



Configuring Siebel Business Applications

Siebel Innovation Pack 2016

May 2016

ORACLE®

Copyright © 2005, 2016 Oracle and/or its affiliates. All rights reserved.

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

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

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

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

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

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

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

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

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

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

Contents

Chapter 1: What's New in This Release

Chapter 2: Architecture of Siebel Business Applications

About Siebel Objects 21

About the Siebel Object Architecture 23

Overview of the Physical User Interface Layer 25

Overview of the Logical User Interface Object Layer 26

Overview of the Business Object Layer 28

Overview of the Data Objects Layer 30

Hierarchy of Object Types and Relationships 31

About Classes 32

About the Siebel Operating Architecture 32

Components That the Siebel Operating Architecture Uses 33

Infrastructure That the Siebel Web Engine Uses 33

How the Siebel Web Engine Creates a Siebel Application 35

Integration with Java EE 35

About Siebel Technologies That Configure Siebel CRM Behavior 36

Chapter 3: About Tables and Columns

About Siebel Tables 41

Overview of Siebel Tables 41

Naming Format for a Siebel Table 42

How an Extension Table Stores Custom Data 43

How an Intersection Table Defines a Many-To-Many Relationship 50

About Columns and Indexes in a Siebel Table 55

How a User Key Creates a Unique Set of Values 57

How the S_Party Table Controls Access 58

Options to Configure the Data Objects Layer 59

Options to Configure Predefined Objects and Perform Advanced Configuration 59

Options to Use a Predefined One-to-One Extension Table 60

Options to Use a Predefined One-to-Many Extension Table 62

Guidelines for Configuring the Data Objects Layer 62

Chapter 4: About Business Components, Fields, Joins, and Links

About Business Components	69
Overview of Business Components	70
How a Business Component Gets Data from an External Database	72
Business Components That Hold Temporary Data for a Task UI	73
Class Property of a Business Component	74
How a Business Component Sorts Records	74
Guidelines for Creating a Business Component	76
About Business Component Fields	78
Overview of Business Component Fields	78
How a Business Component Field Identifies the Type of Data	81
How a Business Component Field Calculates a Value	81
How a Business Component Field Sequences Records	82
How Siebel CRM Defines Read-Only Behavior for a Business Component Field	85
System Fields of a Business Component	89
Guidelines for Defining the Name of a Business Component Field	90
About Joins	91
How Siebel CRM Creates a Join	92
Guidelines for Creating a Join	94
About Multi-Value Links	95
How Siebel CRM Creates a Direct Multi-Value Link	96
How Siebel CRM Creates an Indirect Multi-Value Link	99
About Links	104

Chapter 5: About Business Objects

About Business Objects	105
How Siebel CRM Creates a Business Object	108
Guidelines for Creating a Business Object	109

Chapter 6: About Applets, Controls and List Columns

About the Form Applet and List Applet	111
How Siebel CRM Creates a Form Applet	111
How Siebel CRM Creates a List Applet	112
About Applet Controls and List Columns	113
Types of Applet Controls and List Columns	114
Options to Create an Applet	115
Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data	116

- Options to Filter Data That Siebel CRM Displays in an Applet 118
- Options to Determine How Siebel CRM Displays Controls and List Columns in a Siebel Application 123

- Guidelines for Creating an Applet 125
 - Guidelines for Naming an Applet 126
 - Guidelines for Creating an Applet Title 127
- Guidelines for Creating a Control or List Column 127
 - Guidelines for Creating a Check Box 128
 - Guidelines for Creating a Text Control or List Column 128

Chapter 7: About Views, Screens, and Applications

- About the Siebel Client Navigation Model 129
- About Views 130
 - About List-Form Views 130
 - About Master-Detail Views 131
- About Screens 133
 - About Screen Views 133
 - Guidelines for Creating a View 136
- Options to Create a View or Screen 138
 - Options to Drill Down to Another View 138
 - Options to Toggle Between Applets in a View 141
- About Applications 142

Chapter 8: About Siebel Web Templates and Siebel Tags

- About Siebel Web Templates 145
 - Overview of Siebel Web Templates 145
 - How Siebel CRM References Web Pages 149
 - How Siebel CRM Uses HTML Frames in the Container Page 150
- About View Web Templates 152
- About Applet Web Templates 154
 - Overview of Applet Web Templates 155
 - About Grid Form Applet Templates 156
 - About Nongrid Form Applet Templates 157
 - About List Applet Templates 159
 - About Tree Applet Templates 162
 - About Catalog List Applets and Rich List Templates 165
- About Siebel Tags 168
 - Overview of How Siebel CRM Uses Siebel Tags 169

- About Singleton and Multipart Tags, and the This Tag 170
- About Iterator Tags 171
- About Search and Find Tags 172
- About Siebel Conditional Tags 175

Guidelines for Configuring Siebel Web Templates and Siebel Tags 178

Chapter 9: Configuring a Siebel Application

- About Configuring a Siebel Application 183
- Roadmap for Configuring a Siebel Application 184
- Developing an Implementation Plan 184
 - Developing a Configuration Strategy 185
 - Developing a Plan to Control File Versions for the Physical User Interface Layer 187
- Using Development Tools and Setting Up the Development Environment 187
 - Overview of the Development Process 188
 - Guidelines for Developing a Siebel Application 189
 - Setting Up the Configuration File for Siebel Tools 191
 - Displaying Object Types You Use to Configure Siebel CRM 192
 - Setting Up a Developer with a Local Database 193
 - Creating Scripts to Configure Siebel CRM 197

Chapter 10: Reusing Predefined Objects

- Reasons to Reuse or Not Reuse a Predefined Object 203
- Guidelines for Reusing a Predefined Object 204
 - Reasons to Reuse a Predefined Object 204
 - Guidelines for Reusing a Predefined Table 205
 - Guidelines for Reusing a Predefined Business Component 208
 - Guidelines for Reusing a Predefined Business Object 209
 - Guidelines for Reusing an Applet 209
 - Guidelines for Reusing a Predefined View 210
 - Guidelines for Reusing a Predefined User Interface Object 210
 - Reasons Not to Reuse a Predefined Object 211
- Process of Determining Whether You Can Reuse a Predefined Object 213
 - Determining Functional Fit for Reusing a Predefined Object 213
 - Determining Technical Fit for Reusing a Predefined Object 214
 - Determining Whether You Can Reuse a Predefined Table Column 215
 - Determining Whether You Can Reuse a Predefined Business Component Field 217
 - Determining Whether You Can Reuse a Predefined Business Component 219

Chapter 11: Using the Entity Relationship Designer

- About the Entity Relationship Designer 221
- Process of Creating and Binding an Entity Relationship Diagram 223
 - Creating an Entity Relationship Diagram 224
 - Binding an Entity to a Business Component 224
 - Associating an Entity Attribute with a Business Component Field 225
 - Binding a Relationship to a Link or Join 225
- Opening or Modifying an Entity Relationship Diagram 226
 - Opening an Entity Relationship Diagram 226
 - Viewing the Entities and Relations Lists of an ERD 226
 - Modifying the Properties of a Relationship 227
 - Copying the Drawing of an Entity Relationship Diagram 228
- Modifying Shapes and Lines in the Entity Relationship Designer 228
 - Modifying Shapes in the Entity Relationship Designer 228
 - Modifying Relationships in the Entity Relationship Designer 229
 - Moving Shapes in the Entity Relationship Designer 230
 - Resizing Shapes in the Entity Relationship Designer 231
 - Zooming, Displaying, and Snapping the Grid 231

Chapter 12: Configuring Tables

- Using the New Table Wizard to Create a New Table 233
- Creating a Custom Index 235
- Adding an Extension Column to a Base Table 236
- Configuring Objects to Use a One-To-Many Extension Table 237
- Configuring an Extension Table 237
- Applying a Custom Table to the Local Database 240
- Applying a Data Layer Customization to the Server Database 242
- Downloading a Data Layer Customization to Remote Users 243

Chapter 13: Configuring Business Components, Links, and Business Objects

- Configuring a Business Component 245
 - Creating a New Business Component 245
 - Determining How a Business Component Sorts Records 246
 - Determining How a Business Component Sequences Records 246
 - Defining Read-Only Behavior for a Business Component 248
 - Creating a Recursive Join on a Business Component 249

- Configuring a Business Component to Copy Child Records If the User Copies the Parent Record 250
- Allowing the User to Set a Primary Team Member 251
- Configuring a Business Component Field 251
 - Creating a New Business Component Field 252
 - Activating a Multi-Value Field 252
 - Validating Data That the User Enters in a Business Component Field 253
 - Creating a Business Component Field That Displays More Than One Currency 254
 - Configuring Client-Side Import to Update a Business Component Field 257
 - Creating a Joined Business Component Field 258
 - Creating a Predefault Value for a Joined Business Component Field 260
- Configuring a Link 261
 - Configuring a Link That Deletes Child Records If the User Deletes the Parent Record 261
 - Configuring a Link That Creates a One-to-Many Relationship 263
 - Configuring Two Links That Create a Many-to-Many Relationship 263
 - Creating an Association Between One Parent and Multiple Child Records 264
- Creating a Business Object 266

Chapter 14: Configuring Views, Screens, and Applications

- Process of Creating a View 267
 - Creating a View 267
 - Editing the Layout of a View 268
 - Registering and Associating a View with a Responsibility 269
- Configuring a View 270
 - Using the Views List to Create a View 270
 - Configuring the Thread Bar 271
 - Defining the Drilldown Sequence to Configure Search for an Account 272
 - Creating an Applet Toggle 273
 - Defining Whether a View Is Cachable 276
 - Controlling How the User Can Modify View Layouts 276
 - Creating a Secure View 278
 - Creating a View That Requires an Explicit User Login 278
 - Restricting Access to Records in a View 279
 - Troubleshooting a View That Siebel CRM Does Not Display in the Siebel Client 279
- Process of Creating a Screen 281
 - Creating a Screen 281
 - Creating a Page Tab 282
 - Creating a Screen Menu Item 282
 - Creating a Screen View 283
 - Defining the Sequence That Siebel CRM Uses to Display Screen Views 283

Process of Creating a Screen Home Page View	285
Defining Business Components for the Screen Home Page View	285
Creating Links to Frequently Accessed Data	288
Determining How Siebel CRM Displays Recent Records	289
Defining the Business Object for the Screen Home Page View	291
Creating Simplified Screen Home Page Applets	292
Creating a Screen Home Page View	294
Adding the Screen View to the Screen	296
Creating and Deploying an Application	297
Creating a New Application	297
Configuring the Sort Order for Siebel CRM	298
Configuring Keyboard Shortcuts for an Application or Applet	300

Chapter 15: Configuring Applet Layouts

Process of Using the Applet Layout Editor	303
Setting the Language Mode of the Applet Layout Editor	303
Setting the Configuration Context	304
Defining the Applet Mode	305
Adding a Control or List Column to an Applet Layout	305
Previewing the Applet Layout	307
Exporting an Applet Preview to an HTML File	308
Options for Configuring an Applet Layout	308
Configuring the Display Name for a Control Caption or List Column	309
Displaying a Parent Applet Field in the Title of a Detail Applet	309
Displaying a Subset of Fields or CRM Records	310
Displaying a Field Only If the User Chooses Show More	311
Setting the Tab Order for Fields in an Applet	312
Setting the Input Method Editor Mode on a Control or List Column	312
Copying Controls and Labels from an Applet to a Web Template	313
Verifying the Map Between a Control or List Column and a Placeholder	314
Using Grid Layout for an Applet	315
Accessing Grid Layout Web Templates	316
Using the Conversion Wizard to Convert a Form Applet to Grid Layout	316
Modifying the Web Template to Convert a Form Applet to Grid Layout	318
Identifying an Applet or Applet Web Template That You Cannot Convert to a Grid Layout	319
Modifying the Background Color of an Applet	319
Troubleshooting a Grid Layout Conversion Problem	320
Guidelines for Working with Grid Layout	321

Chapter 16: Configuring Applets

- Creating an Applet 323
 - Creating a List Applet 323
 - Creating a Form Applet 325
- Configuring Pop-Up Applets and Windows 327
 - Guidelines for Creating a Pop-Up Applet or Window 327
 - Creating a Pop-Up Control in an Applet 328
 - Creating a Pop-Up Applet That Siebel CRM Opens from an Applet 328
 - Creating a Pop-Up Applet That Siebel CRM Opens from a Menu Item 330
 - Creating a Pop-Up Wizard 331
- Configuring Applet Buttons, Controls, and List Columns 333
 - Configuring a Spell Check Button on an Applet 334
 - Calling a Method from a Button in an Applet 336
 - Identifying the Controls and List Columns That Siebel CRM Displays in the Siebel Client 337
 - Modifying the Text Style of a Control or List Column in an Applet 338
 - Displaying Totals for a List Column in an Applet 338
 - Defining the Properties of a Control or List Column If HTML Type Is Text 341
 - Using a Control to Allow the User to Click a Link to Activate a Record 343
 - Displaying the Save Button 343
- Configuring How Siebel CRM Displays Data in an Applet 344
 - Controlling How the User Creates, Edits, Queries, and Deletes CRM Data 344
 - Controlling Query Behavior If the User Presses CTRL+ENTER 345
 - Filtering Data That Siebel CRM Displays in an Applet 346
 - Displaying HTML Content in an Applet 346
 - Displaying a System Field in an Applet 351
 - Avoiding Losing Context During a Drilldown 352
 - Configuring Quick Fill for a Custom Applet 352
- Process of Configuring Drilldown from the Calendar Applet 352
 - Preparing Siebel Tools to Configure Drilldowns 353
 - Defining Fields in the Business Component 353
 - Defining the Applet User Properties 356
 - Creating the Drilldown Objects and Controls 358
 - Configuring a Different Icon for the Dynamic Drilldown 360
 - Configuring a Different Destination for the Dynamic Drilldown 361

