Service tree, and then click Business Service User Prop.
6. In the Business Service User Props list, query the Name property for Applications.
7. Add the Siebel application to the Value property.

   For example, to use Customer Dashboard for the Siebel Call Center application, the user property must include Siebel Universal Agent, as shown in the following table.

Name Property	Value Property
Applications	Siebel Universal Agent; Siebel Field Service; Siebel Sales Enterprise

Process of Configuring the Customer Dashboard

To configure the Customer Dashboard to display data, perform the following tasks:

1. *Adding a Business Component to the Customer Dashboard*.
2. *Mapping a Business Component Field to a Customer Dashboard Field*.
3. Also perform any necessary tasks described in *Modifying the Appearance and Layout of the Customer Dashboard*, such as *Creating a Label for a Customer Dashboard Field*.

Adding a Business Component to the Customer Dashboard

This task is a step in *Process of Configuring the Customer Dashboard*.

The Customer Dashboard displays a set of fields from multiple business components, such as the Account, Contact, Employee and Service Request business components. You can configure the Customer Dashboard to display information from other business components. To do this, you add the business component to the Persistent Customer Dashboard business object.

The Customer Dashboard does not simultaneously display data from multiple business components. It does display data in different contexts. For example, Siebel CRM displays the following information in the Customer Dashboard:

- If the user is in the Accounts screen and clicks Update, then it displays account data.
- If the user is in the Contacts screen and clicks Update, then it displays contact data.

To add a business component to the Customer Dashboard

1. In Siebel Tools, display the Business Service User Prop object type, which is a child of the Business Service object type.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. Associate the business component with the business object:
   a. In the Object Explorer, click Business Object.
   b. In the Business Objects list, query the Name property for Persistent Customer Dashboard.
   c. In the Object Explorer, expand the Business Objects tree, and then click Business Object Component.
   d. To determine whether a record already exists for the business component that must provide data to the Customer Dashboard, examine the Bus Comp property. If it does exist, then exit this task.
   e. Add a new business object component using values from the following table.

Property	Description
Bus Comp	Choose the name of the business component you must add.

3. Define the business component list:
   a. In the Object Explorer, click Business Service.
   b. In the Business Services list, locate the Persistent Customer Dashboard business service.
   c. In the Object Explorer, expand the Business Service tree, and then click Business Service User Prop.

d. In the Business Service User Props list, add a user property using values from the following table. For more information, see *About the Business Component List*.

Property	Description
Name	Enter a name for this business service user property that represents the business component list. Use the following format: `List integer` where integer is a number. For example: `List 1`
Value	Enter the name of the business component and the names of the business component fields that must provide data to the Customer Dashboard. Use the following format: `business component name;business component field name;business component field name` where: - business component name is the value that Siebel Tools displays in the Name property of the business component. You must begin this list with the business component name. - business component field name is the value that Siebel Tools displays in the Name property of the business component field. If you list multiple field names, then you must use a semicolon to separate each name, as in the following example: `Contact;Last Name;First Name;Full Name`

About the Business Component List

A *business component list* identifies the business component and the list of business component fields that provide data to the Customer Dashboard. To create this list, you define the following properties of a user property of the Persistent Customer Dashboard business service:

- **Name.** Identifies the name of the business component list.
- **Value.** Identifies the name of the business component and the list of business component fields that constitute the business component list.

The following table describes the predefined List 1 business component list for contacts and List 2 for opportunities.

Name Property	Value Property
List 1	Contact;Last Name;First Name;Full Name;Email Address;Work Phone #;Account;Account Location;Fax Phone #;Job Title;Mobile Phone #
List 2	Opportunity;Name;Account;Account Location;Oppty Id;Close Date;Sales Rep;Revenue;Sales Stage

Mapping a Business Component Field to a Customer Dashboard Field

This task is a step in *Process of Configuring the Customer Dashboard*.

A *customer dashboard field* is a field that Siebel CRM displays in the Customer Dashboard. To create this field, you define a business service user property for the Persistent Customer Dashboard business service. You define the following properties for this business service user property:

- **Name.** Identifies the name of the Customer Dashboard field. For example, Field 1.
- **Value.** Identifies the business component list and a field from the business component list.

Siebel CRM predefines the following field names for the Customer Dashboard:

- Field 1.
- Field 2.
- Field 3.
- Field 4. Formatted to display a phone number.
- Field 5.
- Field 12.
- Field Time.

To map a business component field to a customer dashboard field

1. In the Object Explorer, click Business Service.
2. In the Business Services list, locate the Persistent Customer Dashboard business service.
3. In the Object Explorer, expand the Business Service tree, and then click Business Service User Prop.
4. In the Business Service User Props list, add a user property using values from the following table.

Property	Description
Name	Enter a name for this business service user property that represents the customer dashboard field. Use the following format: `Field integer` where: o integer is a number. For example: `Field 1`
Value	You must use the following format:

Property	Description
	`name of the business component list.position` where: o position is the position of the business component field in the business component list. Consider the following examples from the table in *About the Business Component List*: o Assume you must reference the Last Name field of the Contact business component. This field is the first field that is included in the Value property for the List 1 business component list. You use the following format: `List 1.1` o Assume you must reference the First Name field of the Contact business component. This field is the second field that is included in the Value property for the List 1 business component list. You use the following format: `List 1.2`

Configuring a Customer Dashboard Field to Display Data According to Context

You can display data from fields from more than one business component in a single customer dashboard field. To do this, you define multiple values in the Value property of a business service user property for the Persistent Customer Dashboard business service.

The Customer Dashboard business service searches through the list of user properties, starting with Field, and looks for fields that Siebel CRM maps to the Customer Dashboard from the current business component.

If you create a new Customer Dashboard field, then make sure to include a member from every business component list. If you do not do this, then the Customer Dashboard might retain data from the previous business component when Siebel CRM updates the dashboard.

To configure a customer dashboard field to display data according to context

1. In Siebel Tools, in the Object Explorer, click Business Service.
2. In the Business Services list, locate the Persistent Customer Dashboard business service.
3. In the Object Explorer, expand the Business Service tree, and then click Business Service User Prop.
4. In the Business Service User Props list, add a user property using values from the following table.

Property	Description
Name	Enter a name for this business service user property that represents the customer dashboard field. For example: `Field 1`
Value	Reference each field from each business component list. You must use the following format: `name of the business component list.position;name of the business component list.position`

Property	Description
	where: o position is the position of the business component field in the business component list. For example, using the business component lists described in the table in *About the Business Component List*, assume you must display the following information: o If the Customer Dashboard is in the Contacts context, then the Field 1 customer dashboard field must display the Last Name of the contact. o If the Customer Dashboard is in the Opportunities context, then the Field 1 customer dashboard field must display the Opportunity Name. You use the following code to implement this example: `List 1.1;List 2.1` List 1.1 represents the first field of List 1. List 2.1 represents the first field of List 2.

Example of Configuring a Customer Dashboard Field to Display Data According to Context

Assume the Contact business component is active. The Persistent Customer Dashboard business service does the following:

- Locates business service user properties with Field in the Name property.
- If a Field business service user property references the Contact business component, then Siebel CRM displays data from the business component field.
- If a Field business service user property does not reference the Contact business component, then Siebel CRM does nothing and the Customer Dashboard field remains empty.
- If a Field business service user property references the Opportunity business component, and if the Opportunity business component is active, then Siebel CRM displays data from the Opportunity business component field.

Modifying the Appearance and Layout of the Customer Dashboard

This topic describes how to modify the appearance and layout of the Customer Dashboard. It includes the following information:

- *Creating a Label for a Customer Dashboard Field*
- *Formatting a Customer Dashboard Phone Number Field*
- *Modifying the Go To List in the Customer Dashboard*
- *Modifying the Background Color and Border of the Customer Dashboard*
- *Adding a Custom Control to the Customer Dashboard*

- *Modifying a Custom Control in the Customer Dashboard*

Creating a Label for a Customer Dashboard Field

This topic is part of *Modifying the Appearance and Layout of the Customer Dashboard*.

Siebel CRM modifies the field labels that it displays in the Customer Dashboard depending on the data that it displays in the Customer Dashboard. For example:

- If it displays contact information, then the labels are Customer Name, Work Phone #, Email Address, and so on.
- If it displays opportunity information, then the labels are Opportunity Name, Account, Sales Stage, and so on.

If no data is available for the Customer Dashboard, then Siebel CRM displays labels for the default business component. The default business component is defined in the Persistent Customer Dashboard business service. The Contacts business component is predefined as the default business component.

The Siebel runtime repository contains placeholder controls, such as Label 1, Label 2, and so on. It contains predefined business service user properties, named Label 1, Label 2, and so on. These business service user properties map the placeholder labels to fields in the Customer Dashboard.

If you add a field to the Customer Dashboard, then you must define the label that replaces the placeholder label in the Siebel client. To create the label, you create an applet control for each business component field that you must display. The naming format for the applet control identifies it as a Label and identifies the business component and field that determine when Siebel CRM displays it.

To create a label for a customer dashboard field

1. In Siebel Tools, in the Object Explorer, click Applet.
2. In the Applets list, query the Name property for Persistent Customer Dashboard Applet.
3. In the Object Explorer, expand the Applet tree, and then click Control.
4. In the Controls list, create a new control using values from the following table.

Property	Description
Name	Enter a name. Use the following format: `Label business component name.business component field name` For example, to reference the SR Number field of the Service Request business component, you enter the following: `Label ServiceRequest.SR Number`
Caption	Define the label text that Siebel CRM must display in the Customer Dashboard. For example, to display the text SR Number before the customer dashboard field, you enter the following: `SR Number`

5. Repeat the previous step for each label that you must display in the Customer Dashboard.
6. Test and then deliver your Workspace.

Formatting a Customer Dashboard Phone Number Field

This topic is part of *Modifying the Appearance and Layout of the Customer Dashboard*.

You can configure the Customer Dashboard to recognize different telephone extensions. You use the Phone Number Prefix business service user property to define the parameters that associate a telephone switch extension to a complete phone number.

To format a Customer Dashboard phone number field

1. In Siebel Tools, in the Object Explorer, display the Business Service User Prop object type, which is a child of the Business Service object type.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. In the Object Explorer, click Business Service.
3. In the Business Services list, locate the Persistent Customer Dashboard business service.
4. In the Object Explorer, expand the Business Service tree, and then click Business Service User Prop.
5. In the Business Service User Props list, add a user property using values from the following table.