Chapter 17: Configuring Special-Purpose Applets

- Configuring a Tree Applet 363
 - Overview of Configuring a Tree Applet 363
 - Using the Tree Applet Wizard to Create a Tree Applet 367

- Configuring a Tree Node 368
- Using the Applet Layout Editor to Add a Tree Control 370
- Configuring a Recursive Tree Applet 371
- Configuring the Graphic Elements of a Tree Applet 372
- Configuring a Hierarchical List Applet 375
 - Viewing an Example of a Hierarchical List Applet 375
 - Configuring Indentation and Order of a Hierarchical List Applet 376
 - Limiting the Number of Records That Siebel CRM Returns in a Hierarchical List Applet 377
 - Configuring a Hierarchical List Applet to Use External Data 377
- Configuring a File Attachment Applet 382
 - Configuring an Attachment Business Component 384
 - Configuring an Attachment Table 386
- Configuring an Organization Analysis Applet 387

Chapter 18: Configuring Lists and Pick Applets

- About Lists and Pick Applets 389
 - About Static Lists 389
 - About Pick Applets 393
 - About Dynamic Lists 395
 - About Hierarchical Lists 405
- Configuring Lists and Pick Applets 407
 - Using the Pick List Wizard to Create a Static List 407
 - Creating a Static List Manually 408
 - Using the Pick Applet Wizard to Create a Pick Applet 411
 - Using the Pick List Wizard to Create a Dynamic List 412
 - Constraining a Dynamic List 414
 - Creating a Hierarchical List 414
- Creating a List of Values 416
- Associating an Organization with a List of Values 418
 - Guidelines for Associating an Organization with a List of Values 420
 - Guidelines for Using Script to Associate a List of Values with an Organization 420
 - Creating a Value to Display for More Than One Organization 421
 - Using the Organization Specifier Property to Display Custom Lists of Values 421

Chapter 19: Configuring Multi-Value Group, Association, and Shuttle Applets

- Creating Multi-Value Groups and Multi-Value Group Applets 423
 - About the Multi-Value Group Applet 423
 - How Siebel CRM Creates a Multi-Value Group 426

- Guidelines for Creating Multi-value Group Applets and Pick Applets 430
- Creating a Multi-Value Group 430
- Creating a Multi-Value Group Applet 432

About Association Applets 433

- Overview of Association Applets 433
- How Siebel CRM Creates an Association Applet 435
- How Siebel CRM Calls an Association Applet from a Master-Detail View 439
- Constraining an Association Applet 441

About Shuttle Applets 441

Creating a Shuttle Applet 443

- Creating an Association Applet 443
- Creating the Multi-Value Group Applet 444
- Creating the View 446

Chapter 20: Configuring Menus, Toolbars, and Icons

About Menus and Toolbars 449

- Objects That Siebel CRM Uses in a Menu or Toolbar 450
- About the Method, Business Service, and Target Properties of the Command Object 451

Configuring Menus and Toolbars 455

- Creating a Command Object 455
- Creating a New Toolbar 456
- Adding a New Toolbar Icon to a Predefined Toolbar 456
- Activating Menu Items and Toolbars 457
- Creating an Applet Menu 458
- Activating or Suppressing an Applet Menu Item 459
- Using JavaScript to Configure a Toolbar 460

Configuring Icons 461

- Overview of Configuring Icons That Siebel CRM Displays in Siebel Clients 462
- Configuring a Bitmap Category and a Bitmap 463
- Displaying an Icon on a Button 464
- Displaying an Icon as a Link 465
- Using Icons to Represent Values in a Field 465
- Configuring Icons in a Tree Applet 467

Chapter 21: Configuring Siebel Web Templates and Siebel Tags

Configuring Siebel Web Templates and Siebel Tags 471

- Editing the Layout of a Web Page 471
- Adding Graphics to a Web Template 472
- Displaying Multiple Views on a Page 473

Configuring How Siebel CRM Displays an Error That Occurs on the Siebel Server	475
Configuring Web Templates to Display Menus, Toolbars, and Thread Bars	477
Using Web Templates to Display Menus and Buttons	477
Using Web Templates to Configure Toolbars	479
Using Web Templates to Configure the Thread Bar	481
Configuring an HTML Control Type	483
Creating a New HTML Type	484
How the Siebel Web Engine Uses a Custom HTML Type	485
Configuring an HTML Type	485

Chapter 22: Improving the Performance of Siebel Business Applications

Using the Case Insensitivity Wizard to Improve Query Performance	489
How a CIAI Index Can Improve a Query	490
Overview of the Case Insensitivity Wizard	491
Variables You Can Use with the Case Insensitivity Wizard	493
Using the Case Insensitivity Wizard on a Table	496
Using the Case Insensitivity Wizard on a Table Column	498
Using the Case Insensitivity Wizard on Columns That Do Not Contain an Index	498
Using the Case Insensitivity Wizard to Do Various CIAI Configuration Tasks	500
Using the Case Insensitivity Wizard to Deactivate CIAI Configuration	501
Choosing the Correct Repository when Running the Case Insensitivity Wizard	501
Limiting the Length of Schema Object Names Manually	501
Other Configurations to Set Case Sensitivity	502
Improving the Performance of a Siebel Application	503
Preventing a Secondary Query on a Foreign Key	503
Defining the Primary ID Field of a Multi-Value Link	504
Modifying Custom Search Specifications	507
Using Declarative Configuration to Enable a Button	508
Improving Performance When Using Applet Toggles	508
Deactivating Unused Screens	508
Considering Factors That Affect Chart Performance	509
Considering Factors That Affect MLOV Performance	509

Chapter 23: Mapping a Custom Table to an Interface Table for Siebel EIM

Overview of Using Siebel EIM for Bulk Import and Export of Data	511
About Interface Tables	511
Object Types That Enterprise Integration Manager Uses	512
Mapping a Custom Table to an Interface Table	518

- Mapping a Custom Table to an Interface Table Using the EIM Table Mapping Wizard 518
- Relations That the EIM Table Mapping Wizard Creates 520
- Guidelines for Using the EIM Table Mapping Wizard 520
- Starting the EIM Table Mapping Wizard for a Table That Does Not Use the Foreign Key 521
- Deactivating Instead of Deleting an EIM Attribute Mapping 522
- Modifying Data from NULL to No Match Row Id 523

Chapter 24: Configuring Dock Objects for Siebel Remote

- About Dock Objects 525
 - Dock Object Table 526
 - Dock Object Visibility Rule 526
- Configuring Dock Objects 530
 - Reusing a Predefined Dock Object 531
 - Creating a New Dock Object 532
 - Adding a Dock Object Table to an Existing Dock Object 535
 - Verifying That Siebel Tools Created Dock Objects 536
 - Rebuilding the Databases After You Run the Docking Wizard 537
 - Cleansing Dock Objects 537
 - Creating a Table for a Dock Object 538

Chapter 25: Localizing Siebel Business Applications

- Overview of Localizing a Siebel Application 539
 - About Localization in the Development Environment 540
 - Deleting a Control or List Column While in Language Override Mode 541
 - Localizing an Application Menu 541
 - Localizing Help 541
- Localizing a Multilingual List of Values 542
 - Overview of Language-Independent Code 543
 - Configuring a Multilingual List of Values 544
 - Defining Properties of an MLOV 548
 - Adding Records for All Supported Languages 549
 - Searching a Multilingual List of Values 550
 - Searching Fields That an MLOV Controls 551
 - Deactivating an MLOV Record Instead of Deleting It 552
 - Guidelines for Localizing a Multilingual List of Values 552
- Converting Your Current Data for an MLOV 555
 - Guidelines for Converting Your Current Data for an MLOV 556
 - Parameters You Use to Run the MLOV Converter Utility 557
 - Resuming the MLOV Converter Utility If an Error Occurs 558

Using the MLOV Converter Utility to Convert Multiple Languages	559
Troubleshooting Problems with an MLOV Conversion	560
Configuring Certain Siebel Modules to Use MLOV Fields	561
Configuring Siebel Workflow to Use MLOV Fields	561
Configuring Siebel Assignment Manager to Use MLOV Fields	565
Configuring Siebel Anywhere to Use MLOV Fields	566

Chapter 26: Configuring the Customer Dashboard

Overview of the Customer Dashboard	567
Enabling the Customer Dashboard	568
Process of Configuring the Customer Dashboard	569
Adding a Business Component to the Customer Dashboard	569
Mapping a Business Component Field to a Customer Dashboard Field	571
Modifying the Appearance and Layout of the Customer Dashboard	574
Creating a Label for a Customer Dashboard Field	574
Formatting a Customer Dashboard Phone Number Field	575
Modifying the Go To List in the Customer Dashboard	577
Modifying the Background Color and Border of the Customer Dashboard	580
Adding a Custom Control to the Customer Dashboard	580
Modifying a Custom Control in the Customer Dashboard	581
Options to Update the Customer Dashboard	582
Configuring a Button to Update the Customer Dashboard	583
Configuring Communications Events to Update the Customer Dashboard	584
Configuring SmartScript to Update the Customer Dashboard	586
Using Siebel Visual Basic or Siebel eScript to Update Information in the Customer Dashboard	588
Using Personalization to Update the Customer Dashboard	591

Appendix A: Reference Materials for Configuring Siebel Business Applications

Properties of Object Types	593
Properties of a Siebel Table	594
Properties of a Table Column	594
Properties of an Index of a Siebel Table	598
Properties of an Index Column	599
Properties of a Business Component	599
Type Property of a Business Component Field	600
Display Format Property of a Control or List Column	604
Properties of a Screen View	606
Properties of an Application	607

Properties of Objects You Use with a Menu or Toolbar 608
Types of Applet Controls and List Columns 613
Objects You Use with Enterprise Integration Manager 617
Types of Tables and Columns That CIAI Query Supports 624
Extensive Code Examples That This Book Uses 626

Index

1

What's New in This Release

What's New in Configuring Siebel Business Applications, Siebel Innovation Pack 2016

Table 1 lists the changes in this revision of the documentation to support Siebel Innovation Pack 2016.

NOTE: Siebel Innovation Pack 2016 is a continuation of the Siebel 8.1/8.2 release.

Table 1. What's New in Configuring Siebel Business Applications, Siebel Innovation Pack 2016

Topic	Description
"Overview of the Logical User Interface Object Layer" on page 26 "Overview of Siebel Web Templates" on page 145	Modified topics. As of Siebel Innovation Pack 2016, Siebel Wireless and Siebel Handheld applications are no longer supported.
"Integrations That Use Siebel Partner Connect and Siebel Tools for Partner Connect"	Deleted topic. Deleted obsolete topic.
"About Standard Interactivity and High Interactivity" (removed) "Configuring the Show Caption in Standard Interactivity" (removed) "Configuring How Siebel CRM Displays the Currently Chosen Record in Standard Interactivity" (removed) Multiple topics	Removed and modified topics. As of Siebel Innovation Pack 2016, Siebel Business Applications use Siebel Open UI. The high interactivity and standard interactivity clients are no longer supported.
"About Chart Applet Templates" "Configuring a Chart Applet"	Removed topics. Removed obsolete topics, which no longer apply in Siebel Innovation Pack 2016.
"Setting Up the Development Environment" (removed) "Setting Up a Developer with a Local Database" on page 193 (modified)	Modified and removed topics. Developers must now use Oracle Database XE for the local database for Siebel Tools. SAP SQL Anywhere is no longer supported, as of Siebel Innovation Pack 2016. (Although the topic "Setting Up the Development Environment" was removed, all of its subtopics remain, under the main topic now called "Using Development Tools and Setting Up the Development Environment" on page 187.)

Table 1. What's New in Configuring Siebel Business Applications, Siebel Innovation Pack 2016

Topic	Description
"Defining Whether a View Is Cachable" on page 276	Modified and renamed topic. Removed references to high interactivity and standard interactivity and clarified the supported values for the property High Interactivity Enabled. (The name of the described property is unchanged from previous releases.)
"Controlling How the User Can Modify View Layouts" on page 276	Modified topic. Removed the obsolete subtopic "How the Layout Control Applet Works." Deleted other related content elsewhere.
"Rebuilding the Databases After You Run the Docking Wizard" on page 537	Modified topic. Removed the obsolete parameter TsDbRecreate for the Transaction Processor component.
"Displaying Different Sections of a Template Depending on the Browser Type"	Removed topic. Removed obsolete topic.
"Defining the Pop-Up Start Window"	Removed topic. Removed obsolete topic.
"Modifying the Background Color and Border of the Customer Dashboard" on page 580 (modified) "Adding a Custom Control to the Customer Dashboard" on page 580 (new) "Modifying a Custom Control in the Customer Dashboard" on page 581 (new) "Modifying the Size and Location of the Customer Dashboard" (removed)	Modified, new, and removed topics. Made updates in this chapter to reflect configuration options available for the Customer Dashboard in the current release.
"Configuring Help"	Removed chapter. In Siebel Innovation Pack 2016, the Siebel online help does not support context sensitivity. For this reason, the chapter about configuring the online help for Siebel Business Applications has been removed.

What's New in Configuring Siebel Business Applications, Siebel Innovation Pack 2015, Rev. B

Table 2 lists the changes in this revision of the documentation to support Siebel Innovation Pack 2015, Patchset 5. The new features listed in this table are available only if you install and deploy Innovation Pack 2015, Patchset 5 or later. For more information, see *Siebel Patchset Installation Guide* on My Oracle Support.

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

Table 2. What's New in Configuring Siebel Business Applications, Siebel Innovation Pack 2015, Rev. B

Topic	Description
"Setting Up a Developer with a Local Database" on page 193	Modified topic. Developers can now use Oracle Database XE for the local database for Siebel Tools, as an alternative to SAP SQL Anywhere. This option requires that you install Siebel Innovation Pack 2015, Patchset 5 or later.

What's New in Configuring Siebel Business Applications, Siebel Innovation Pack 2015, Rev. A

Table 3 lists the changes in this revision of the documentation to support this release of the software.

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

Table 3. What's New in Configuring Siebel Business Applications, Siebel Innovation Pack 2015, Rev. A

Topic	Description
"Setting Up a Developer with a Local Database" on page 193 "Downloading a Data Layer Customization to Remote Users" on page 243	Modified topics. Notes have been added to this guide regarding a change in the software used with the local database used by Siebel Tools and by Siebel Remote.

What's New in Configuring Siebel Business Applications, Siebel Innovation Pack 2015

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

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

2

Architecture of Siebel Business Applications

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

- [About Siebel Objects on page 21](#)
- [About the Siebel Object Architecture on page 23](#)
- [About the Siebel Operating Architecture on page 32](#)
- [About Siebel Technologies That Configure Siebel CRM Behavior on page 36](#)

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.

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 Business Applications 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 SRF (Siebel Repository File) 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 blue line.

Depending on the software configuration you purchase, your Siebel Business 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 on page 25](#)
- [Overview of the Logical User Interface Object Layer on page 26](#)
- [Overview of the Business Object Layer on page 28](#)
- [Overview of the Data Objects Layer on page 30](#)
- [Hierarchy of Object Types and Relationships on page 31](#)
- [About Classes on page 32](#)

Siebel CRM separates the metadata that defines objects and files, such as web templates and style sheets, into several architectural layers. Figure 1 describes this Siebel object architecture.

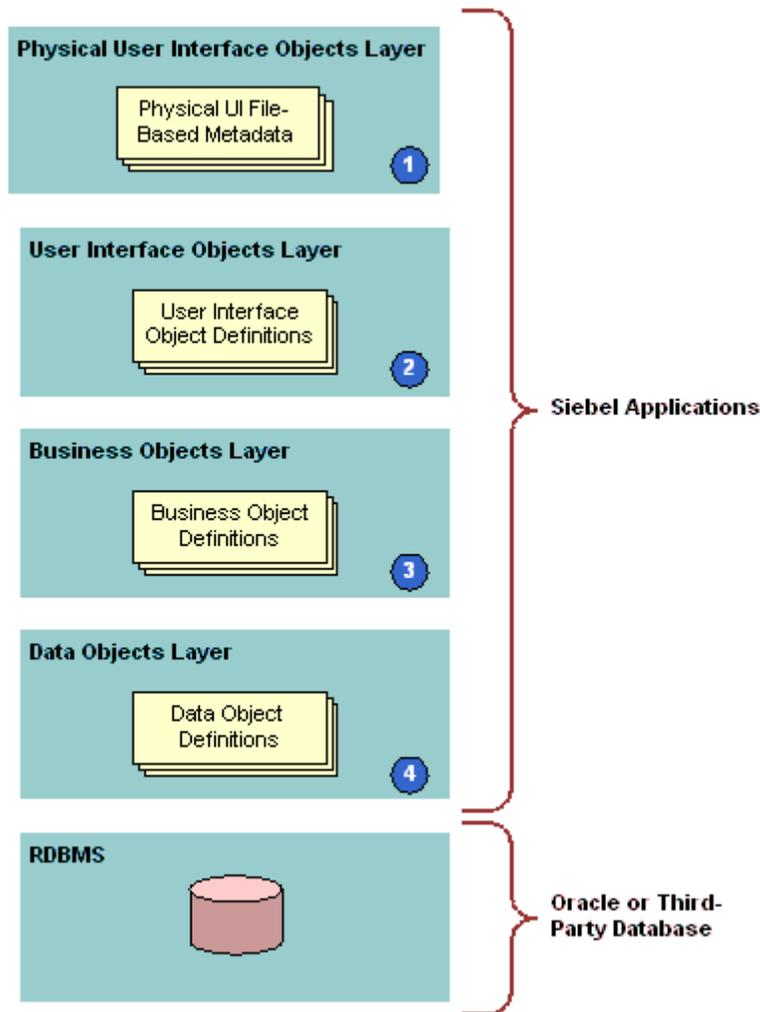


Figure 1. Siebel Object Architecture

Explanation of Callouts

The architecture includes the following items:

- 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” on page 25.](#)

- 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” on page 26.](#)
- 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” on page 28.](#)
- 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” on page 30.](#)

Siebel CRM insulates objects in each layer of the Siebel Object Architecture from the layers above and below each layer, including the RDBMS (Relational Database Management System) that resides at the bottom 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.

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” on page 187.](#)

Siebel Web Template

A *Siebel Web template* (SWT) is an HTML file 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” on page 145.](#)

Siebel Tag

A *Siebel tag* is a special tag that resides in a 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” on page 168.](#)

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” on page 477](#).

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” on page 111](#).

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” on page 130](#).

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” on page 133](#).

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 Business 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” on page 142](#).

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” on page 282](#).

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” on page 282](#).

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” on page 113](#).

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 far left 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” on page 112](#).

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” on page 323](#).

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 on page 111](#)
- [About Views on page 130](#)
- [About Siebel Web Templates on page 145](#)

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” on page 111](#).

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” on page 130](#).

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” on page 69](#).

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” on page 78](#).

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” on page 91](#).

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” on page 93](#).

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 [“About Business Objects” on page 105](#).

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 [“About Business Objects” on page 105](#).

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” on page 104](#).

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

A *multi-value group* is a set of child records that Siebel CRM associates with a parent record. A *multi-value link* is an object that allows you to create a multi-value group. A *multi-value group applet* is a dialog box that displays the records that constitute a multi-value group. If the user opens the applet from a parent record in a list or form applet, then Siebel CRM uses a multi-value link to build, and then display a multi-value group in the multi-value group applet. For more information, see the following topics:

- [Viewing an Example of a Multi-Value Group Applet on page 424](#)
- [About Multi-Value Links on page 95](#)
- [How Siebel CRM Creates a Multi-Value Group on page 426](#)

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 the *Siebel Installation Guide* for the operating system you are using.

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” on page 41](#).

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” on page 55](#).

Index

An *index* is an object that identifies a physical index file in the RDBMS. For more information, see [“Indexes of a Siebel Table” on page 57](#).

Hierarchy of Object Types and Relationships

Siebel CRM builds an object that resides in an upper layer on top 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.

Figure 2 describes the relationships that exist between the major object types that Siebel CRM uses.

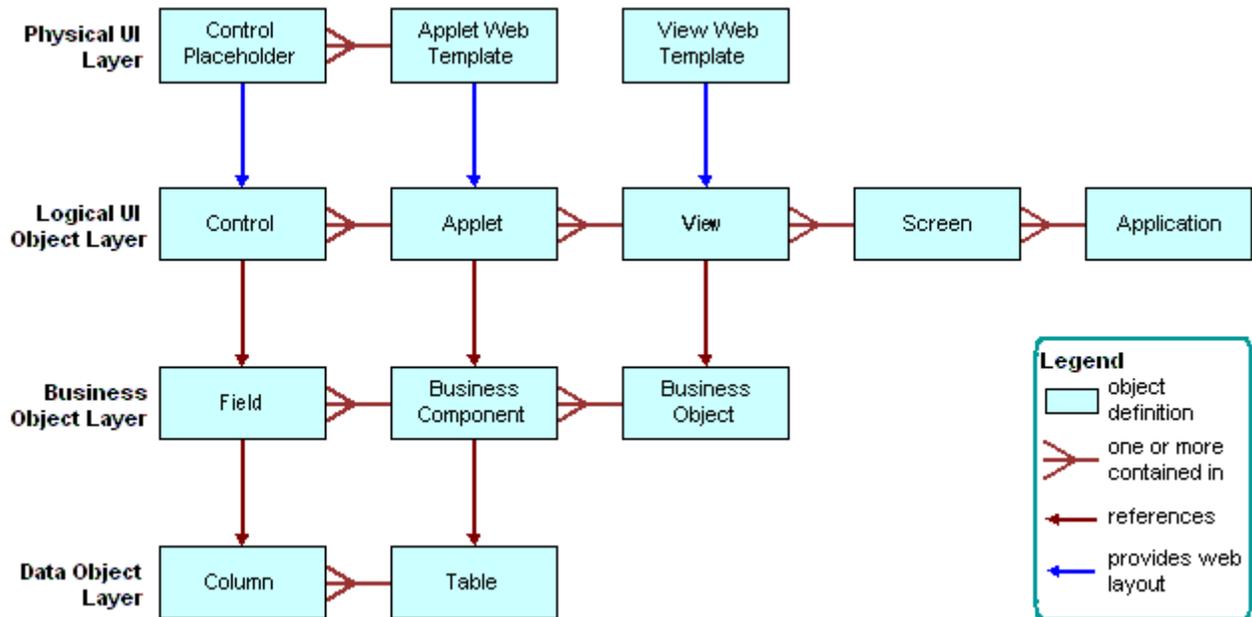


Figure 2. Relationships and the Major Object Types That Siebel CRM Uses

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 on page 33](#)
- [Infrastructure That the Siebel Web Engine Uses on page 33](#)
- [How the Siebel Web Engine Creates a Siebel Application on page 35](#)
- [Integration with Java EE on page 35](#)

Components That the Siebel Operating Architecture Uses

This topic describes some of the major components of the Siebel operating architecture.

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*.

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 Web Server. The Siebel Web Server 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* for the operating system you are using.

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 Web Server Extension (SWSE).

SWSE runs on the Web server and interfaces with the Siebel Web Engine, with most of the work occurring in the Siebel Web Engine. SWSE maintains the session and works as a communication intermediary. Network communication between SWSE 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 SWSE and the object manager. From the perspective of SWSE, 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, such as by clicking a button or link, the Siebel Web Engine does the following work:

- 1 Reads the object definitions from the SRF (Siebel Repository File) file.
- 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 file.
- 6 Displays the HTML output to the user.

Figure 3 describes the relationships between style sheets, templates, and objects in the Siebel repository, and the final HTML output.

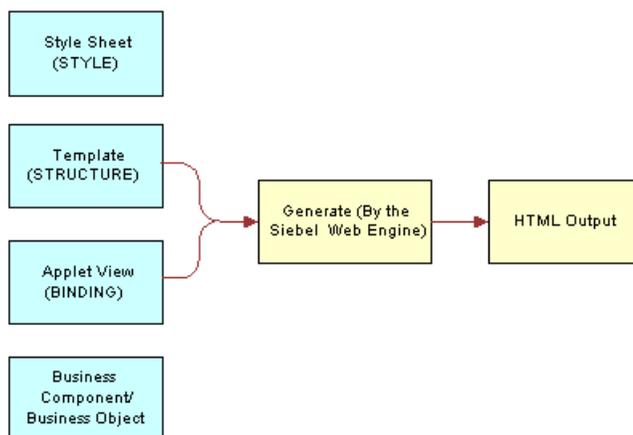


Figure 3. Relationships Between Style Sheets, Templates, and Objects in the Siebel Repository, and the Final HTML Output

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” on page 23](#).

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” on page 23](#).

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 on page 37](#)
- [Task UI on page 38](#)
- [Siebel Workflow on page 38](#)
- [Siebel SmartScript on page 38](#)

- [Siebel Assignment Manager on page 38](#)
- [State Model on page 39](#)
- [Siebel Pricer on page 39](#)

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 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

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

- [About Siebel Tables on page 41](#)
- [Options to Configure the Data Objects Layer on page 59](#)
- [Guidelines for Configuring the Data Objects Layer on page 62](#)

About Siebel Tables

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

- [Overview of Siebel Tables on page 41](#)
- [Naming Format for a Siebel Table on page 42](#)
- [How an Extension Table Stores Custom Data on page 43](#)
- [How an Intersection Table Defines a Many-To-Many Relationship on page 50](#)
- [About Columns and Indexes in a Siebel Table on page 55](#)
- [How a User Key Creates a Unique Set of Values on page 57](#)
- [How the S_Party Table Controls Access on page 58](#)

For more information, see [“Overview of the Data Objects Layer” on page 30](#).

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” on page 189](#).

Naming Format for a Siebel Table

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

PREFIX_NAME_SUFFIX

Table 4 describes the naming format.

Table 4. Parts of the Table 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.

Table 5 describes some of the prefixes that Siebel CRM commonly uses. Each prefix indicates the part of the Siebel schema that contains the table.

Table 5. Table Prefixes That Siebel CRM Commonly Uses

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.

Table 6 describes some of the suffixes that Siebel CRM commonly uses. Each suffix indicates a table type.

Table 6. Table Suffixes That Siebel CRM Commonly Uses

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.
_X	A one-to-one extension table that you can use to add custom data to the Siebel database.