Property	Description
Name	Enter a name for this business service user property that represents the phone number prefix. You must use the following format: `Phone Number Prefix integer or letter` For example: `Phone Number Prefix 1`
Value	Define the parameters for the phone number prefix. You must use the following format: `extension digits;remove digits;prefix` where: o extension digits is number of digits in the extension. o remove digits is the number of digits to remove from the front of the extension. o prefix is the prefix to append to the beginning of the number. For example: `5;1;650555` Assume the main number for your organization is 650-555-0000. A user dials the 24565 extension. In this example, Siebel CRM does the following: o Specifies that the extension is 5 digits in length.

Property	Description
	○ Removes the first digit of the extension, which is the number 2. ○ Adds the 650555 prefix to the remaining part of the extension, which is 4565. The resulting phone number is 650-555-4565.

6. Test and then deliver your Workspace.

Modifying the Go To List in the Customer Dashboard

This topic is part of *Modifying the Appearance and Layout of the Customer Dashboard*.

The Go To list in the Customer Dashboard allows the user to navigate to other views that Siebel CRM relates to the current record. Siebel CRM modifies the list of views depending on the data that it currently displays in the Customer Dashboard. In the Siebel client, the Persistent Customer Dashboard business service does the following:

1. To locate records that begin with View in the Name property, searches the list of user properties.
2. Locates the display name for the associated view.
3. Adds the name to the Go To list.

The following table describes some predefined business service user properties. Each business service user property represents a view that Siebel CRM displays in the Go To list. For example, View 1 specifies that if the Customer Dashboard contains data from the Contact business component, then it displays the All Activities view in the GoTo View list. If the user chooses the All Activities view from the Go To list, then it displays only records for the current Contact ID in the view.

Name	Value
View 1	Contact; All Activity List View; Activity List Applet With Navigation; Contact Id
View 2	Contact; Contact Activity Plan; Contact Form Applet
View 3	Contact; Agreement List View; Agreement List Applet No Parent; Contact Person Id

To modify the Go To list in the Customer Dashboard

1. In Siebel Tools, in the Object Explorer, display the Business Service User Prop object type, which is a child of the Business Service object type.

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. In the Object Explorer, click Business Service.
3. In the Business Services list, locate the Persistent Customer Dashboard business service.
4. In the Object Explorer, expand the Business Service tree, and then click Business Service User Prop.

5. To add a view to the Go To list, in the Business Service User Props list, add a user property using values from the following table.

Property	Description
Name	Enter a name for this business service user property that represents a view that Siebel CRM displays in the Go To list. You must use the following format: `View integer` For example: `View 1`
Value	Create a view that Siebel CRM displays in the Go To list. You must use the following format: `business component name; view name; name of the primary applet on the view; name of the foreign key field` For example: `Contact; All Activity List View; Activity List Applet With Navigation; Contact Id` The name for each of these items must match exactly the name that is defined in the Siebel runtime repository. The foreign key field is conditional. For more information, see *Referencing a Foreign Key Field from the Go To List*.

6. Optional. Configure a label for the view.

   For more information, see *Configuring the Label for the View in the Go To List*.
7. To modify a view or remove a view from the Go To list, do the following:
   a. Locate the view in the Business Service User Props list.
   b. Modify or delete the record, as required.
8. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Referencing a Foreign Key Field from the Go To List

If a view in the Go To list references a business component other than the current business component that provides data to the Customer Dashboard, then you must reference the name of the foreign key field.

For example, if the Customer Dashboard currently displays data from the Contact business component, and if the All Activities view is listed in the GoTo list, then you must define the Contact Id field as the foreign key field. The Contact Id field is the foreign key field in the Action business component that references the Contacts business component. This foreign key field allows Siebel CRM to query all activities that it relates to the contact that it currently displays in the Customer Dashboard.

Configuring the Label for the View in the Go To List

You can configure Siebel CRM to display a text label that represents the view that you configure. It displays this label in the GoTo list.

To configure the label for the view in the Go To list

1. In Siebel Tools, in the Object Explorer, click Applet.
2. In the Applets list, query the Name property for Persistent Customer Dashboard Applet.
3. In the Object Explorer, expand the Applet tree, and then click Control.
4. In the Controls list, create a new control using values from the following table.

Property	Description
Name	Enter a name. Use the following format: **`Label name of the user property`** where: o name of the user property is the name of the business service user property that defines the view that Siebel CRM displays in the Go To list. For example, to reference the business service user property that references the All Activity List View in Step 5 in *Modifying the Go To List in the Customer Dashboard*, you enter the following: **`Label View 1`**
Caption	Define the label text that Siebel CRM must display as the list item in the Go To list. For example, to display the text Activities for This Contact, enter the following: **`Activities for This Contact`**

5. Repeat Step 4 for each label that you must display in the Go To list.
6. Test and then deliver your Workspace.

Modifying the Background Color and Border of the Customer Dashboard

This topic is part of *Modifying the Appearance and Layout of the Customer Dashboard*.

You can modify the background color and border of the Customer Dashboard by updating a custom cascading style sheet (CSS) file.

To modify the background color and border of the Customer Dashboard

1. Apply a custom theme for Siebel Open UI.
   For more information, see *Configuring Siebel Open UI* .

2. Locate a custom style sheet file in the following directory of your Siebel Application Interface installation:

   ```
   SAI_ROOT\applicationcontainer_external\siebelwebroot\FILES\CUSTOM
   ```

3. Open the custom style sheet file with a text editor, such as Notepad.
4. Add or modify the following parameters and values, using values appropriate for your customization goals:

   ```
   /*---------------------*/
   /*Dashboard Definitions*/
   /*---------------------*/
   . siebui-dashboard-frame {background:#999999;}
   . siebui-dashboard-frame {border:2px solid #f0f0f0;}
   ```

   The siebui-dashboard-frame background and border parameters use standard HTML color values. In this example, the value #999999 is medium gray and the value #f0f0f0 is light gray.

Adding a Custom Control to the Customer Dashboard

This topic is part of *Modifying the Appearance and Layout of the Customer Dashboard*.

You can add a custom control to the Customer Dashboard.

To add a custom control to the Customer Dashboard

1. In Siebel Tools, open the Object Explorer.
2. Select the Applet object definition Persistent Customer Dashboard Applet.
3. In the Applet Control list, add a new applet control. Specify values for the following:
   - **HTML Type.** Specifies the control type, such as MiniButton, Field, or FieldLabel.
   - **Field.** Specifies the field that this control maps to.
   - **Caption.** Specifies the text to be shown for the control, such as for a button.
   - **Method Invoked.** Specifies the method to be invoked when the button for this control is clicked.
4. Under Applet Web Template, create a new web template item in the Applet Web Template Item list.
5. Add the new control into the web template as a web template item. Write down the Item Identifier ID for the web template item, such as 2902.
6. Under Web Template, locate the Applet Dashboard web template and edit the `definition` to add the new SWE control, as follows:

   For example, to add a new Button control, add a control tag similar to the following:

   ```
   <div od-type="control" id="2902">
   <div od-property="FormattedHtml" hintText="Button"/>
   <!--od section control close-->
   </div>
   ```

   For example, to add a new Field control, add a control tag similar to the following:

   ```
   <div od-type="control" id="1300">
   <div od-property="FormattedHtml" hintText="Field" hintMapLabelId="od-attr-currentId+100"/>
   <!--od section control close-->
   ```

```
</div>
```

For example, to add a new Label control, add a control tag similar to the following:

```
<div od-type="control" id="1702">
<div><div od-property="<FormattedHtml>" hintText="Label"/></div>
<!--od section control close-->
</div>
```

For more information about using the control tag, see *Configuring Siebel Open UI* and *Siebel Developer's Reference* . The ID specified in the control tag must match the Item Identifier ID for the web template item.

7. Deploy your changes to the Siebel runtime repository.

Modifying a Custom Control in the Customer Dashboard

This topic is part of *Modifying the Appearance and Layout of the Customer Dashboard*.

You can modify a custom control in the Customer Dashboard. Doing this is similar to adding a custom control, as described in *Adding a Custom Control to the Customer Dashboard*.

To modify a custom control in the Customer Dashboard

1. Edit the AppletDashboard web template.
2. Determine which field you want to modify.

   See the list of predefined field and label names for custom controls, in *Overview of the Customer Dashboard*.
3. In Siebel Tools, open the Object Explorer.
4. Select the Applet object definition Persistent Customer Dashboard Applet.
5. In the Applet Control list, select the applet control that you want to modify.
6. Under Applet Web Template, in the Applet Web Template Item list, select the web template item corresponding to the field that you want to modify (see Step 2).
7. Modify the web template item as desired.
8. Save your changes and then deploy your changes to the Siebel runtime repository.

Options to Update the Customer Dashboard

This topic describes optional configurations you can define to update the Customer Dashboard. It includes the following information:

- *Configuring a Button to Update the Customer Dashboard*
- *Configuring Communications Events to Update the Customer Dashboard*
- *Configuring SmartScript to Update the Customer Dashboard*
- *Using Siebel Visual Basic or Siebel eScript to Update Information in the Customer Dashboard*
- *Using Personalization to Update the Customer Dashboard*

Overview of Updating the Customer Dashboard

Information in the Customer Dashboard can be updated in multiple ways, including the following:

- **Selected Record.** After choosing a record in a view, the user can either select Update Customer Dashboard in the View menu to open the dashboard with updated information, or open the Communications Panel manually and then click Update in the Customer Dashboard to update the dashboard. Siebel CRM updates the Customer Dashboard with data from fields in the primary business component that the view references. It updates fields in the dashboard with data from the business component record.
- **Communications event.** When the user accepts an incoming call, Siebel CRM can enter contact information from the caller into the Customer Dashboard. For more information, see *Configuring Communications Events to Update the Customer Dashboard*.
- **SmartScript answer.** Siebel CRM can enter the answer to a question from a SmartScript into the Customer Dashboard. For more information, see *Configuring SmartScript to Update the Customer Dashboard*.
- **Scripts.** You can use Siebel Visual Basic or Siebel eScript to update information in the Customer Dashboard or pull information from the Customer Dashboard. For more information, see *Using Siebel Visual Basic or Siebel eScript to Update Information in the Customer Dashboard*.
- **Personalization.** You can use personalization to update information in the Customer Dashboard or pull information from the Customer Dashboard For more information, see *Using Personalization to Update the Customer Dashboard*.
- **Search Center results.** If Siebel CRM cannot identify the customer from an inbound call, then the user can search for the contact in the Search Center, and then update the dashboard manually as described earlier.