Table 6. Table Suffixes That Siebel CRM Commonly Uses

Suffix	Description
_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 on page 43](#)
- [How a One-To-One Extension Table Extends Data Storage for a Single Business Component on page 44](#)
- [How an Implicit Join Creates a Relationship Between a Base Table and a Business Component on page 44](#)
- [An Explicit Join Creates a Relationship Between an Extension Table and a Business Component on page 46](#)
- [A One-To-Many Extension Table Stores Data From Multiple Business Components on page 48](#)
- [Summary of Support for Extension Tables and Extension Columns on page 48](#)

For more information, see the following topics:

- [Options to Use a Predefined One-to-One Extension Table on page 60](#)
- [Options to Use a Predefined One-to-Many Extension Table on page 62](#)
- [Manually Creating a One-to-One Extension Table on page 238](#)
- [Using the New Table Wizard to Create a New Table on page 233](#)

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.

Figure 4 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.

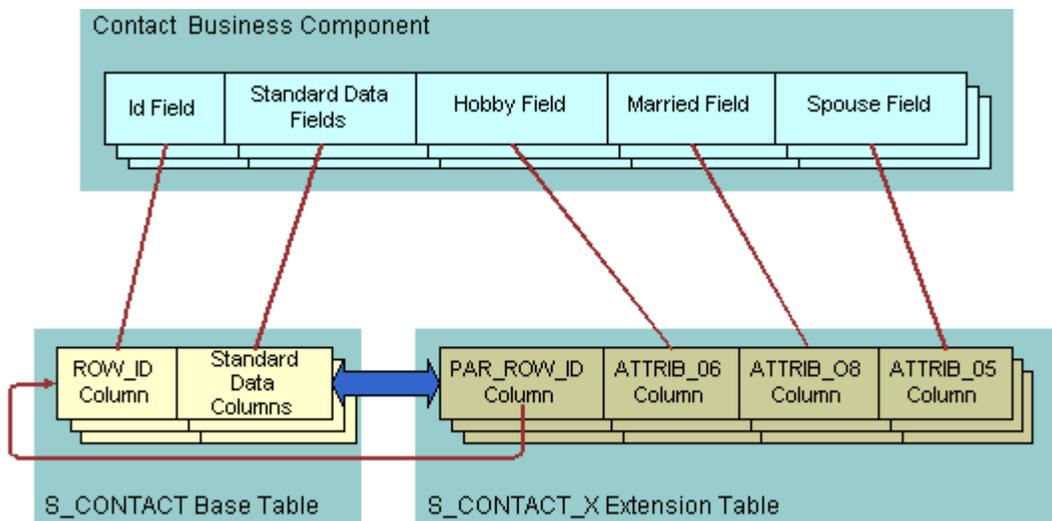


Figure 4. Example of How a One-To-One Extension Table Extends Data Storage for a Single Business Component

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

An *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” on page 43.](#)

How Siebel CRM Creates an Implicit Join

Figure 5 describes how Siebel CRM creates an explicit join.

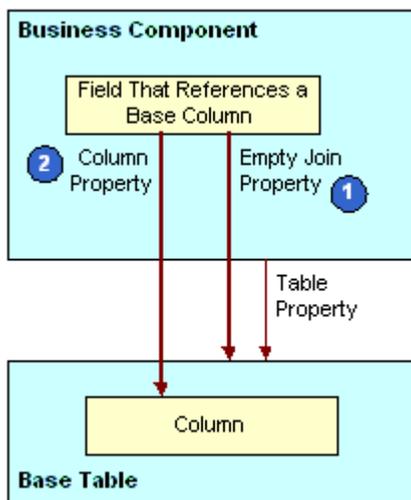


Figure 5. How Siebel CRM Creates an Implicit Join

Explanation of Callouts

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” on page 91](#).

How Siebel CRM Creates an Explicit Join

Figure 6 describes how Siebel CRM creates an explicit join.

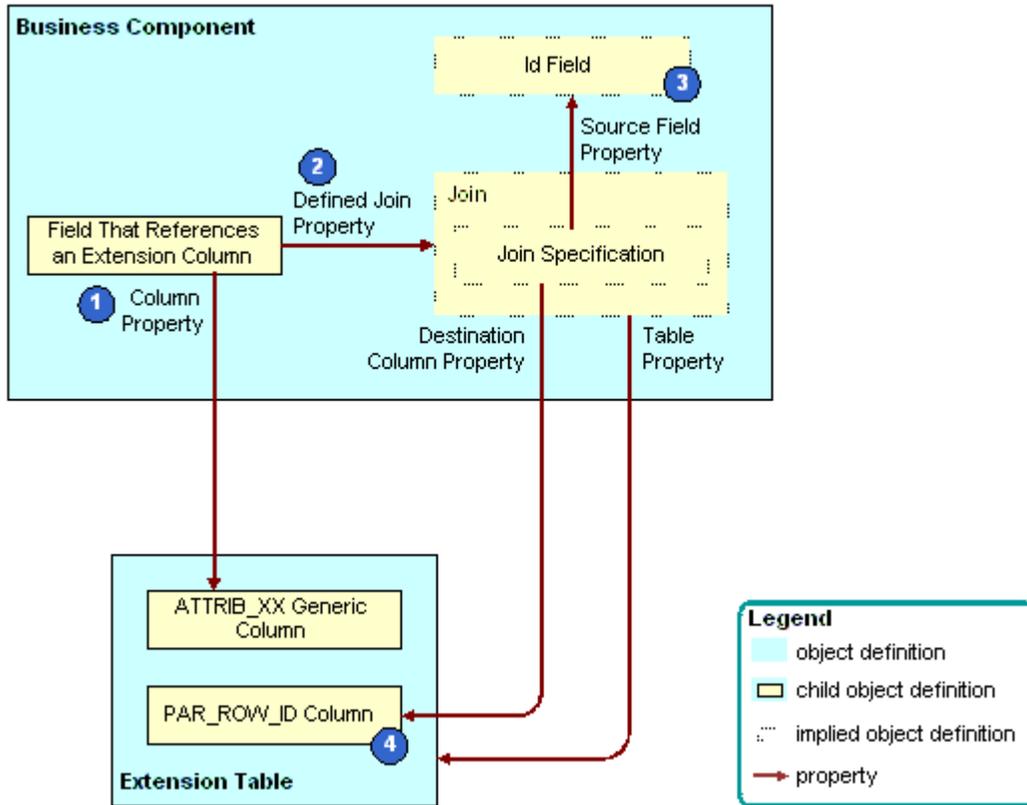


Figure 6. How Siebel CRM Creates an Explicit Join

Explanation of Callouts

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” on page 89](#).
- 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” on page 60](#).

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 on page 62](#)
- [Configuring Objects to Use a One-To-Many Extension Table on page 237](#)
- [Configuring a Business Component on page 245](#)
- [Configuring a Link That Creates a One-to-Many Relationship on page 263](#)
- [Creating a Business Object on page 266](#)

Summary of Support for Extension Tables and Extension Columns

Table 7 summarizes support for extension tables and extension columns.

Table 7. Summary of 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	<p>The following support is available for the private data table:</p> <ul style="list-style-type: none"> ■ Cannot contain an extension column ■ Cannot add an extension column to a private data table <p>A <i>private data table</i> is a table with the Type property set to Data (Private). Some interface tables are private, but most are public.</p>
Intersection table	<p>The following support is available for an intersection table:</p> <ul style="list-style-type: none"> ■ Can be extended with an extension column ■ Cannot be extended with a custom extension table

Table 7. Summary of Support for Extension Tables and Extension Columns

Object	Description
LOV Bounded, LOV Type property of a table column	<p>Read-only for a predefined column in Siebel CRM but is editable for a custom extension column.</p> <p>MLOV (multilingual list of values) is allowed with a custom extension column.</p>
Predefined one-to-one extension column	<p>It is recommended that you do not modify or delete a predefined one-to-one extension column.</p>
Predefined extension column	<p>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.</p>
Custom extension column	<p>The following support is available for a custom extension column:</p> <ul style="list-style-type: none"> ■ 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” on page 236.
Custom extension table	<p>The following is available for a custom extension table:</p> <ul style="list-style-type: none"> ■ 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 Table 10 on page 61.
EIM mapping	<p>The following support is available for Enterprise Integration Manager (EIM) mapping:</p> <ul style="list-style-type: none"> ■ The EIM Table Mapping Wizard allows you to create or associate a new table with an interface table that uses EIM: <ul style="list-style-type: none"> ■ You can create EIM Table Mapping objects to import data to a table that you define. ■ 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. <p>For more information, see “Mapping a Custom Table to an Interface Table” on page 518.</p>
Custom extension to a dock object	<p>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.</p>

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 on page 50](#)
- [How Siebel CRM Creates an Intersection Between Tables on page 51](#)
- [How Siebel CRM Creates a Many-To-Many Relationship on page 53](#)
- [Intersection Data in an Intersection Table on page 54](#)
- [How Siebel CRM Uses an Implicit Join With an Intersection Table on page 54](#)

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.

[Figure 7](#) describes an example of how an intersection table defines a many-to-many relationship.

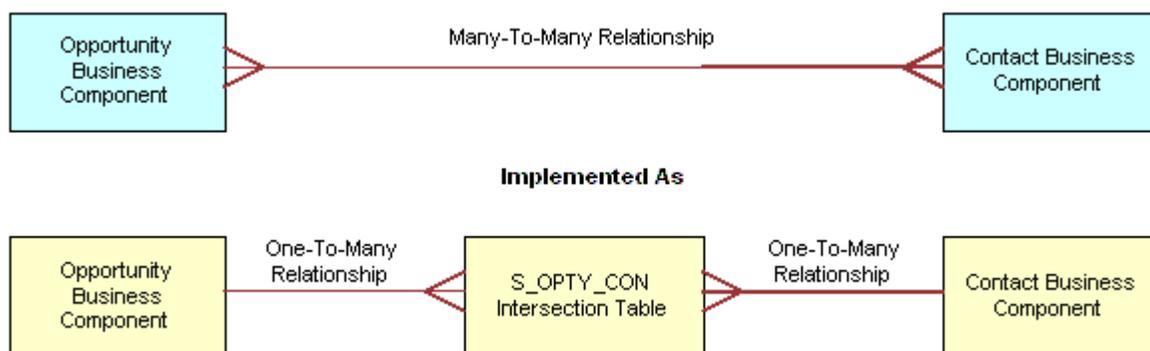


Figure 7. Example of How an Intersection Table Defines a Many-to-Many Relationship

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" on page 104](#).

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.

Figure 8 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.

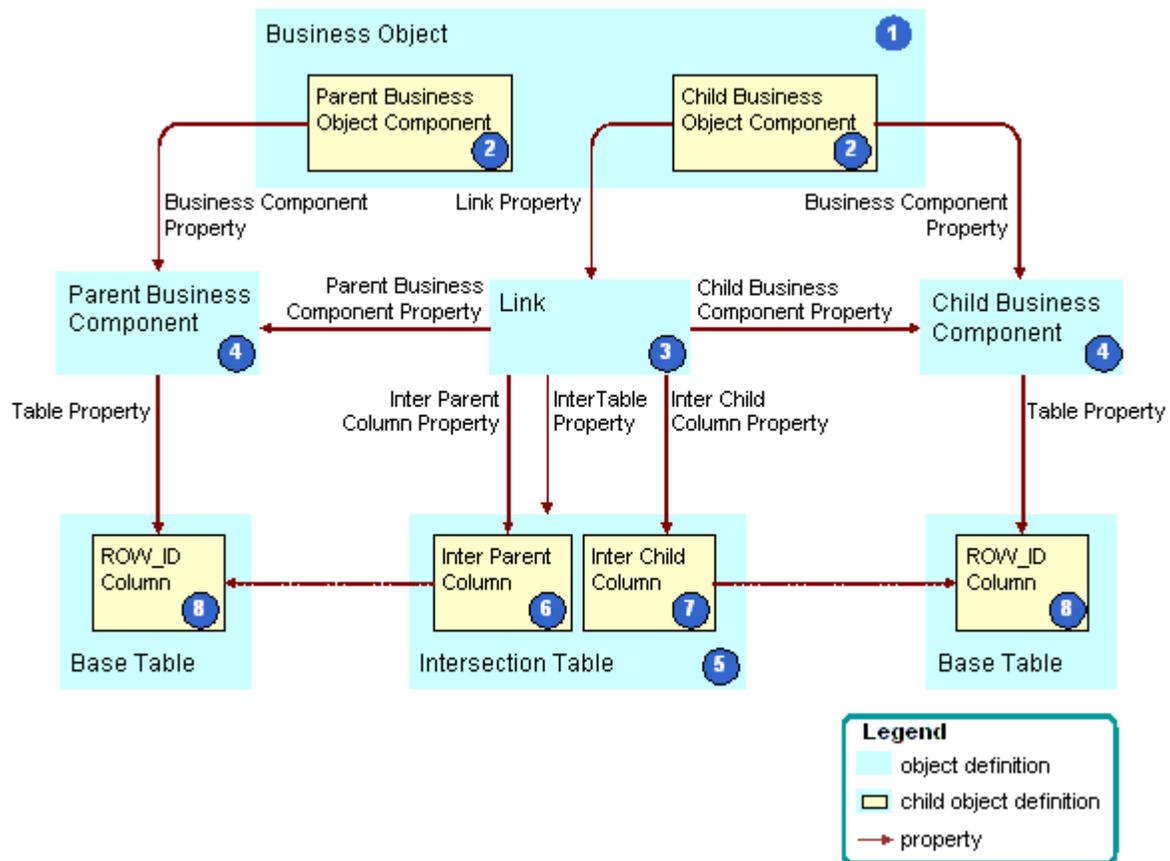


Figure 8. How Siebel CRM Creates an Intersection

Explanation of Callouts

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

Figure 9 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.