Configuring a Button to Update the Customer Dashboard

This topic is part of *Options to Update the Customer Dashboard*.

The Customer Dashboard includes application programming interfaces (APIs) to pull information from or push information to the Customer Dashboard through Siebel Visual Basic script or Siebel eScript script. The Customer Dashboard resides in a separate frame, so it requires a user interface event to update customer dashboard fields. To create a user event, you must add a button to an applet that uses a script to call the Update Dashboard command.

To configure a button to update the Customer Dashboard

1. In Siebel Tools, in the Object Explorer, click Applet.
2. In the Applets list, locate the applet you must modify.
3. In the Object Explorer, expand the Applet tree, and then click Control.
4. In the Controls list, add a button as an applet control and set properties using values from the following table.

Property	Value
Target View Frame	Dashboard

5. Define the script for the new button.

   Your script must do the following:

   - Call the Update Dashboard command. For more information, see *Using Siebel Visual Basic or Siebel eScript to Update Information in the Customer Dashboard*.
   - Use the InvokeMethod function and send a set of name-value pairs. The following are examples of name-value pairs:

     - Source Name: Base View
     - BusComp Name: Contact
     - RowId: srowid

6. Test and then deliver your Workspace.

   For more information, see *Using Siebel Tools* .

Updating the Customer Dashboard With Data from a Virtual Business Component

An update to the Customer Dashboard requires a row ID. A virtual business component uses a virtual row ID. To display information from a virtual business component in the customer dashboard, you must script the UpdateDashboard function.

Customer Dashboard Allows Only One Update for Each User Interface Event

The Customer Dashboard allows only one user interface update for each user interface event. For example, if you add a button to a view, then one click of that button is one user interface event. For that event, Siebel CRM can run only one user interface update, such as updating the Customer Dashboard. The code behind a single button cannot include two user interface updates, such as updating the Customer Dashboard, and then displaying a new view in the main frame of the Siebel application.

Configuring Communications Events to Update the Customer Dashboard

This topic is part of *Options to Update the Customer Dashboard*.

You can use communications events to update the Customer Dashboard. The following are examples of communications events:

- Inbound email message
- Voice call

For example, you can define communications events to update the Customer Dashboard with contact information. To meet your display requirements, you can configure any communications event for any business component. (You can also configure communications commands in a similar way, as appropriate for your implementation.) For more information, see *Siebel CTI Administration Guide* . See also *Siebel Email Administration Guide* .

When a communications event is triggered, the Update Dashboard from CTI method of the Persistent Customer Dashboard business service can be invoked, based on how communications events are configured in your environment. For an example of how event parameters can function in your application, see the following table.

Siebel CRM predefines the CTI administration views to call the UpdateDashboard business service method when a significant event occurs, and to send variables as arguments. Example variables include the phone number and number of calls in queue.

To update the Customer Dashboard as a result of a communications event, you must call the method to update the Customer Dashboard and send the following parameters:

- Business component name
- Name of the business component field
- Value that you require from this communications event

For example, the parameters listed in the following table configure the Customer Dashboard to get data from contact information for the contact for whom Work Phone # matches the ANI (automatic number identification) of the inbound call.

Parameter	Example Value
ServiceMethod	Persistent Customer Dashboard.Update Dashboard from CTI
ServiceParam.Field	Work Phone #
ServiceParam.Value	{ANI}
ServiceParam.BuscompName	Contact

Calling the Customer Dashboard Business Service from the Communications Event Log

You can call the Customer Dashboard business service from the communications event log.

To call the Customer Dashboard business service from the communications event log

1. In the Siebel client, navigate to the Administration - Communications screen, and then click the All Event Handlers link.
2. Query the Name property of the Event Handlers list for InboundCallReceived.
3. Click the Associated Event Logs tab.
4. Click the LogIncomingCallContactFound link.
5. In the Event Log Parameters list, set the parameters. For example, for contacts, you can use values from the following table.

Name	Value
ServiceMethod	Persistent Customer Dashboard.Update Dashboard from CTI
ServiceParam.Field	Id
ServiceParam.Value	{Contact.Id}

Name	Value
WorkTrackingObj.ContactId	{Contact.Id}

Configuring SmartScript to Update the Customer Dashboard

This topic is part of *Options to Update the Customer Dashboard*.

You can configure the Customer Dashboard so that Siebel CRM updates the Customer Dashboard with the answer to a question that it gets from a SmartScript script. For more information about Siebel SmartScript, see *Siebel SmartScript Administration Guide* .

To configure SmartScript to update the Customer Dashboard

1. In Siebel Tools, in the Object Explorer, display the following object types:
   - Applet User Prop object type, which is a child of the Applet object type
   - Business Service User Prop object type, which is a child of the Business Service object type

   For more information, see *Displaying Object Types You Use to Configure Siebel CRM*.
2. Make sure the SmartScript Player is active:
   a. In the Object Explorer, click Applet.
   b. In the Applets list, locate the Smart Script Player Applet (Tree Only) applet.
   c. In the Object Explorer, expand the Applet tree, and then click Applet User Prop.
   d. In the Applet User Props list, locate the Notify Dashboard applet user property, and then verify the property described in the following table.

Property	Value
Value	Y

3. Map SmartScript variables to customer dashboard fields:
   a. In the Object Explorer, click Business Service.
   b. In the Business Services list, locate the Persistent Customer Dashboard business service.
   c. In the Object Explorer, expand the Business Service tree, and then click Business Service User Prop.
   d. In the Business Service User Props list, create a new record using values from the following table.

Name	Value
SmartScript List	Fname;Lname;Phone;Interests

Name	Value
	These values represent the variables from the SmartScript script that Siebel CRM displays in the Customer Dashboard. These values must match exactly the variable names defined in the SmartScript script.

This mapping configuration is similar to defining user properties for a business component list. For more information, see *Mapping a Business Component Field to a Customer Dashboard Field*.

4. Inspect your Workspace.
5. Map SmartScript answers to the Customer Dashboard:
   a. In the Siebel client, navigate to the Administration - SmartScript screen, and then click the Questions link.
   b. In the Questions list, choose a question.
   c. In the More Info form, in the Save User Parameters field, enter `SmartScript List`.

      This step allows Siebel CRM to save the answer as a global variable to the script. The name of the variable you enter must match exactly the name of the business service user property that you defined in Step 3.
   d. Click the Scripts link, and then locate the appropriate script in the Scripts list.
   e. In the Translation form, enter the name of the variables from each question into the Dashboard Text field. Use the following format:

      ```
      [name of variable] [name of variable]
      ```

      For example:

      ```
      [Fname] [Lname]
      ```

      Siebel CRM sends the values for the variables in the Dashboard Text field to the Customer Dashboard when it runs the SmartScript.
   f. Repeat Step a through Step e for each question you must configure for the Customer Dashboard.
6. Test your modifications.

Updating the Customer Dashboard from Scripts That Siebel CRM Runs in SmartScript

You cannot update the Customer Dashboard from Siebel Visual Basic script or Siebel eScript script that runs in a SmartScript script. A one-to-one relationship exists between a user interface event and the ability to update a frame in Siebel CRM. Each user interface event in a SmartScript script updates the SmartScript frame, so it cannot update the Customer Dashboard frame. If you send parameters to the Customer Dashboard from a Siebel Visual Basic script or Siebel eScript script in a SmartScript script, then the Customer Dashboard receives the parameters but it cannot display them.

Using Siebel Visual Basic or Siebel eScript to Update Information in the Customer Dashboard

This topic is part of *Options to Update the Customer Dashboard*.

You can use Siebel Visual Basic or Siebel eScript to update information in the Customer Dashboard or pull information from the Customer Dashboard. The Customer Dashboard is a business service, You must use the following command:

```
GetService("Persistent Customer Dashboard")
```

You use the following commands to pull information from the dashboard:

- GetCurrentContactId
- GetDashboardFieldValue

Command to Get the Record ID of the Current Dashboard Record

This GetCurrentContactId command returns the record ID for the record that Siebel CRM currently displays in the Customer Dashboard. For example:

- If the record is from the Contact business component, then GetCurrentContactId returns the ContactId
- If the record is from the Account business component, then GetCurrentContactId returns the AccountId.

Do not define any input arguments.

Always define ContactId as the output argument. The Customer Dashboard uses the ContactId variable. In this situation, this variable includes the record ID of the business component whose data Siebel CRM currently displays in the Customer Dashboard.

Example of the GetCurrentContactId Command

The following code is an example of the GetCurrentContactId command:

```
bs.InvokeMethod("GetCurrentContactId",inpargs,outargs);

var fvalue = outargs.GetProperty("Contact Id");

// do something with the contact ID
```

Command to Get the Value of the Current Dashboard Field

The GetDashboardFieldValue command returns the current field value of the current record in the Customer Dashboard. The input argument is the name-value pair for the Customer Dashboard field. The output argument is Field Value.

Example of the GetDashboardFieldValue Command

The following code is an example of the GetDashboardFieldValue command:

```
inpargs.SetProperty("Field Name","Field 4");

bs.InvokeMethod("GetDashboardFieldValue",inpargs,outargs);

var fvalue = outargs.GetProperty("Field Value");

// do something with the field value
```

Update Dashboard Command

You use the Update Dashboard command to enter a new record in the Customer Dashboard. This example uses the following name-value pairs as input arguments:

- Source Name: Base View
- Buscomp Name: Contact
- RowId: E301

Example of the Update Dashboard Command

The following code is an example of the Update Dashboard command:

```
inpargs.SetProperty("Source Name","Base View", "Buscomp Name", "Contact", "RowId",
"E301");

bs.InvokeMethod("Update Dashboard",inpargs,outargs);
```

Examples of Using Customer Dashboard Commands with Scripts

The examples in this topic use Customer Dashboard commands to do the following:

- Get the contact ID, Field 4, and Field Time of the current record in the Customer Dashboard.
- Print values of the contact ID, Field 4, and Field Time to a file.