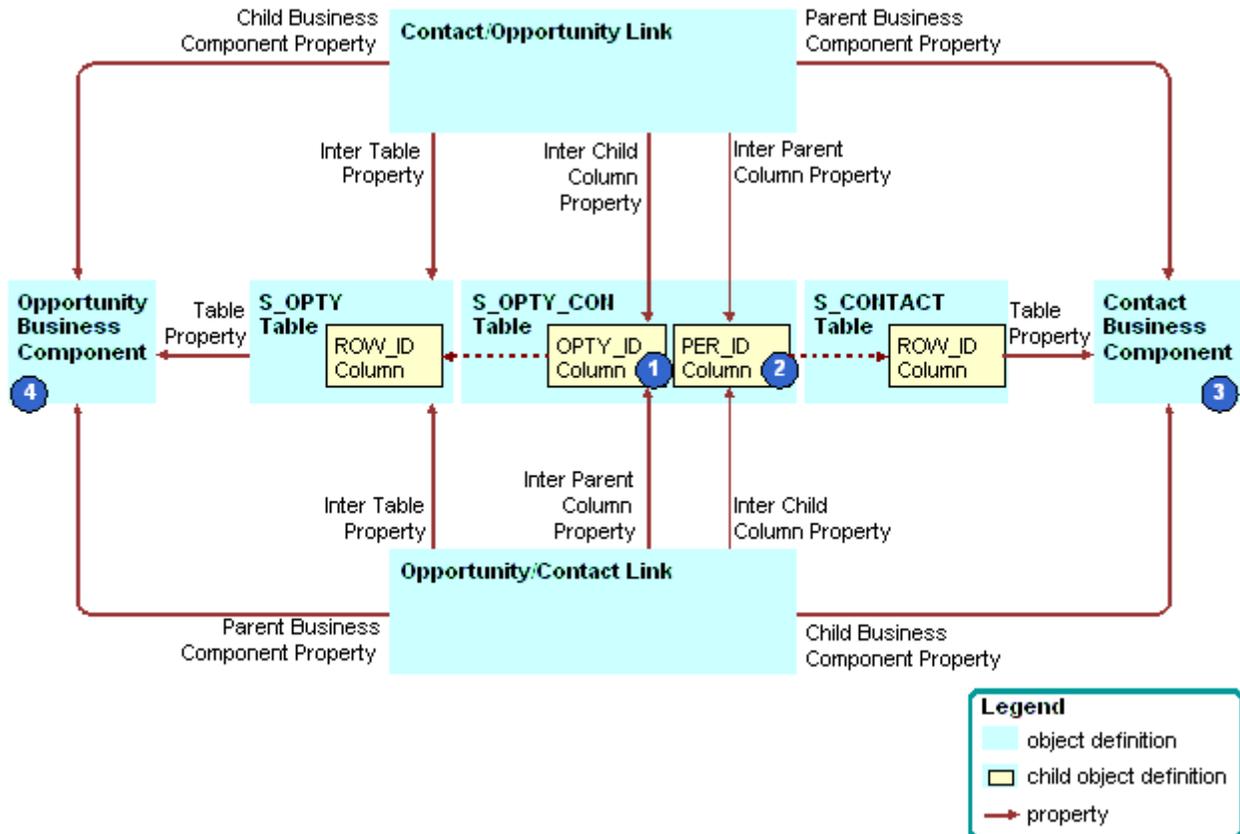


Figure 9. Example of How Siebel CRM Creates a Many-To-Many Relationship

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 on page 594](#)
- [How a CIAI Index Can Improve a Query on page 490.](#)

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” on page 89](#).

Table 8 describes some of the system columns that Siebel CRM commonly uses.

Table 8. 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” on page 89 .
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” on page 598](#) and [“Creating a Custom Index” on page 235](#).

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” on page 599](#) and [“Creating a Custom Index” on page 235](#).

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 *_Un* 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” on page 23](#).

For more information, see [“About Interface Tables” on page 511](#).

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.

Table 9 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.

Table 9. S_PARTY Extension Tables and Corresponding EIM Interface Tables

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
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 on page 59](#)
- [Options to Use a Predefined One-to-One Extension Table on page 60](#)
- [Options to Use a Predefined One-to-Many Extension Table on page 62](#)

For more information, see the following topics:

- [Configuring Tables on page 233](#)
- [Properties of a Siebel Table on page 594](#)

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” on page 60](#) and [“Options to Use a Predefined One-to-Many Extension Table” on page 62](#).

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” on page 236](#).

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” on page 233](#).

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 [Chapter 23, “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 [Chapter 24, “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” on page 65](#).

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” on page 43](#).

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.

Table 10 lists the different data types in a Siebel extension table and the number of columns of each data type.

Table 10. Data Types in a Predefined Extension Column

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” on page 43](#).

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 on page 63](#)
- [Guidelines for Creating a New Table on page 63](#)
- [Guidelines for Adding an Extension Column to a Base Table on page 63](#)
- [Guidelines for Creating a Custom Index on page 64](#)
- [Guidelines for Creating a LONG Column on page 64](#)
- [Guidelines for Modifying a Predefined One-to-One Extension Table on page 65](#)
- [Guidelines for Creating a Custom One-to-One Extension Table on page 66](#)
- [Guidelines for Configuring a Base Table or Configuring a One-To-Many Extension Table on page 66](#)
- [Guidelines for Configuring a Foreign Key That Affects Enterprise Integration Manager on page 66](#)
- [Guidelines for Creating a Custom Docking Rule on page 67](#)

For more information, see [“Guidelines for Reusing a Predefined Table” on page 205](#).

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” on page 60](#).
- 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” on page 41](#).

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” on page 42](#).
- 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.

Guidelines for Creating a LONG Column

If you create a LONG column, then use the following guidelines:

- Only one LONG column can exist for each table.

- You can add a LONG column only to a one-to-one extension table whose Base Table property includes a valid base table.
- You cannot add a LONG column to a one-to-many extension table because it is a Data (Public) table.
- You cannot add a LONG column to a Data (Public) table, such as the S_EVT_ACT table. Only Oracle can create a LONG column in a Data (Public) table.
- You can use a LONG column to store a maximum of 16 KB or 16383 characters.
- Querying a LONG column starts more input and output operations in your RDBMS that are not necessary with other types of column data. This extra input and output increases the time Siebel CRM requires to get each row of data from the database. This increase can add up to a noticeable reduction in performance if Siebel CRM gets many rows of data from the database.
- For DB2 on z/OS, use a 32K tablespace if 16K is too small. If 32K is too small, then convert the LONG type to a CLOB type. For more information, see *Implementing Siebel Business Applications on DB2 for z/OS*.

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 [Chapter 24, "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:

- If you must use a LONG column because the database permits only one LONG column for each database table.
- 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” on page 525](#).

4

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 on page 69](#)
- [About Business Component Fields on page 78](#)
- [About Joins on page 91](#)
- [About Multi-Value Links on page 95](#)
- [About Links on page 104](#)

About Business Components

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

- [Overview of Business Components on page 70](#)
- [How a Business Component Gets Data from an External Database on page 72](#)
- [Business Components That Hold Temporary Data for a Task UI on page 73](#)
- [Class Property of a Business Component on page 74](#)
- [How a Business Component Sorts Records on page 74](#)
- [Guidelines for Creating a Business Component on page 76](#)

For more information, see the following topics:

- [Overview of the Business Object Layer on page 28](#)
- [Configuring a Business Component on page 245](#)
- [Properties of a Business Component on page 599](#)

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” on page 29](#) and [“Options to Filter Data That Siebel CRM Displays in an Applet” on page 118](#).

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.

Figure 10 describes an example of how fields in the Contact business component reference columns in the S_CONTACT table.

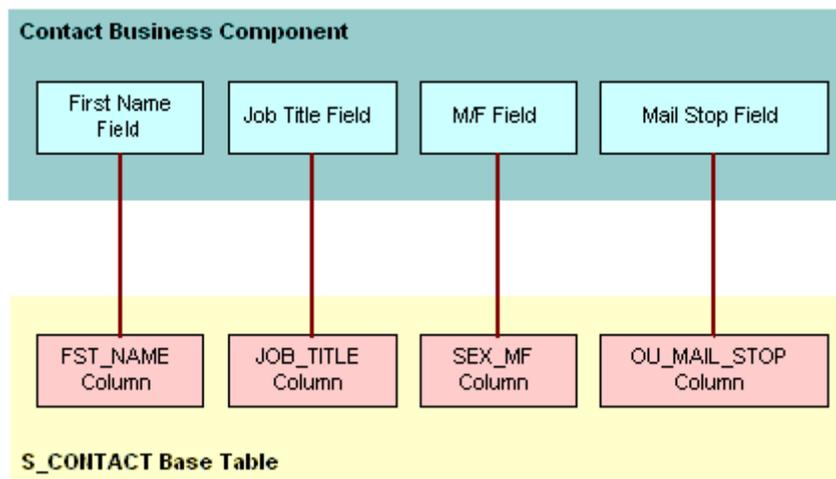


Figure 10. Example of How Fields in a Business Component Reference Columns in a Base Table

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”](#) on page 44.

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”](#) on page 58.

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.

Figure 11 describes fields in a business component that reference columns in a base table and a joined table.

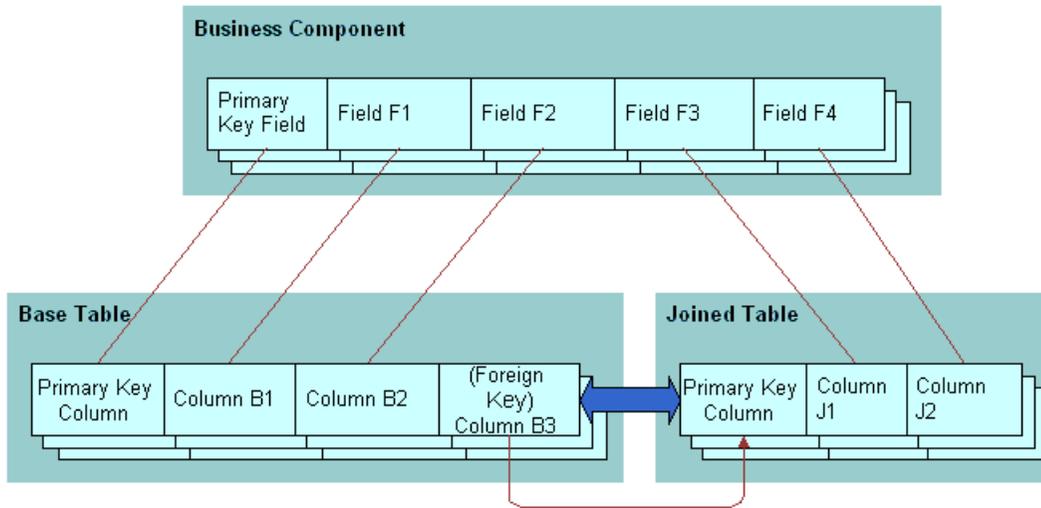


Figure 11. How Fields in a Business Component Reference Columns in a Base Table and a Joined Table

You Can Reuse a Business Component

Figure 12 describes how you can create a business component once in terms of a logical collection of columns from one or more tables, and then use it in multiple business object contexts. For more information, see “About Business Objects” on page 105.

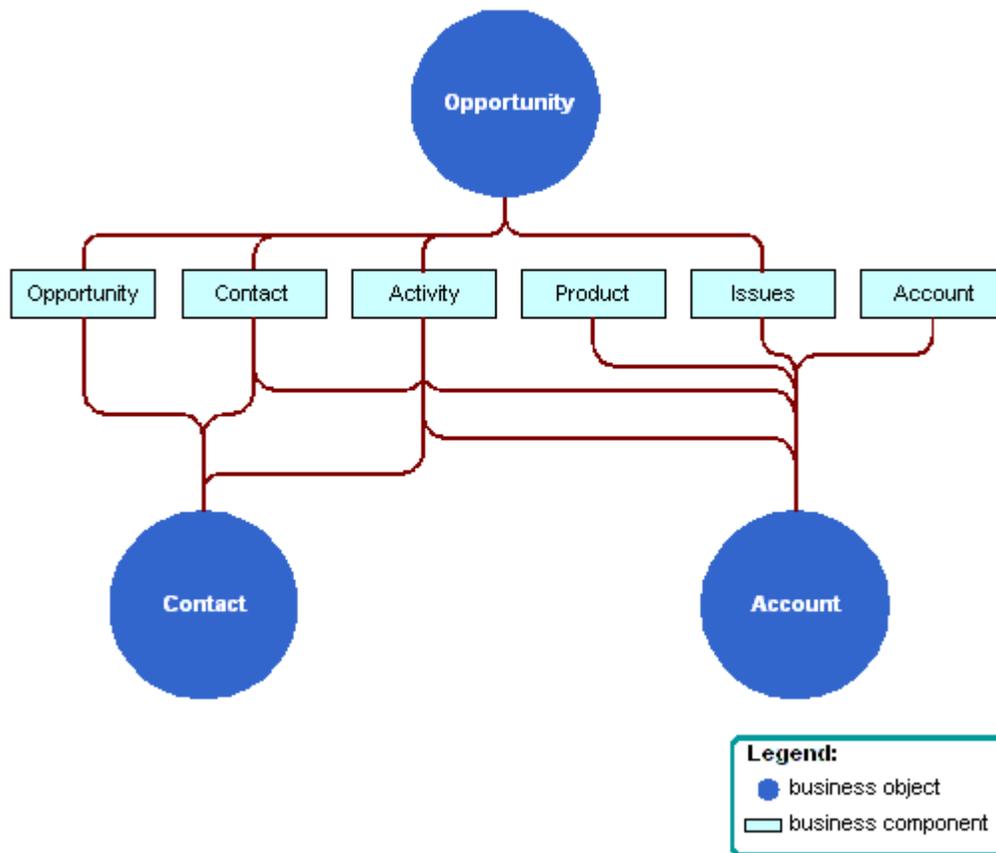


Figure 12. Example of How You Can Reuse a Business Component

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” on page 141](#).
- 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*.