Example of Using Customer Dashboard Commands with Siebel eScript

The following example script is written in Siebel eScript. For more information, see *Siebel eScript Language Reference* :

```
function Script_Open ()
{
 var fn1=Clib.fopen("d:\\sabari5.txt", "wt");
 var bs = TheApplication().GetService("Persistent Customer dashboard");
 var inpargs= TheApplication().NewPropertySet();
 var outargs = TheApplication().NewPropertySet();

 bs.InvokeMethod("GetCurrentContactId",inpargs,outargs);
 var fvalue = outargs.GetProperty("Contact Id");
 Clib.fprintf (fn1, "The current id in the dashboard = %s \n",fvalue);

 inpargs.SetProperty("Field Name","Field 4");
 bs.InvokeMethod("GetDashboardFieldValue",inpargs,outargs);
 var fvalue = outargs.GetProperty("Field Value");
 Clib.fprintf (fn1, "The Account Name in the dashboard = %s \n",fvalue);

 inpargs.SetProperty("Field Name","Field Time");
 bs.InvokeMethod("GetDashboardFieldValue",inpargs,outargs);
 var fvalue = outargs.GetProperty("Field Value");
 Clib.fprintf (fn1, "The current time of the agent/customer in the dashboard =
 %s \n",fvalue);

 Clib.fclose(fn1);
return(ContinueOperation);
}
```

Example of Using Customer Dashboard Commands with Siebel Visual Basic

The following example script is written in Siebel Visual Basic. For more information, see *Siebel VB Language Reference* :

```
Sub Script_Open

 Dim bs as Service
 Dim inpargs as PropertySet
 Dim outargs as PropertySet
 Dim fvalue as String

Open "d:\sabari.txt" for Output as #1
Set bs = TheApplication().GetService("Persistent Customer dashboard")
 Set inpargs = TheApplication.NewPropertySet
 Set outargs = TheApplication.NewPropertySet

 bs.InvokeMethod "GetCurrentContactId",inpargs,outargs
 fvalue = outargs.GetProperty("Contact Id")
```

```
    Write #1, "The current id in the dashboard = " & fvalue

    Inpargs.SetProperty "Field Name","Field 4"
    bs.InvokeMethod "GetDashboardFieldValue",inpargs,outargs
    fvalue = outargs.GetProperty("Field Value")
    Write #1," The Account Name in the dashboard = "& fvalue
    Close #1

End Sub
```

Using Personalization to Update the Customer Dashboard

This topic is part of *Options to Update the Customer Dashboard*.

The Personalization engine can personalize the Siebel Call Center application according to the agent profile and the customer profile. Siebel CRM loads the agent profile when the agent logs into Siebel CRM. It loads the customer profile when it enters values for customer information in the Customer Dashboard. This allows the agent to view customer information according to personalization rules that your Siebel administrator creates.

For example, you can display a different applet or view to the agent according to the customer profile. You can display a Recommended Products applet that only displays products for this customer according to products that the customer previously purchased.

To access the profile information, you create personalization rules. The following attributes allow you to access different types of information:

- **Me attribute.** Provides access to agent profile information.
- **You attribute.** Provides access to customer profile information.

The following are examples of these commands.

- GetProfileAttr("You.Last Name")
- GetProfileAttr("Me.Last Name")

For more information, see *Siebel Personalization Administration Guide* .

28 Reference Materials for Configuring Siebel CRM

Reference Materials for Configuring Siebel CRM

This chapter provides reference information for configuring Siebel CRM applications. It includes the following topics:

- *Properties of Object Types*
- *Types of Applet Controls and List Columns*
- *Objects You Use with Enterprise Integration Manager*
- *Types of Tables and Columns That CIAI Query Supports*
- *Extensive Code Examples That This Book Uses*

Properties of Object Types

This topic describes some of the properties that Siebel CRM frequently uses with various objects that you can configure. It includes the following information:

- *Properties of a Siebel Table*
- *Properties of a Table Column*
- *Properties of an Index of a Siebel Table*
- *Properties of an Index Column*
- *Properties of a Business Component*
- *Type Property of a Business Component Field*
- *Display Format Property of a Control or List Column*
- *Properties of a Screen View*
- *Properties of an Application*
- *Properties of Objects You Use with a Menu or Toolbar*

For a complete list of properties of objects, see *Siebel Object Types Reference* .

Properties of a Siebel Table

The following table describes properties that Siebel CRM commonly uses with a Siebel table.

Property	Description
Name	The name of the table in the RDBMS.

Property	Description
Type	The table type. For more information, see the table in *Type Property of a Table Column*.
Base Table	The base table if the table is an extension table. If the table is a base table, then this property is empty. An extension table always identifies a base table.
User Name	A longer, descriptive name that helps you identify the table.
Comments	A text description of the table, such as the type of data that Siebel CRM stores in the table.
Status	The current status of a table. Indicates if Siebel CRM can use, in the most recent version of Siebel CRM, a table from a previous version of Siebel CRM.

Properties of a Table Column

The following table describes properties that Siebel CRM commonly uses with a table column.

Property	Description
Default	Stores a default value when Siebel CRM adds new table rows.
Foreign Key Table	Stores the table that this column references a foreign key. Siebel Enterprise Integration Manager (EIM) uses this information. For more information, see *Mapping a Custom Table to an Interface Table*.
LOV Bounded	Stores import behavior for EIM. If LOV Bounded is TRUE, then, during import, EIM checks the values against the values contained in a list defined in the LOV Type property. In that situation, LOV data must be imported first into the S_LST_OF_VAL table, and the LOV Type property must be defined. This property is read-only for a predefined column in Siebel CRM but you can edit it for a custom extension column.
LOV Type	Stores the list of values domain that the Siebel schema uses to validate this column. Siebel CRM uses it in conjunction with the LOV Bounded property. You define a list of values domain in the Administration - Data screen, List of Values view in the Siebel client. This property is read-only for a predefined column in Siebel CRM but you can edit it for a custom extension column.
Name	Stores the name of the database column in the database table.
Nullable	Indicates if the Siebel database can or cannot store NULL in this column. If TRUE, then Siebel CRM can store NULL.
Physical Type	For more information, see *Physical Type Property of a Table Column*.
Precision	Stores the maximum number of digits in a number column. For a noninteger column, the precision is 22. For an integer column, the precision is 10.

Property	Description
Primary Key	Stores the primary key for the table. If TRUE, then this column is the primary key for the table. With minor exceptions, the ROW_ID column in a table is the primary key, and it contains a TRUE value for this property.
Scale	Stores the maximum number of digits after the decimal point. For a noninteger column, the scale is 7. For an integer column, the scale is 0.
Type	For more information, see *Type Property of a Table Column*.
User Key Sequence	Stores the sequence in the user key where this column fits.

Physical Type Property of a Table Column

The following table describes the physical types that Siebel CRM supports. The Physical Type property identifies the physical type of the column in the Siebel database.

Physical Type	Description
Character	Stores text that is fixed in length. Also used for a *Boolean column*, which is a character column that contains a length of 1. You cannot use Char greater than 1, by default. To modify the default setting in Siebel Tools, you can click the View menu, Options, and then click the Database tab. Make sure the following option contains a check mark: Allow to create column of type 'Character' being greater than 1 If you define a Column as a Char column, and if the data that Siebel CRM stores in the column varies in length, then it pads the data with empty spaces in the Siebel database. It is recommend that you use the Varchar data type for all but Boolean columns that Siebel CRM defines as CHAR(1).
Character Large Object (CLOB)	For more information, see *Extensive Code Examples That This Book Uses*.
Date	Stores the date only, without time.
Date Time	Stores combined date and time in the same column.
Long	(Obsolete) Use CLOBs to store large amounts of text or other data. If you select this physical type on a Column object, it will be implemented as a CLOB in the physical database.
Number	Stores any numeric data. Typical numeric columns in Siebel CRM are 22,7 for typical numbers, and 10,0 for integers. For more information, see *Physical Type Property of a Table Column*.
Time	Stores time only, without the date.
UTC Date Time	Stores the date and time. Siebel CRM saves time in Greenwich Mean Time (GMT).

Physical Type	Description
Varchar	Stores text that varies in length. Used for most alphanumeric columns in the data objects layer, including ROW_ID, foreign key, list of values, and other free form text columns.

Character Large Object (CLOB) Physical Type

The Character Large Object (CLOB) physical type stores a large, variable amount of text. CLOBs are similar to the deprecated "LONG" physical type, but it can contain much more data. In an Oracle database, the maximum size is (4 GB minus 1 byte) multiplied by the value in DB_BLOCK_SIZE.

Note the following requirements:

- Siebel CRM allows no more than three CLOB columns for each table.
- In Siebel Tools, you can only set the physical type to CLOB when you define a column. You cannot modify a predefined column, such as a Long column, to a CLOB column.
- Siebel Tools displays the CLOB Physical Type as L (Long) in the Properties window.
- Microsoft SQL Server does not implement a CLOB type, so these are implemented at the physical database layer as VARCHAR(max) or NVARCHAR(max).
- To query on a DTYPE_CLOB field, you must use at least one wildcard in the search expression. You use an asterisk (*) to express a wildcard. For example, use TEST*. Do not use an equal sign (=) in the query. For example, do not use =TEST. If you use an equal sign, then Siebel CRM creates an error.

Maximum Number of Digits for a Numeric Physical Type

If the Physical Type property of a table column is Numeric, then the table column can contain up to 16 digits. Note the following for the numeric physical type:

- As Siebel CRM increases the number of digits it uses before the decimal point, the number of usable digits after the decimal point decreases by an equal amount.
- Data is limited to 16 digits without a decimal point.
- If you use a decimal point, then data is limited to 15 digits before the decimal point.
- You cannot use more than 7 digits after the decimal point.
- You cannot modify precision or scale properties to modify this support.
- Some rounding errors can occur with a 16 or 15 digit number.

Type Property of a Table Column

The type property specifies the type of column. Siebel CRM commonly uses the following values for the Type property of a table column:

- **Data.** For more information, see *Data Columns of a Siebel Table*.
- **Extension.** For more information, see *Extension Columns of a Siebel Table*.
- **System.** For more information, see *System Columns of a Siebel Table*.
- **IFMGR.** Occurs in an interface table. The Siebel Enterprise Integration Manager uses it internally. The name of an IFMGR type follows the IFMGR:nn format. For example, IFMGR:ROW_ID.

The following table describes possible values for the Type property of a Siebel table.

Type	Description
Data (Public)	Contains data that Siebel CRM makes available through business components. To configure a public data table, you can use an extension table and extension column. It is among the predefined set of tables.
Data (Private)	Similar to a public data table, except it cannot contain an extension column.
Data (Intersection)	Defines a many-to-many relationship between two data tables.
Extension	Provides more columns that you can use to store data. Contains an implicit one-to-one relationship to the parent base table, which is the table that an extension table logically extends. A table with an implicit one-to-many relationship to a base table is also known as an extension table. The Type property for this type of table is Data (Public), and not Extension.
Interface	Siebel Enterprise Integration Manager (EIM) uses the Interface type to enter values in one or more base tables and subsequently to perform periodic batch updates between Siebel CRM and other enterprise applications. The suffix in the name of an interface table is _IF or _XMIF.
Database View	Reserved for Oracle internal use.
Dictionary	Reserved for Oracle internal use.
Journal	Reserved for Oracle internal use.
Log	Reserved for Oracle internal use.
Repository	Reserved for Oracle internal use.
Virtual Table	Reserved for Oracle internal use.
Warehouse Types	Reserved for Oracle internal use.
Extension (Siebel)	Reserved for Oracle internal use. An Extension (Siebel) table is typically an extension of the S_PARTY table.