Class Property of a Business Component

Siebel CRM contains a hierarchy of business component classes. The CSSBusComp class resides at the top 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” on page 410](#).

For more information, see the following topics:

- [Determining How a Business Component Sorts Records on page 246](#)
- [Guidelines for Configuring How a Business Component Sorts Records on page 77](#)
- [Configuring the Sort Order for Siebel CRM on page 298](#)

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” on page 98](#).

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 on page 189](#)
- [Guidelines for Reusing a Predefined Business Component on page 208](#)
- [Guidelines for Reusing a Predefined Business Object on page 209](#)

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 a top 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” on page 118](#).

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” on page 74](#).

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” on page 118](#).
- 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 on page 78](#)
- [How a Business Component Field Identifies the Type of Data on page 81](#)
- [How a Business Component Field Calculates a Value on page 81](#)
- [How a Business Component Field Sequences Records on page 82](#)
- [How Siebel CRM Defines Read-Only Behavior for a Business Component Field on page 85](#)
- [System Fields of a Business Component on page 89](#)
- [Guidelines for Defining the Name of a Business Component Field on page 90](#)

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” on page 81](#).

For more information, see [“Business Component Field” on page 29](#).

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

Table 11 describes several examples of how a business component field provides data to the control or list column of an applet.

Table 11. 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.

Figure 13 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”](#) on page 44.

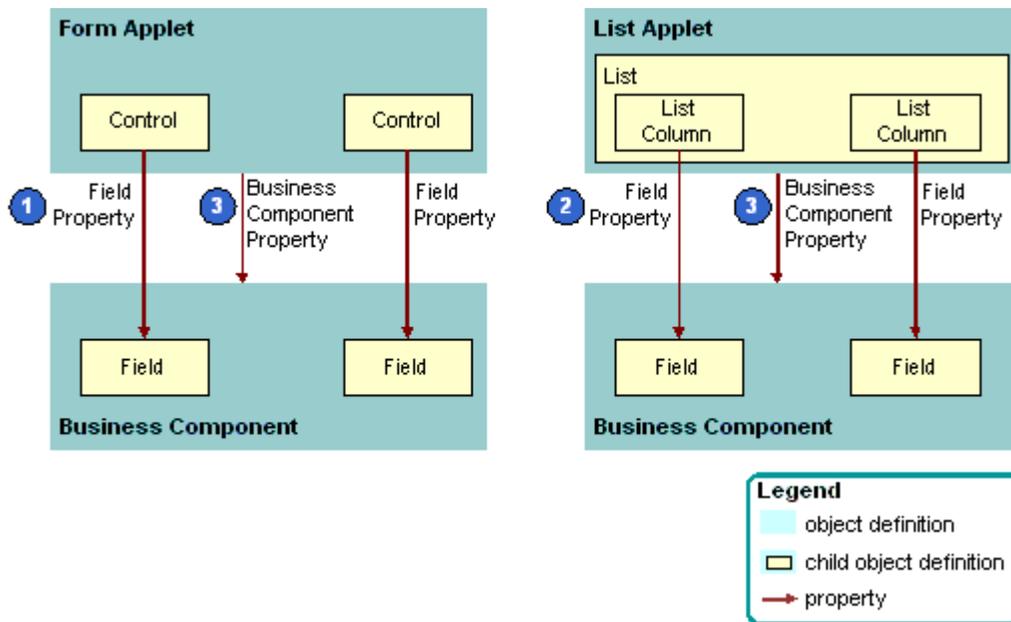


Figure 13. Comparison Between How a Form Applet and a List Applet Reference a Business Component

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” on page 600](#).

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” on page 605](#).

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” on page 98](#).

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 on page 246](#).

How Siebel CRM Creates a Sequence Field

Figure 14 describes how Siebel CRM creates a sequence field.

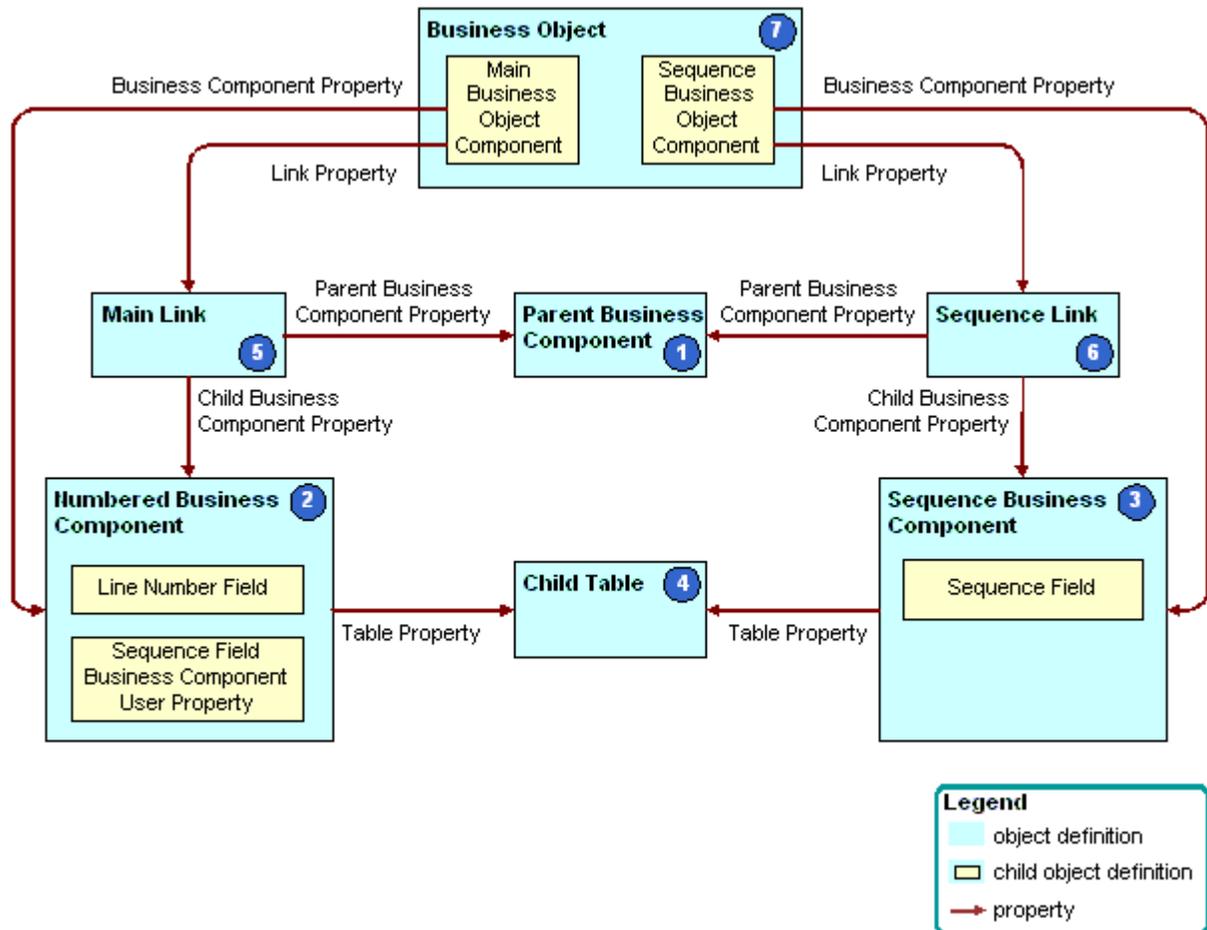


Figure 14. How Siebel CRM Creates a Sequence Field

Explanation of Callouts

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” on page 74](#).

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

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 on page 86](#)
- [How the Field Read Only Field User Property Works on page 86](#)
- [How the Parent Read Only Field User Property Works on page 87](#)
- [How the Parent Read Only Field: Business Component Name User Property Works on page 88](#)
- [Guidelines for Using a Business Component User Property With the Admin Mode Property on page 88](#)

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” on page 248](#).

[Table 12](#) 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*.

Table 12. Business Component User Properties That You Can Use to Define Read-Only Behavior

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” on page 86 .
Field Read Only Field: <i>fieldname</i>	<p>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.</p> <p>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.</p> <p>For more information, see “How the Field Read Only Field User Property Works” on page 86.</p>

Table 12. Business Component User Properties That You Can Use to Define Read-Only Behavior

Business Component User Prop	Description
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” on page 87.
Parent Read Only Field: <i>business component name</i>	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” on page 88.

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 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 [“Example of Using the BC Read Only Field User Property” on page 86](#) 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” on page 56](#), and [“Displaying a System Field in an Applet” on page 351](#).

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.

Table 13 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.

Table 13. Relationship Between a System Field and a System Column

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” on page 504.
(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 on page 189](#)
- [Chapter 10, “Reusing Predefined Objects”](#)
- [Guidelines for Reusing a Predefined Object on page 204](#)

About Joins

This topic describes joins. It includes the following information:

- [How Siebel CRM Creates a Join on page 92](#)
- [Guidelines for Creating a Join on page 94](#)

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” on page 29](#).

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.

[Figure 15](#) 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.

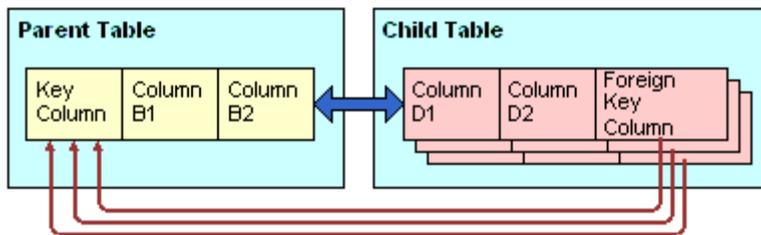


Figure 15. Parent-Child Relationship in a Join

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” on page 43](#).

How Siebel CRM Creates a Join

Figure 16 describes how Siebel CRM creates a join.

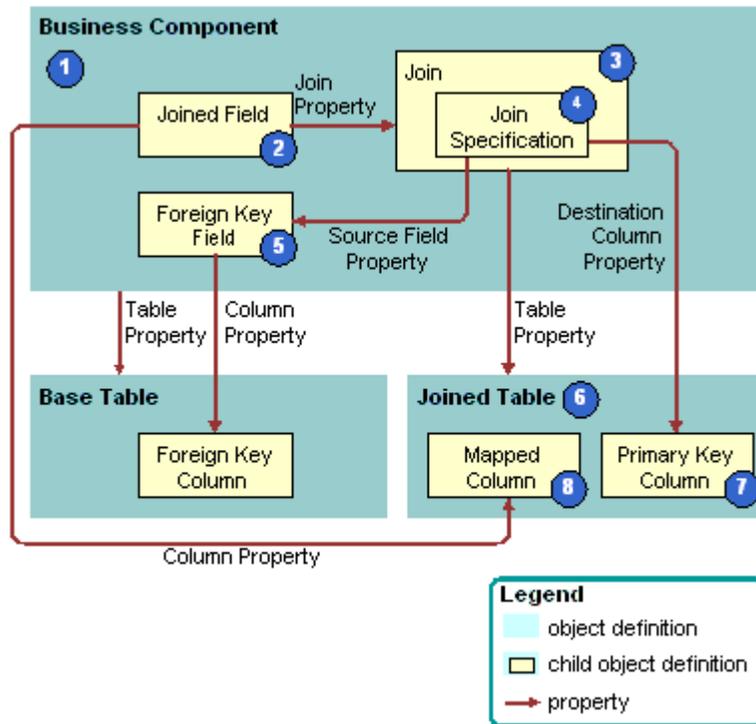


Figure 16. How Siebel CRM Creates a Join

Explanation of Callouts

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” on page 93](#).
- 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” on page 118](#).

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” on page 93](#).
- 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” on page 89](#).

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” on page 29](#).

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” on page 189](#).

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” on page 58](#).

About Multi-Value Links

This topic describes the multi-value link. It includes the following information:

- [How Siebel CRM Creates a Direct Multi-Value Link on page 96](#)
- [How Siebel CRM Creates an Indirect Multi-Value Link on page 99](#)

A multi-value link 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 applet references. The multi-value link 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 multi-value group contains.

A parent-child relationship exists between the business component that the originating applet references and the business component that the multi-value group 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 multi-value link and multi-value group. A many-to-many relationship can exist. For example, between opportunities and positions. In the context of the multi-value group, Siebel CRM presents only one parent-child relationship.

For more information, see [“Multi-Value Group, Multi-Value Link, and Multi-Value Group Applet” on page 30](#).

How Siebel CRM Creates a Direct Multi-Value Link

Figure 17 describes how Siebel CRM creates a direct multi-value link.

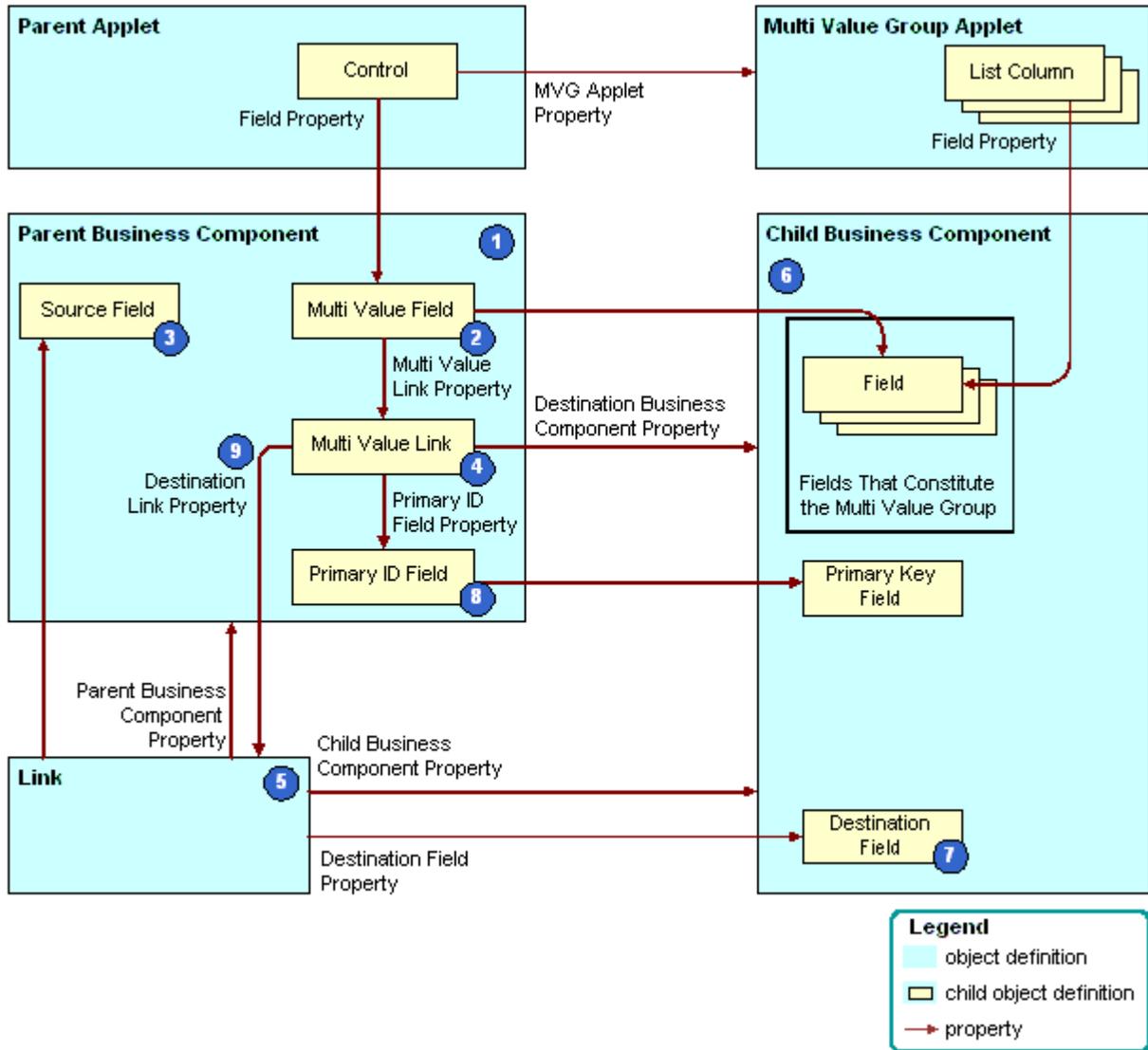


Figure 17. How Siebel CRM Creates a Direct Multi-Value Link

Explanation of Callouts

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” on page 424](#), 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” on page 98](#).
- 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” on page 424](#), 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” on page 424](#), 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” on page 424](#), the foreign key field is Account Id. For more information, see [“About the Destination Field” on page 97](#).

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” on page 98](#).
- 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. 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 does not 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 are not empty. In this situation, the Source Field and Destination Field properties are not 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 the following items:

- Contains the row ID value of the primary record for each record of the multi-value group 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” on page 504](#) and [“Determining Whether You Can Reuse a Predefined Business Component Field” on page 217](#).

About the Multi-Value Field

The *multi-value field* is a business component field that contains the name of the multi-value link in the Multi Value Link property and a check mark in the Multi Valued 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 multi-value link.

If you must query 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.

[Table 14](#) describes some of the properties of the multi-value field.

Table 14. Some of the Properties of the Multi-Value Field

Property	Description
Multi Value Link	Identifies the multi-value link 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 multi-value link 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.
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 multi-value link object and the link object.

In [“Viewing an Example of a Multi-Value Group Applet” on page 424](#), 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” on page 75](#) and [“Activating a Multi-Value Field” on page 252](#).

How Siebel CRM Creates an Indirect Multi-Value Link

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

Figure 18 describes how Siebel CRM creates an indirect multi-value link.

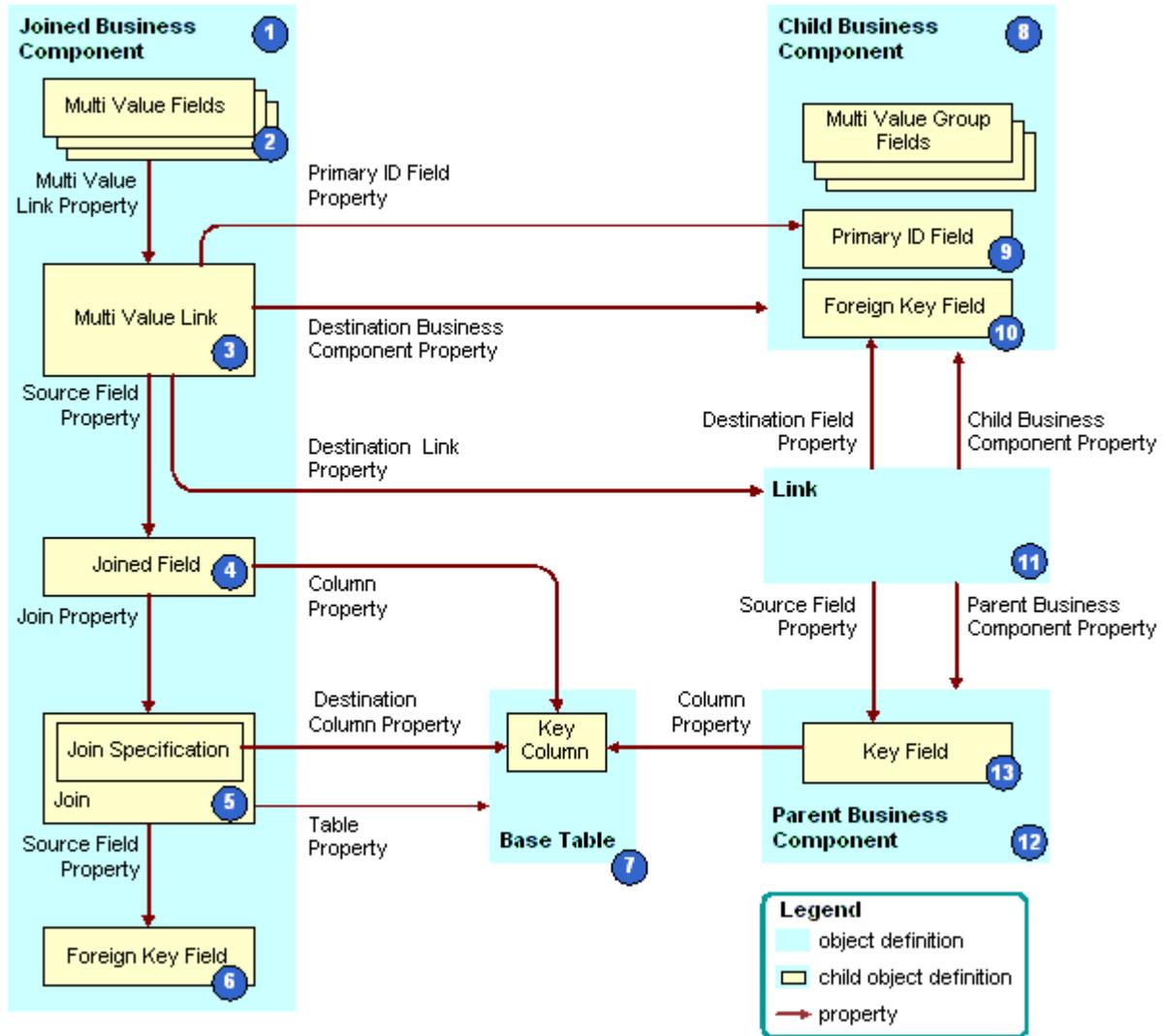


Figure 18. How Siebel CRM Creates an Indirect Multi-Value Link

Explanation of Callouts

Siebel CRM uses the following objects to create an indirect multi-value link:

- 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 multi-value link as a child of the join business component.
- 2 Multi-value fields.** For more information, see [“About the Multi-Value Field”](#) on page 98.

- 3 **Multi-Value link.** The multi-value link uses the following 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 multi-value link. For more information, see [“About the Primary ID Field” on page 98.](#)
 - **Destination Business Component property.** Identifies the child business component.
 - **Destination Link property.** Identifies the link.
- 4 **Joined field.** The Source Field property in:
 - A multi-value link is empty.
 - An indirect multi-value link defines a joined field in the same business component as the multi-value link. 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. 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” on page 91.](#)
- 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” on page 98.](#)
- 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 is 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

Table 15 describes some of the objects that Siebel CRM uses to create an indirect multi-value link that involves the Business Address in the Contact business component. The Contact business component and the Account business component each contain the Business Address multi-value link.

Table 15. Example of How Siebel CRM Creates an Indirect Multi-Value Link

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
Link	Account Address
Parent Business Component	Account

How Siebel CRM Uses the Source Field Property

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

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 multi-value link is the same as the parent business component of the underlying link that the multi-value link references.

For example, [Table 16](#) lists some of the properties that Siebel CRM defines for the Business Address multi-value link in the Account business component.

Table 16. Properties of the Business Address Multi-Value Link 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 multi-value link references the Account/Business Address link. [Table 17](#) lists some of the properties that Siebel CRM defines for the Account/Business Address link.

Table 17. Properties of the Account/Business Address Link

Property	Value
Name	Account/Business Address
Parent Business Component	Account
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 multi-value link 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 is currently chosen in the Account business component.

Usage of a Predefined Indirect Multi-Value Link

If a predefined link exists that is appropriate for use in a multi-value link, and if the originating business component is different from the parent business component, then you can use an indirect multi-value link instead of a conventional multi-value link. 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 multi-value link.

5

About Business Objects

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

- [About Business Objects on page 105](#)
- [How Siebel CRM Creates a Business Object on page 108](#)
- [Guidelines for Creating a Business Object on page 109](#)

About Business Objects

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” on page 29](#). For an introduction to the relationships that this topic describes, see [“Hierarchy of Object Types and Relationships” on page 31](#).

[Figure 20](#) 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, and other business components.

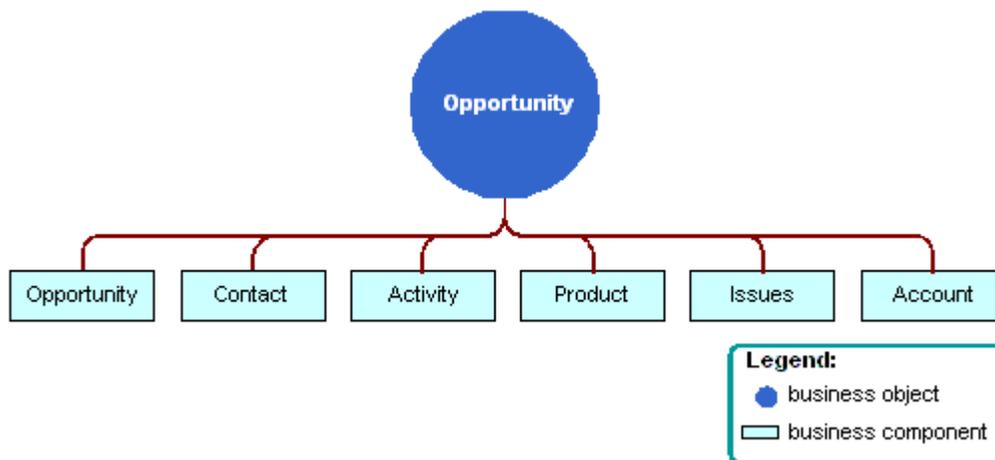


Figure 20. How The Opportunity Business Object Groups Business Components

Each business object includes one business component that works as the parent business component. In [Figure 20](#) 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.

Figure 21 describes the relationships and objects that Siebel CRM uses with a business object, screen, and view.

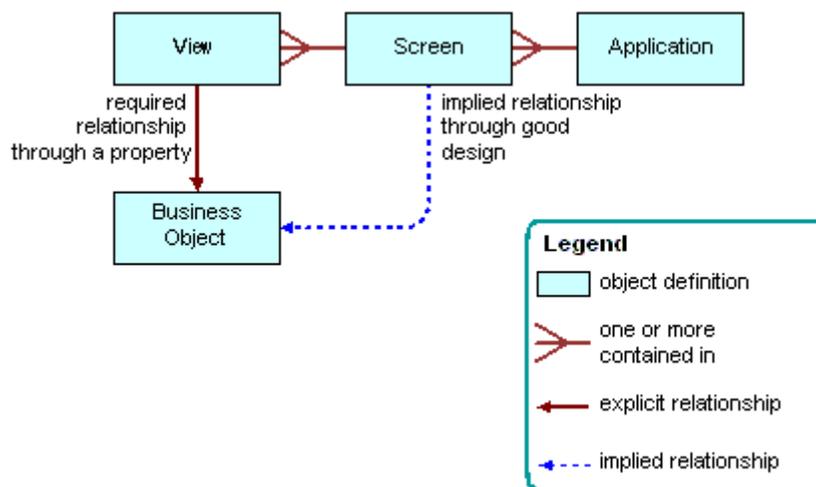


Figure 21. Relationships and Objects That Siebel CRM Uses with a Business Object, Screen, and View

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 above 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

Figure 22 describes how Siebel CRM creates a business object.