Properties of an Index of a Siebel Table

The following table describes some of the properties that Siebel CRM commonly uses with an index.

Property	Description
Name	Stores the name of the database index.

Property	Description
Unique	TRUE indicates that Siebel CRM does not allow multiple rows with the same value. You must not define a custom unique index without assistance. For more information, see *Getting Help From Oracle*.
Type	Indicates the type of index. Siebel CRM commonly uses the following types: • **Primary Key.** An index that Siebel CRM indexes on the ROW_ID column. • **User Key.** A custom index that you define. You specify the set of index columns. It must consist of a unique combination of columns. • **Extension.** An extension index that Siebel Tools creates by default if you add an index. Siebel CRM specifies the set of index columns. • **System.** A predefined index. You must not modify a predefined index.

Properties of an Index Column

The following table describes some of the properties that Siebel CRM commonly uses with an index column.

Property	Description
Column Name	Stores the name of the column where the parent index does an operation, such as a sort operation.
Sequence	An integer value that stores the order of the column in the index relative to other columns. You must define the Sequence property even if only one index column is defined for an index.
Sort Order	Stores the sort order of the index column. The order is one of the following: • Asc (ascending) • Desc (descending)

Properties of a Business Component

The following table describes some of the properties that Siebel CRM commonly uses with a business component.

Property	Description
Class	The C++ class that defines the functionality of the business component. For more information, see *Class Property of a Business Component*.

Property	Description
Name	A name that must be unique among all business components that reside in the Siebel runtime repository. Siebel CRM uses the name to reference a business component.
No Delete No Insert No Update	If set to TRUE, then the user cannot perform the defined data manipulation operation. For example, if No Delete is TRUE, then the user cannot delete a record that is associated with this business component. The default value is FALSE.
Search Specification	A conditional expression that Siebel CRM uses to restrict the records that it gets. Defining the search specification on a business component is very similar to defining the search specification on an applet. For more information, see *Options to Filter Data That Siebel CRM Displays in an Applet*.
Sort Specification	A sort specification that Siebel CRM uses to order the records returned. For more information, see *Determining How a Business Component Sorts Records*.
Table	The name of the SQL table where Siebel CRM gets the records that it uses to update most fields in the business component.

Type Property of a Business Component Field

The following table describes the data types that Siebel CRM includes in the Type property of a business component field. Siebel CRM prefaces all field data types with the following text: DTYPE_.

Field Data Type	Physical Type	Maximum Length	Description
DTYPE_BOOL	Character	1	Describes data as Y or N. Siebel Tools often displays this type in the following ways: • TRUE or FALSE • checked or unchecked
DTYPE_CURRENCY	Number	22	Describes data as currency. You can use the Windows Control Panel to control the appearance of currency values that Siebel CRM displays in a screen. To define an explicit format mask in the Display Format property, you can use the following symbols: • Dollar sign ($). Specifies the position for the currency symbol. • Trailing period (.). Specifies the default precision for the currency. • All valid symbols described for DTYPE_NUMBER.

Field Data Type	Physical Type	Maximum Length	Description
DTYPE_DATE	Date	7	Describes data as a date. When Siebel CRM returns the date, it ignores other information, such as time. For more information, see *Formats for the Date Physical Type*.
DTYPE_DATETIME	Date Time	7	Describes data as a date and time. You can use the Windows Control Panel to set the appearance of time and date values. You can use a combination of DTYPE_DATE and DTYPE_TIME to configure an explicit date format. Alternatively, you can use one of the following properties. Each of these properties use the format configured in the Windows Control Panel: • **Date.** Displays only the date portion of the value. • **Time.** Displays only the time portion of the value. • **TimeNoSec.** Displays only the hour and minute portion of the value.
DTYPE_UTCDATETIME	UTC Date Time	30	Describes data as date information that includes a date and a time component that Siebel CRM stores in the Siebel database in UTC time. UTC is the equivalent of Greenwich Mean Time without any adjustments for daylight savings time. A field of this type must correspond to a database column of type U. The default time zone that is configured in the user preferences determines how Siebel CRM converts the display value for this field to or from UTC time.
DTYPE_ID	Varchar	15	Describes data as the primary key that Siebel CRM creates. A field mapped to an extension column with a physical type of Varchar(15) defaults to a DTYPE_ID data type.
DTYPE_INTEGER	Number	22*	Describes data as a whole number ranging in value from negative 2147483648 to 2147483647.
DTYPE_NOTE	CLOB	16 KB	Describes data as a very long string or series of bytes.
DTYPE_NUMBER	Number	22	Describes data as a number. For more information, see *Formats for a DTYPE_NUMBER Business Component Field*.
DTYPE_PHONE	Number	40	Describes data as a phone number. Siebel CRM ignores the DisplayFormat property for DTYPE_PHONE values.
DTYPE_TEXT	Varchar	2 KB	Describes data as a string that is less than or equal to 4000 bytes. The default value is 255. Siebel CRM ignores the DisplayFormat property for DTYPE_TEXT values.
DTYPE_TIME	Time	7	Describes data as a time. When Siebel CRM gets the time, it ignores other information. You can set the appearance of time values through the Windows

Field Data Type	Physical Type	Maximum Length	Description
			Control Panel, or you can use the following symbols to specify an explicit time: • **HH.** Hour according to a 24-hour clock without a leading zero. • **H.** Hour according to a 24-hour clock with a leading zero. • **hh.** Hour according to a 12-hour clock without a leading zero. • **h.** Hour according to a 12-hour clock with a leading zero. • **mm.** Minute without a leading zero. • **m.** Minute with a leading zero. • **ss.** Second without a leading zero. • **s.** Second with a leading zero. • **Colon (:).** The position of the time separator. You specify the character in the Windows Control Panel.

Formats for the Date Physical Type

You can use the Windows Control Panel to control the appearance of date values that Siebel CRM displays in a screen.

To define an explicit date format, you can use the following symbols:

- **YY.** Two-digit year without a leading zero.
- **Y.** Two-digit year with a leading zero.
- **YYYY.** Four-digit year without a leading zero.
- **YYY.** Four-digit year with a leading zero.
- **MM.** Month without a leading zero.
- **M.** Month with a leading zero.
- **DD.** Day without a leading zero.

For more information, see *How Siebel CRM Handles Certain Date Formats*.

Formats for a DTYPE_NUMBER Business Component Field

You can use the Windows Control Panel to control the appearance of numeric values, or you can use the following symbols to specify an explicit format mask:

- **Zero (0).** Specifies the position of a mandatory digit.
- **Pound sign (#).** Specifies the position of an optional digit.
- **Comma (,).** Specifies the position of the thousands separator. You specify the character in the Windows Control Panel.
- **Period (.).** Specifies the position of the decimal separator. You specify the character in the Windows Control Panel.
- **Trailing period (.).** Specifies default display precision.
- **Plus sign (+).** Specifies the position and appearance of the positive value indicator.
- **Minus sign (-).** Specifies the position and appearance of the negative value indicator.

Restrictions exist regarding the number of digits you can use with a business component field whose Type property is DTYPE_NUMBER. For more information, see *Physical Type Property of a Table Column*.

Display Format Property of a Control or List Column

The following table describes values for the Display Format property of a control or list column.

Property	Description
DTYPE_NUMBER	Can include the following values: • 0 (zero) • # (pound sign) • + (plus sign) • - (minus sign) • , (comma) • . (period) If empty, then to determine the appearance of numeric values, Siebel CRM uses the Regional Settings section of the Windows Control Panel.
DTYPE_CURRENCY	Uses the same symbols as with the DTYPE_NUMBER property, in addition to the dollar sign. To control how Siebel CRM displays currency, in the Siebel client, you navigate to the Application Administration screen, choose the Currencies view, and then modify the Scale field.
DTYPE_DATETIME	Stores one of the following values in the Display Format property: • Date • DateTime • DateTimeNoSec • TimeNoSec
DTYPE_DATE	Uses a combination of the following values: • MM/DD/YY where: • MM is the month. • DD is the day. • YY is the year. If empty, then to determine the appearance of date values, Siebel CRM uses the Regional Settings section of the Windows Control Panel. For more information, see *How Siebel CRM Handles Certain Date Formats*.
DTYPE_TIME	You can use one of the following formats: • Specify TimeNoSec.

Property	Description
	• Specify a format mask using a combination of the following values: o HH:hh:mm:ss where: o HH is the hours. o hh is the hours. o mm is the minutes. o ss is the seconds. If empty, then to determine the appearance of time values, Siebel CRM uses the Regional Settings section of the Windows Control Panel.
DTYPE_PHONE	If empty, then to determine the appearance of phone values, Siebel CRM uses the Regional Settings section of the Windows Control Panel.

How Siebel CRM Handles Certain Date Formats

If you set DDD, DD/MM/YYYY as the display format on a date list column, then Siebel CRM displays the date with the expected format but you cannot update the date. If you attempt to update the date, then it displays an error that is similar to the following:

```
Can't convert formatted string to its internal representation.
Please check if your data is in correct format.
```

The same situation occurs if you use the DDD/MM/YYYY format.

Note: The Display Format property controls how Siebel CRM displays date values; it does not define every supported input format for *updating* date values. When users update a date field, they must enter a value in the expected locale or date input format. If users enter a date by using an unsupported display format, then Siebel CRM might store an incorrect date or display a conversion error.

Properties of a Screen View

The following table describes some of the properties that Siebel CRM commonly uses with a screen view.

Property	Description
Category Default View	Defines the default view for a screen view as an Aggregate Category or Detail Category. It is recommended that you define the Category Default View property. If this property is empty, then Siebel CRM lists views alphabetically. If the category for the screen view is Aggregate View or Detail View, then the Category Default View property is read-only.

Property	Description
Category Name	Name of the Aggregate Category or Detail Category. This property must be unique in each screen. If the category for the screen view is Aggregate View or Detail View, then the Category Name property is read-only.
Display in Page	If Siebel CRM must display the screen view in the page, then you must set the Display in Page property to TRUE. If the category for the screen view is Aggregate Category or Detail Category, then you must set the Display in Page property to TRUE. If no views in the category are included, then Siebel CRM does not display the screen view.
Display in Site Map	If Siebel CRM must display the screen view on the Site Map, then the Display in Site Map property must be TRUE.
Menu Text	Text that Siebel CRM displays in the Site Map. To modify the text style that it displays in the Site Map, you can use HTML tags in the Menu Text property. For more information, see *Modifying the Text Style of a Control or List Column in an Applet*.
Name	Calculated field.
Parent Category	If the Type property is Aggregate Category, then the Parent Category property is empty because no parent exists. If the Type property is Aggregate View, then the Parent Category property might or might not be empty. If the Type property is Detail View or Detail Category, then the Parent Category property must contain a value.
Type	Can be one of the following values: • Aggregate Category • Aggregate View • Detail Category • Detail View For more information, see *About Screen Views*.
View	If the Type property of the screen view is Aggregate View or Detail View, then you must define the view property. The View property associates the view with the screen view. The View property is read only for Aggregate Category and Detail Category.

Properties of an Application

The following table describes properties of the Application object type.

Property	Description
Acknowledgement Web Page	The Web page that Siebel CRM displays after the user successfully logs in. If a login occurs after a timeout, then it displays the view that the user attempted to access when the timeout occurred.
Acknowledgement Web View	The view that Siebel CRM displays after the user successfully logs in, except in the following situations: • **Timeout.** If the user logs in after a timeout, then Siebel CRM displays the view that the user attempted to access when the timeout occurred. • **Explicit login required.** Assume the following occurs: o An anonymous user logs in to a customer application. o The Explicit Login property is set to TRUE for the view that the user attempts to access. o The user successfully enters the login credentials. In this situation, Siebel CRM displays the view that the user was attempting to access. It does not display the view defined in the Acknowledgement Web View property.
Container Web Page	A Web page that defines the structure of the Siebel application. This page can contain the common user interface components, such as screen bars, view bars, logos, and so on. You can use this page to define the HTML Frame definition document for the Siebel application. Siebel CRM displays all views and, as an option, all pages in the context of the container page. For more information, see *About the Container Page*.
Error Web Page	The page Siebel CRM displays if an error occurs in the Siebel application.
Login Web Page	The page Siebel CRM displays as the login page.
Logoff Acknowledgement Web Page	The page Siebel CRM displays when the user logs out of the Siebel application.
Sort Web Page	The page Siebel CRM uses to create a dialog to perform an advanced sort on list applet columns.

Properties of Objects You Use with a Menu or Toolbar

This topic describes some of the properties of objects that you use with a menu or toolbar.

Properties of a Command

The following table describes some of the properties that Siebel CRM commonly uses with a command.

Property	Description
Business Service	Specifies the business service that handles the method. The service is browser or server depending on the Target property. Note the following: • If the Business Service property is empty, then Siebel CRM targets the browser or server infrastructure rather than a specific service.

Property	Description
	• If the Business Service property is not empty, then the business service that is defined must handle CanInvokeMethod and InvokeMethod for the method that is defined in the Method property.
HTML Bitmap	Specifies that the Command object uses a bitmap.
HTML Popup Dimensions	If the Show Popup property contains a check mark, then the HTML Popup Dimensions property specifies the dimensions of the pop-up window, in pixels. For example, you can specify a popup with a dimension of 640x480. Do include the *x* between the dimensions. Do not include spaces.
Method	Specifies the name of the method that Siebel CRM calls if the user clicks the menu item or clicks the toolbar icon. This is a required property. For more information, see *About the Method, Business Service, and Target Properties of the Command Object*.
Method Argument	Allows you to send an argument to the method defined in the Method property. For example, assume a command item opens a new window and navigates to a URL in that window. This command can use the Method Argument property to define the GotoURL method in the Method property and the URL to navigate to.
Show Popup	Note the following: • If the Show Popup property contains a check mark, then Siebel CRM opens a new browser window before it calls the method. • If the Show Popup property does not contain a check mark, then Siebel CRM does not open a new browser window before it calls the method.
Target	Specifies the entity that handles the method that the command calls. The following options are available: • **Browser.** Specifies the method handler as a JavaScript service on the browser or the JavaScript application, depending on if a service is defined or not defined in the Business Service property. • **Server.** Specifies the method handler and object manager service on the Siebel Server or the object manager infrastructure, depending on if a service is defined or not defined in the Business Service property. The object manager infrastructure is the Siebel Web Engine UDF loader or the model. • **Browser Applet.** For more information, see *About the Method, Business Service, and Target Properties of the Command Object*.
Tooltip Text	The tooltip text that Siebel CRM displays if the user positions the cursor over a toolbar icon. For a predefined method, leave the Tooltip Text property empty. If the Tooltip Text property is empty, then the method dynamically supplies the text, and language localization takes place as a part of this process. If you define the method, then you must enter literal text. If you define literal text, then Siebel CRM does not use language localization for this tooltip text. For more information, see *Localizing Siebel CRM*.

Properties of a Toolbar

The following table describes some of the properties that Siebel CRM commonly uses with a toolbar.

Property	Description
Class	Note the following: • For an HTML toolbar, leave the Class property empty. • For a Java toolbar, enter the name of the Java class that implements the toolbar.
Display Name	Siebel CRM uses this property for the History button and to display or hide toolbars by name.
Name	Referenced by other object definitions and by the tag in the name clause.

Properties of a Menu Item

The following table describes some of the properties that Siebel CRM commonly uses with a menu item.

Property	Description
Caption	The text that Siebel CRM displays in the menu or menu item.
Command	Name of the Command object definition that provides the method and target for the menu item.
Name	Uniquely identifies the menu or menu item.
Position	Determines the order of menu items and the parent-child relationships. For example, a menu item with a Position of 150 is a parent to menu items with Position values of 150.10, 150.20, and so on. Note the following: • If the parent menu item is active, then Siebel CRM displays the child menu items. • If the parent menu item is not active, then Siebel CRM does not display the child menu items.

Properties of a Toolbar Item

The following table describes some of the properties that Siebel CRM commonly uses with a toolbar item.

Property	Description
Command	Name of the Command object definition that provides the bitmap, method, and target for the toolbar item. To configure Siebel CRM to insert a separator between icons, you can define one or more hyphens instead of the name of a Command object.
HTML Type	Identifies the type of control that Siebel CRM displays in the toolbar in the browser. You can specify one of the following types: • ComboBox • Button • Edit • Label

Property	Description
	• Hyperlink • MiniButton • Timer
Name	Name of the toolbar item. Siebel Tools uses this property for internal use only. The Name property must be unique in the toolbar.
Position	Determines the order of toolbar items and the parent-child relationships. For example, a Toolbar item with a Position of 150 is a parent to Toolbar items with Position values of 150.10, 150.20, and so on. Note the following: • If the parent toolbar item is active, then Siebel CRM displays the child toolbar items. • If the parent toolbar item is not active, then Siebel CRM does not display the child toolbar items.

Properties of an Applet Method Menu Item

The following table describes some of the properties that Siebel CRM commonly uses with an applet method menu item.

Property	Description
Command	Name of the command that provides the bitmap, method, and target for the applet method menu item. Use Browser as the target of the Command object for a menu item that Siebel CRM displays on an applet menu.
Menu Text	The text that Siebel CRM displays in the menu item.
Position	The sequence of the menu item in the list of menu items.
Suppress Menu Item	If the Suppress Menu Item property contains a check mark, then Siebel CRM removes the class-level menu item of the defined name from the applet menu in the applet where this property is defined.

Properties of a Class Method Menu Item

The following table describes some of the properties that Siebel CRM commonly uses with a class method menu item.

Property	Description
Business Service	Note the following: • If defined, then this property identifies the business service that contains the method that Siebel CRM calls. • If not defined, then Siebel CRM calls the method in the applet class on the browser or server, as defined in the Target property. If not handled, then Siebel CRM does a subsequent retargeting.
Menu Text	The text that Siebel CRM displays in the menu item.

Property	Description
Method	The method that Siebel CRM calls if the user clicks the item.
Position	The sequence of the menu item in the list of menu items.
Suppress Menu Item	If the Suppress Menu Item property contains a check mark, then Siebel CRM removes the applet menu items of the defined name from the applet menu in all applets that it gets from this class and subclasses.
Target	Specifies the entity that handles the method that is defined in the Method property. The following options are available: • **Browser.** Specifies that the method handler is a JavaScript service on the browser, or the JavaScript applet class on the browser, depending on if a service is defined or not defined in the Business Service property. The JavaScript applet class is the JavaScript business component class. • **Server.** Specifies that the method handler is an object manager service on the Siebel Server or the applet and business component and their superclasses, depending on if a service is defined or not defined in the Business Service property.

Types of Applet Controls and List Columns

This topic describes some of the applet controls and list columns that Siebel CRM commonly uses. For more information, see *Siebel Object Types Reference* .

The following table describes some of the applet controls and list columns that Siebel CRM commonly uses, as defined in the HTML Type property of the control or list column.

HTML Type Property	Description
Button	Displays in an applet as a button with three dimensions. It starts an action if clicked. You rarely use the button control or list column. You more commonly use the minibutton. For more information, see *MiniButton Control and MiniButton List Column* and *Calling a Method from a Button in an Applet*.
CheckBox	Represents a TRUE or FALSE situation: • If the user clicks an empty check box, then Siebel CRM displays a check mark in the check box. • If the user clicks a check box that is checked, then Siebel CRM removes the check mark. To view an example of a check box control, open a Siebel application, such as Call Center. Navigate to the Service Request list, drilldown on the SR # column, and then click the More Info tab. In the Categorization section, locate the Alert Me check box.
ComboBox	Defines one of the following special-purpose lists: • In a chart applet, it defines the Show and By combo boxes.

HTML Type Property	Description
	• In a calendar applet, it defines the user name combo box. • In a pick applet, it defines the Find combo box. The ComboBox control includes a field with a button that displays with a down arrow. If the user clicks this button, then Siebel CRM displays a list. If the user chooses an item from this list, then Siebel CRM replaces the previous value in the box. A list provides the list of values. A combo box displays and behaves almost identically to a static list, except the user interacts with a combo box control or list column rather than a text control or list column. For more information, see *About Static Lists*.
DrillDownTitle	Includes a drilldown on the title of an applet. If the user clicks the title, then Siebel CRM displays the target view. Applets on the home page use this control. To view an example of a DrillDownTitle control in a Siebel application, such as Call Center, navigate to the Home screen, and then note the My Activities title.
Field	Displays text in a rectangular box. Includes a native HTML type of Text. For more information, see *Text Control and Text List Column*.
FieldLabel	Displays a field label for a list applet.
File	Allows the user to attach a file.
Hidden	Not visible in the Web page but you can access it through a script.
FormSection	A label that helps to group related fields in an applet. The FormSection label expands to fit the region where it is placed. To set it apart, Siebel CRM displays the label against the FormSection color that is defined in the cascading style sheet. To view an example of a check box control in a Siebel application, such as Call Center, you can navigate to the Service Request list, drilldown on the SR # column, and then click the More Info tab. Siebel CRM creates the Categorization section through a FormSection control. If a form applet does not use a grid, then the FormSection control cannot expand to fit in the layout editor. The FormSection control does display correctly in the Siebel client.
ImageButton	A minibutton that references an image. For more information, see *MiniButton Control and MiniButton List Column*.
InkData	For tablet PC computers. Do not use in a list applet.
JavaApplet	Siebel CRM does not support specialized applet classes for a custom configuration. For more information, see *Caution About Using Specialized Classes*.
Label	Provides a visual aid. It is not tied to a business component field, it does not display data, and it does not provide any data entry capability. If you must place a text label in a form applet, then use a label control. If a caption includes any HTML reserved character, then you must encode the HTML with &, <, >, ", and so on. Example HTML reserved characters include the ampersand (&), less than symbol (<), greater than symbol (>), period (.), and so on.

HTML Type Property	Description
Link	Creates an HTML link that calls a method when activated. Siebel CRM uses it with a control or list column where an InvokeMethod is defined, which can include a method that comes predefined with Siebel CRM. To view an example of a link control, open a Siebel application, such as Call Center. Navigate to the Home screen, and then note the My Activities link that calls the GoToView method.
Mailto	Siebel CRM displays the value of this control or list column as a link. If the user clicks this control or list column, then Siebel CRM opens the default email program for the user, and then enters the value of this control or list column into the email address.
Password	Allows the user to enter a password. Siebel CRM uses asterisks (*) to mask characters that the user enters in this control or list column.
PositionOnRow	Displays the currently chosen record in a list.
RTC	The following RTC (Rich Text Component) controls define the dimensions and font qualities of a container that includes a rich text component: • **RTCEmbedded.** An embedded text editor. Siebel CRM supports the following HTML tags: o Bold < STRONG> o italic <EM> o Underline <U> o Ordered list <OL> o Unordered list <UL> o List items <LI> o <P> o <FONT> o <BLOCKQUOTE > • **RTCEmbeddedLinkField.** Displays graphics and links in the RTCEmbedded object. • **RTCPopup.** Displays graphics and links in a popup object. • **RTC_IO.** Display graphics and links in an IO object. You cannot use an RTC control in a list applet, list, or a multi-value group applet.
RadioButton	Displays a radio button. **None** is also the default value when the user doesn't select any radio button. As a result, the Object Manager can't distinguish between **None** being explicitly selected and **None** being returned as the default. To ensure that **None** is treated as an explicit user selection and saved to the database, add the following Control User Property to the radio button whose value is **None**: `Name: SHOWNONE` `Value: TRUE`

HTML Type Property	Description
RecNavNxt	Displays the next set of records.
RecNavPrv	Displays the previous set of records.
SSNxt	Displays the next question in a SmartScript.
SSPrv	Displays the previous question in a SmartScript.
TextArea	Allows the user to enter text in multiple lines. The HTML Height property of the control or list column determines the number of rows of text Siebel CRM displays in the text area. To view an example of a TextArea control in a Siebel application, open a Siebel application, such as Call Center. Navigate to the Opportunities list, and then note the Sales Objective TextArea control in the opportunity form. For more information, see *Text Control and Text List Column*.
URL	Displays as a link. If the user clicks this control, then Siebel CRM opens the specified URL.

Objects You Use with Enterprise Integration Manager

This topic describes properties of objects that Siebel CRM uses with EIM. For more information, see *Siebel Enterprise Integration Manager Administration Guide* .

Properties of the EIM Interface Table

The following table describes properties of the EIM interface table.

Property	Value
Target Table	Chosen by the developer
EIM Delete Proc Column	T_DELETED_ROW_ID
EIM Export Proc Column	T_EXPORTED_ROW_ID
EIM Merge Proc Column	T_MERGED_ROW_ID

System Columns of the EIM Interface Table

The following information lists system columns of the EIM interface table.

Name	Physical Type	Length	Type	EIM Processing Column
CONFLICT_ID	Varchar	15	System	FALSE
CREATED	Date Time	7	System	FALSE
CREATED_BY	Varchar	15	System	FALSE
IF_ROW_BATCH_NUM	Number	22	System	FALSE
IF_ROW_MERGE_ID	Varchar	15	System	FALSE
IF_ROW_STAT	Varchar	30	System	FALSE
IF_ROW_STAT_NUM	Number	22	System	FALSE
LAST_UPD	Date Time	7	System	FALSE
LAST_UPD_BY	Varchar	15	System	FALSE
MODIFICATION_NUM	Number	22	System	FALSE
ROW_ID	Varchar	15	System	FALSE

EIM Interface Table Columns to Facilitate EIM Processing

The following table describes the generic EIM interface table columns to facilitate EIM processing for each EIM table interface. You cannot modify the values of these columns.

Name	Physical Type	Length	Type	User Name	EIM Processing Column
T_DELETED_ROW_ID	Varchar	15	Data (Private)	Deleted ROW_ID from base table	TRUE
T_EXPORTED_ROW_ID	Varchar	15	Data (Private)	Exported ROW_ID from target table	TRUE

Name	Physical Type	Length	Type	User Name	EIM Processing Column
T_MERGED_ROW_ID	Varchar	15	Data (Private)	Merged into ROW_ID from target table	TRUE

EIM Interface Table Columns for Processing a Mapping to a Defined Table

The following table describes EIM interface table columns for processing a mapping to a defined table.

Column	Value
Name	Derived from the name of the target table using the following format: T_PART ONE_PROCESS SUFFIX where: • PART ONE is the EIM Table Mapping Name without the CX_ prefix.
Physical Type	Depends on the process that Siebel CRM uses with the column.
Length	
Type	
User Name	Name of the EIM Table Mapping object that Siebel CRM creates for the column.
EIM Processing Column	TRUE

The following table describes an example of the columns that Siebel Tools creates and the default properties. For example, if the target table is CX_SEC_LEV, then Siebel Tools creates an EIM table mapping.

Name	Physical Type	Length	Type	User Name	EIM Processing Column
T_SEC_LEV_EXS	Character	1	IFMGR: Exists	CX_SEC_LEV	TRUE
T_SEC_LEV_RID	Varchar	15	IFMGR: ROW_ID	CX_SEC_LEV	TRUE
T_SEC_LEV_STA	Number	22	IFMGR: Status	CX_SEC_LEV	TRUE

Name	Physical Type	Length	Type	User Name	EIM Processing Column
T_SEC_LEV_UNQ	Character	1	IFMGR: Unique	CX_SEC_LEV	TRUE

EIM Interface Table Columns for Processing a Foreign Key

The following table describes the columns that EIM creates for each foreign key on the target EIM table mapping.

Column	Value
Name	Derived from the target table name and the corresponding foreign key column on the target table using the following format: PART ONE FOREIGN KEY COLUMN OF THE TARGET TABLE where: • PART ONE is the target table name with the CX_ prefix replaced with T_.
Type	Set to IFMGR: Fkey
Physical Type	Physical type of foreign key column of the target table, which is typically Varchar.
Length	Length of foreign key column of the target table, which is typically 15.
User Name	Derived using the following format: TARGET TABLE NAME or *EIM TABLE MAPPING NAME*.FOREIGN KEY COLUMN NAME

The following table describes the EIM table columns that EIM creates if the CX_SEC_LEV table contains the following foreign key column mappings:

- The OPTY_ID foreign key column mapping references the S_OPTY table
- The ACCNT_ID foreign key column mapping references the S_ORG_EXT table

Name	Physical Type	Length	Type	User Name
T_SEC_LEV_OPTY_ID	Varchar	15	IFMGR: Fkey	CX_SEC_LEV.OPTY_ID
T_SEC_LEV_ACCNT_ID	Varchar	15	IFMGR: Fkey	CX_SEC_LEV.ACCNT_ID

EIM Interface Table Columns for Foreign Keys

The following table describes the properties of EIM interface table columns for foreign keys. EIM creates a separate foreign key column for each U1 user key column on the foreign key tables.

Column	Value
Name	Derived using the following format: PART ONE_NAME OF THE FOREIGN KEY COLUMN IN THE TARGET TABLE where: • PART ONE is the first four letters of the foreign key table name without the S_ prefix, and trimmed to remove any trailing underscore (_) characters.
Physical Type	Physical type of the user key column on the target table, which is typically Varchar.
Length	Corresponds to the length of user key columns that the column references, which is typically 15.
Type	Data (Public)

The following table describes properties of interface table columns for foreign keys using CX_SEC_LEV as an example. EIM creates the corresponding EIM columns depending on the base column type.

Name	Physical Type	Type
OPTY_BU_ID	Varchar	Data (Public)
OPTY_NAME	Varchar	Data (Public)
OPTY_PR_DEPT_OU_ID	Varchar	Data (Public)
ORG_BU_ID	Varchar	Data (Public)
ORG_NAME	Varchar	Data (Public)
ORG_LOC	Varchar	Data (Public)

EIM Interface Table Columns for Attributes on the Target Table

The following table describes properties of EIM interface table columns for attribute columns on the target table. The EIM interface table column includes the following qualities:

- Physical Type property is Data (Public)
- Foreign Key Table property is empty

Column	Value
Name	Derived using the following format: PREFIX_NAME OF THE CORRESPONDING COLUMN IN THE TARGET TABLE CON and ACCNT are example prefixes.
Physical Type	Data (Public)
Length	Length of corresponding column in the target table.
User Name	Name of corresponding column in the target table.

The following table describes the interface table columns that EIM creates if you enter a prefix of SECL with the following attribute columns in the CX_SEC_LEV table:

- NAME (Varchar 100)
- DESC_TEXT (Varchar 250)
- AUTO_UPDATE (Char 1)

Name	Physical Type	Length	Type	User Name
SECL_NAME	Varchar	100	Data (Public)	Security Level Name
SECL_DESC_TEXT	Varchar	250	Data (Public)	Security Level Description
SECL_AUTO_UPDATE	Char	1	Data (Public)	Auto Update Flag

EIM Table Mappings That Reference the Target Table

The following table describes examples of how EIM sets the name and destination columns to the name of the target table. The column properties correspond to the values that EIM creates.

Column	Value
Name	CX_SEC_LEV
Destination Table	CS_SEC_LEV
EIM Exists Proc Column	T_SEC_LEV_EXS
EIM Row Id Proc Column	T_SEC_LEV_RID
EIM Status Proc Column	T_SEC_LEV_STA
EIM Unique Proc Column	T_SEC_LEV_UNQ

Attribute Mapping Properties of EIM Interface Columns That EIM Creates

The following table describes properties of the attribute mappings for each EIM interface column that EIM creates.

Property	Value
Name	Attribute column on the target table.
Interface Table Data Column	Name of corresponding EIM interface table column created. For more information, see *EIM Interface Table Columns for Attributes on the Target Table*.
Base Table Attribute Column	Name of the attribute column on the target table.

Foreign Key Mapping Properties of Foreign Key Columns on the Target Table

The following table describes the properties of each foreign key mapping that EIM creates for each foreign key mapping column on the target table.

Property	Value
Name	Name of the user key column.
Foreign Key Column	Name of the user key column.

Property	Value
User Key	Name of the U1 user key of the foreign key table.
EIM Foreign Key Proc Column	Corresponding EIM interface table column for foreign key processing derived from the following format: T_PART ONE_NAME OF THE USER KEY COLUMN where: • PART ONE is the name of the target table without the CX_ prefix.

The following table describes the foreign key mapping, using the CX_SEC_LEV table as an example.

Name	Foreign Key Column	User Key	EIM Foreign Key Proc Column
OPTY_ID	OPTY_ID	S_OPTY_U1	T_SEC_LEV_OPTY_ID
ACCNT_ID	ACCNT_ID	S_ORG_EXT_U1	T_SEC_LEV_ACCNT_ID

Foreign Key Mapping Columns for Foreign Key Mappings

The following table describes the properties that EIM sets for each foreign key mapping column. EIM creates a separate foreign key mapping column for each user key column in the user key that is defined for the parent foreign key mapping object in the first table in *Foreign Key Mapping Properties of Foreign Key Columns on the Target Table*.

Column	Value
Name	Name of the foreign key mapping column.
Interface Data Column	EIM interface table column that EIM maps to the user key column on the target table. EIM creates this EIM interface table column according to the specifications in the table in *EIM Interface Table Columns for Foreign Keys*.
User Key Attribute	Name of the user key column that is part of the user key defined in the first table in *Foreign Key Mapping Properties of Foreign Key Columns on the Target Table*.

The following table describes the foreign key mapping, using the CX_SEC_LEV table as an example.

Name	Interface Data Column	User Key Attribute
OPTY_BU_ID	OPTY_BU_ID	BU_ID

Name	Interface Data Column	User Key Attribute
OPTY_NAME	OPTY_NAME	NAME
OPTY_PR_DEPT_OU_ID	OPTY_PR_DEPT_OU_ID	PR_DEPT_OU_ID
ORG_BU_ID	ORG_BU_ID	BU_ID
ORG_NAME	ORG_NAME	NAME
ORG_LOC	ORG_LOC	LOC

Types of Tables and Columns That CIAI Query Supports

The following information lists the types of tables that CIAI query supports.

Table Type	Supported for a CIAI Query
Data (Intersection)	Yes
Data (Private)	Yes
Data (Public)	Yes
Database View	No
Dictionary	No
Extension	Yes
Extension (Siebel)	Yes
External	No
Interface	No
Log	No
Repository	No

The following table lists the types of columns that CIAI query supports.

Column Type	Supported for a CIAI Query
Data (Private)	Yes
Data (Public)	Yes
Denormalized	Yes
Extension	Yes
External	No
IFMGR: Exists	No
IFMGR: FKey	No
IFMGR: Status	No
IFMGR: ROW_ID	No
System	No

The following table lists the physical types of columns that CIAI query supports.

Physical Type		
DB Value	**Type**	**Supported for a CIAI Query**
C	Char	Yes
D	Date	No
X	Long	No (Obsolete data type)
L	CLOB	Yes
N	Number	No
S	Time Stamp	No
T	Date Time	No

Physical Type		
DB Value	**Type**	**Supported for a CIAI Query**
U	UTC Date Time	No
V	Varchar	Yes
X	Text	Yes

Extensive Code Examples That This Book Uses

This topic includes extensive code examples that this book uses.

Script for the Query Method for Configuring the Hierarchical List Applet

The code in this topic creates test data for the query result. To get data from an external system, the query script you use must contain code that is specific to your requirements. To get the data, you must define your own custom algorithm. For more information about using a virtual business component, see *Integration Platform Technologies: Siebel Enterprise Application Integration* . For more information about this example, see *Configuring a Hierarchical List Applet to Use External Data*. For the example in this book, use the following script for the Query of the Hierarchical List Service business service:

```
function Query( Inputs, Outputs )
{
 var row;

 row = TheApplication().NewPropertySet();

 row.SetProperty("Id", "1");
 row.SetProperty("Description", "Haus");
 row.SetProperty("Amount", "740000");
 row.SetProperty ("Parent Row Id", "");
 row.SetProperty("Has Children", "Y");
 row.SetProperty("Is Expanded", "Y");
 row.SetProperty("Outline Number", "1");
 row.SetProperty("Last Child Info", "1");
 Outputs.AddChild( row );

 row = TheApplication().NewPropertySet();
 row.SetProperty("Id", "1.1");
 row.SetProperty("Description", "T1");
 row.SetProperty("Amount", "240000");
 row.SetProperty("Parent Row Id", "1");
 row.SetProperty("Has Children", "Y");
 row.SetProperty("Is Expanded", "Y");
 row.SetProperty("Outline Number", "1.1");
 row.SetProperty("Last Child Info", "10");
 Outputs.AddChild( row );

 row = TheApplication().NewPropertySet();
 row.SetProperty("Id", "1.1.1");
```

```
row.SetProperty("Description", "Kurant");
row.SetProperty("Amount", "200000");
row.SetProperty("Parent Row Id", "1.1");
row.SetProperty("Has Children", "N");
row.SetProperty("Is Expanded", "N");
row.SetProperty("Outline Number", "1.1.1");
row.SetProperty("Last Child Info", "100");
Outputs.AddChild( row );

row = TheApplication().NewPropertySet();
row.SetProperty("Id", "1.1.2");
row.SetProperty("Description", "Blanko");
row.SetProperty("Amount", "40000");
row.SetProperty("Parent Row Id", "1.1");
row.SetProperty("Has Children", "N");
row.SetProperty("Is Expanded", "N");
row.SetProperty("Outline Number", "1.1.2");
row.SetProperty("Last Child Info", "101");
Outputs.AddChild( row );

row = TheApplication().NewPropertySet();
row.SetProperty("Id", "1.2");
row.SetProperty("Description", "T3");
row.SetProperty("Amount", "500000");
row.SetProperty("Parent Row Id", "1");
row.SetProperty("Has Children", "Y");
row.SetProperty("Is Expanded", "N");
row.SetProperty("Outline Number", "1.2");
row.SetProperty("Last Child Info", "11");
Outputs.AddChild( row );

row = TheApplication().NewPropertySet();
row.SetProperty("Id", "1.2.1");
row.SetProperty("Description", "Kurant");
row.SetProperty("Amount", "500000");
row.SetProperty("Parent Row Id", "1.2");
row.SetProperty("Has Children", "N");
row.SetProperty("Is Expanded", "N");
row.SetProperty("Outline Number", "1.2.1");
row.SetProperty("Last Child Info", "111");
Outputs.AddChild( row );

row = TheApplication().NewPropertySet();
row.SetProperty("Id", "2");
row.SetProperty("Description", "Auto");
row.SetProperty("Amount", "13500");
row.SetProperty ("Parent Row Id", "");
row.SetProperty("Has Children", "Y");
row.SetProperty("Is Expanded", "N");
row.SetProperty("Outline Number", "2");
row.SetProperty("Last Child Info", "1");
Outputs.AddChild( row );

row = TheApplication().NewPropertySet();
row.SetProperty("Id", "2.1");
row.SetProperty("Description", "T4");
row.SetProperty("Amount", "240000");
row.SetProperty("Parent Row Id", "2");
row.SetProperty("Has Children", "Y");
row.SetProperty("Is Expanded", "N");
row.SetProperty("Outline Number", "2.1");
row.SetProperty("Last Child Info", "10");
Outputs.AddChild( row );

row = TheApplication().NewPropertySet();
row.SetProperty("Id", "2.1.1");
```

```
row.SetProperty("Description", "Blanko");
row.SetProperty("Amount", "40000");
row.SetProperty("Parent Row Id", "2.1");
row.SetProperty("Has Children", "N");
row.SetProperty("Is Expanded", "N");
row.SetProperty("Outline Number", "2.1.1");
row.SetProperty("Last Child Info", "101");
Outputs.AddChild( row );

row = TheApplication().NewPropertySet();
row.SetProperty("Id", "3");
row.SetProperty("Description", "Nieren");
row.SetProperty("Amount", "8000");
row.SetProperty ("Parent Row Id", "");
row.SetProperty("Has Children", "Y");
row.SetProperty("Is Expanded", "N");
row.SetProperty("Outline Number", "3");
row.SetProperty("Last Child Info", "1");
Outputs.AddChild( row );

row = TheApplication().NewPropertySet();
row.SetProperty("Id", "3.1");
row.SetProperty("Description", "T1");
row.SetProperty("Amount", "8000");
row.SetProperty("Parent Row Id", "3");
row.SetProperty("Has Children", "N");
row.SetProperty("Is Expanded", "N");
row.SetProperty("Outline Number", "3.1");
row.SetProperty("Last Child Info", "11");
Outputs.AddChild( row );

row = TheApplication().NewPropertySet();
row.SetProperty("Id", "3.1.1");
row.SetProperty("Description", "Kurant");
row.SetProperty("Amount", "8000");
row.SetProperty("Parent Row Id", "3.1");
row.SetProperty("Has Children", "");
row.SetProperty("Is Expanded", "N");
row.SetProperty("Outline Number", "");
row.SetProperty("Last Child Info", "");
Outputs.AddChild( row );

return( CancelOperation );
}
```