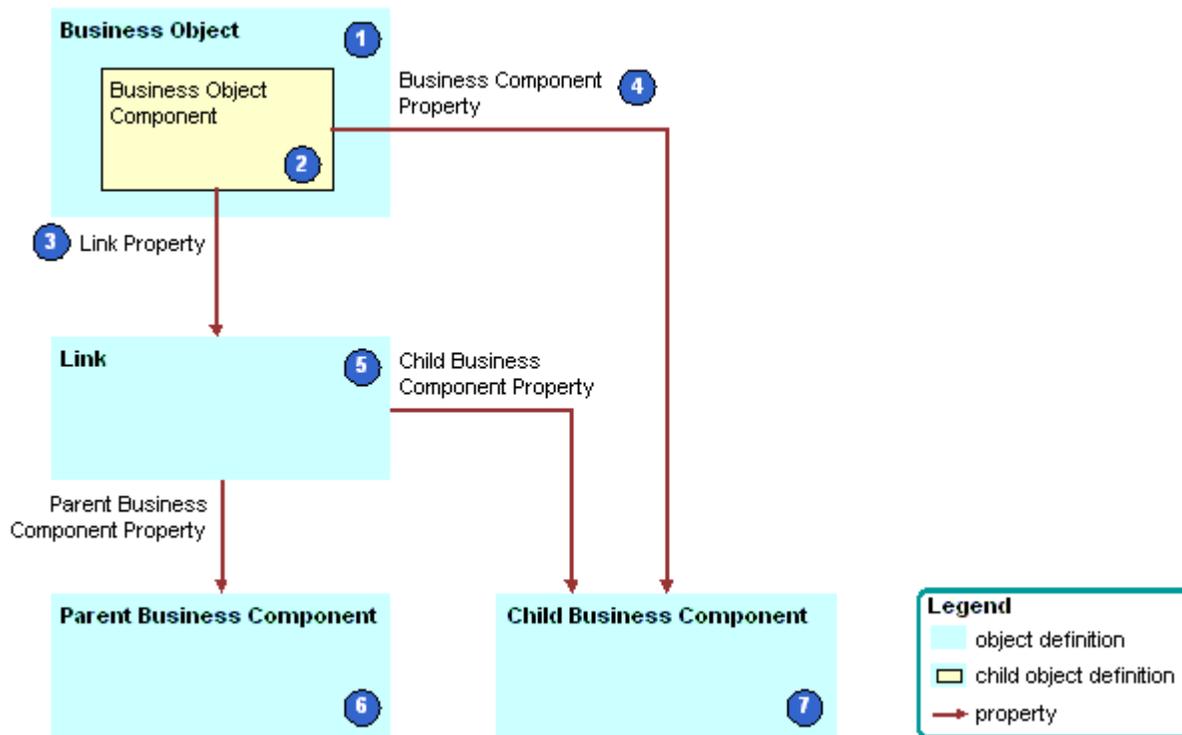


Figure 22. How Siebel CRM Creates a Business Object

Explanation of Callouts

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” on page 104](#).
- 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” on page 110](#).
- 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” on page 118](#).

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” on page 189](#).

6

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 on page 111](#)
- [About Applet Controls and List Columns on page 113](#)
- [Options to Create an Applet on page 115](#)
- [Guidelines for Creating an Applet on page 125](#)
- [Guidelines for Creating a Control or List Column on page 127](#)

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 on page 111](#)
- [How Siebel CRM Creates a List Applet on page 112](#)

An applet allows the user to view, enter, and modify data that it gets from a single business component. For more information, see [“Applet” on page 26](#).

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:

- [Chapter 17, “Configuring Special-Purpose Applets”](#)
- [Chapter 19, “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.

Figure 23 describes how Siebel CRM creates a form applet. For more information, see “Hierarchy of Object Types and Relationships” on page 31.

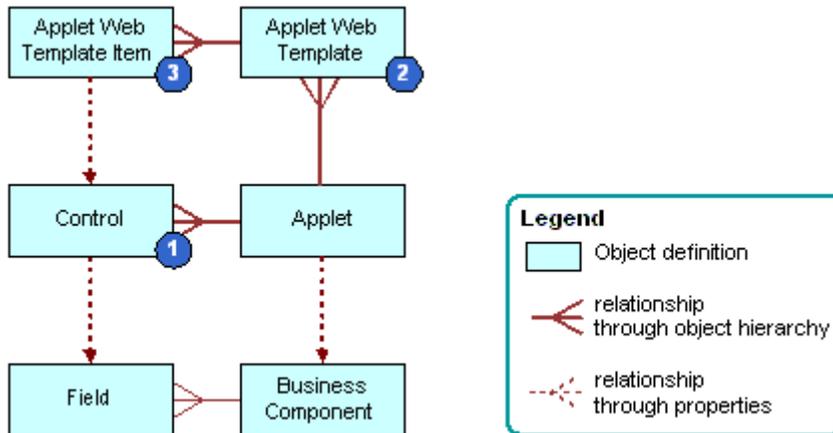


Figure 23. How Siebel CRM Creates a Form Applet

Explanation of Callouts

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” on page 113.
- 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” on page 145 and “Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data” on page 116.
- 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 Layout Editor to drag and drop 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” on page 155.

For more information, see “How a Business Component Field Provides Data to the Control or List Column of an Applet” on page 79 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.

Figure 24 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” on page 111. For background information, see “Hierarchy of Object Types and Relationships” on page 31.

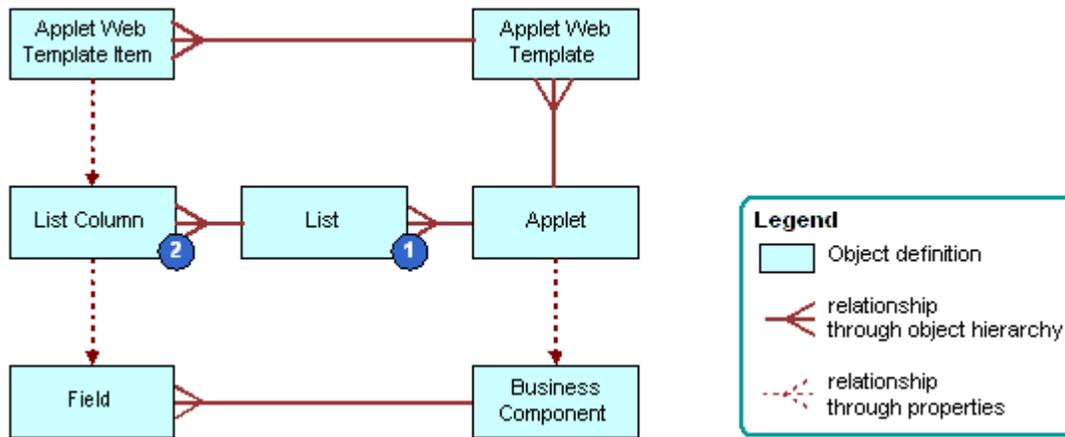


Figure 24. How Siebel CRM Creates a List Applet

Explanation of Callouts

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” on page 113.

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 on page 114](#)

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” on page 138](#).

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” on page 79](#).

You use the New Applet Wizard to create controls and list columns for a new applet. For a predefined applet, you use the Applet Layout 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” on page 305](#) and [“Adding a Control or List Column to an Applet Layout” on page 305](#).

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” on page 613](#), 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 in the CCHtmlType.swf file. This file is located in the WEBTMPL folder of your Siebel installation folder. For more information, see [“Configuring an HTML Control Type” on page 483](#).

For more information, see [“Using Declarative Configuration to Enable a Button” on page 508](#).

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” on page 341](#) and [“TextArea” on page 616](#).

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 gray 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 right 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 right 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” on page 389](#).

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 on page 116](#)
- [Options to Filter Data That Siebel CRM Displays in an Applet on page 118](#)

- [Options to Determine How Siebel CRM Displays Controls and List Columns in a Siebel Application on page 123](#)

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” on page 344](#).

Table 18 describes the types of applet modes.

Table 18. Types of Applet Modes

Applet Mode	Description
Edit	<p>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.</p> <p>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” on page 146.</p>
Edit List	<p>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.</p> <p>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.</p>
Base	<p>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.</p> <p>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.</p> <p>You can use the Read-Only field in the Responsibility Administration View to make a view read-only. For more information, see <i>Siebel Security Guide</i>.</p>

Table 18. Types of Applet Modes

Applet Mode	Description
New	<p>Allows the user to create a new record where the requirements for the new mode are different from the edit or edit list mode.</p> <p>Only use the New and Query modes if the Edit mode does not meet your required functionality.</p>
Query	<p>Allows the user to query if the requirements for the Query mode are different from the requirements for the Edit or Edit List mode. Allows the user to perform a query-by-example (QBE).</p>

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 top applet in a view is in Query mode, then the child bottom 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 Layout 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” on page 305](#).
- Controls appearance of the minibutton. For more information, see [“MiniButton Control and MiniButton List Column” on page 114](#).
- 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” on page 432](#).

For more information, see the following topics:

- [Process of Using the Applet Layout Editor on page 303](#)
- [Process of Creating a Screen Home Page View on page 285](#)
- [Pick Applet Usage in Query Mode on page 395](#)
- [About Siebel Web Templates on page 145](#)

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 on page 346](#)
- [Modifying Custom Search Specifications on page 507](#)
- *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

Table 19 describes some of the parts of a search specification.

Table 19. Parts of a Search Specification

Element	Description
Comparison Operator	<p>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:</p> <ul style="list-style-type: none"> ■ = (equal to) ■ <> (not equal to) ■ > (greater than) ■ < (less than) ■ >= (greater than or equal to) ■ <= (less than or equal to) <p>The following is an example search specification that uses the greater than comparison operator:</p> <pre>[Revenue] > 5000</pre>
String Constant	<p>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:</p> <pre>[Type] <> "COST LIST"</pre>
Logical Operator	<p>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:</p> <pre>[Competitor] IS NOT NULL and [Competitor] <> "N"</pre>
Field Name	<p>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:</p> <pre>[Conflict Id] = 0</pre>

Table 19. Parts of a Search Specification

Element	Description
LIKE Operator	<p>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:</p> <ul style="list-style-type: none"> ■ * (asterisk). Indicates any number of characters. ■ ? (question mark). Indicates a single character. <p>The following is an example search specification that uses the LIKE operator:</p> <p style="padding-left: 40px;">[Last Name] LIKE "Sm*"</p> <p>In this example, the Last Name values of Smith, Smythe, Smallman, and so on, causes the search specification to evaluate to TRUE.</p>
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 you 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

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 list of the Configuration Context toolbar determines how Siebel 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.** Siebel Tools applies any add, move, or delete of a control or list column only to the chosen application.

If you choose a different application in the Application list of the Configuration Context toolbar during an editing session, then Siebel Tools modifies the appearance of the Applet Layout Editor 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. For more information, see [“Setting the Configuration Context” on page 304](#).

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 Business 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 Layout Editor, then Siebel Tools sets the Expression property.

For more information, see [“Properties of the Applet Web Template Item” on page 155](#).

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” on page 155](#).

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 *Name2*. 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” on page 155](#).

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 on page 126](#)
- [Guidelines for Creating a Check Box on page 128](#)

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” on page 209](#).
- 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” on page 190](#).
- 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.
- Avoid using a special character in an applet name.
- Use only alphanumeric characters.
- 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.

Table 20 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” on page 189](#).

Table 20. Naming Formats for an Applet

Type of Applet	Name Format	Example
Association applet	<i>description</i> Assoc Applet	Opportunity Assoc Applet
multi-value group applet	<i>business component</i> Applet	Fulfillment Position Mvg Applet
Pick applet	<i>description</i> Pick Applet	Order Status Pick Applet
List applet	<i>business component name</i> List Applet	Account List Applet
Form applet	If the applet does not contain buttons, then use <i>business component name</i> Form Applet. If the applet contains buttons, then use <i>business component name</i> Entry Applet.	The following examples use this format: <ul style="list-style-type: none"> ■ Account Form Applet ■ Account Entry Applet
Chart applet	<i>description</i> Chart Applet - <i>description</i> Analysis	Bug Chart Applet - Severity Analysis
Tree applet	<i>description</i> 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.

Table 21 describes formats for an applet title. For more information, see [“Guidelines for Naming an Object” on page 189](#).

Table 21. Formats for an Applet Title

Type of Applet	Title Format	Example
Association applet	Add <i>business component name</i>	Add Opportunities
multi-value group applet	<i>business component name</i>	Contacts
Pick applet	Pick <i>business component name</i>	Pick Product
List applet	<i>business component name</i> List	Account List
Form applet	Use one of the following formats: <ul style="list-style-type: none"> ■ <i>business component name</i> Form ■ <i>business component name</i> Entry 	The following examples use this format: <ul style="list-style-type: none"> ■ Account Form ■ Account Entry
Chart applet	Use one of the following formats: <ul style="list-style-type: none"> ■ <i>type of action</i> Analysis ■ <i>description by description</i> 	The following examples use this format: <ul style="list-style-type: none"> ■ Open Defect Analysis ■ Lead Quality By Campaign
Tree applet	<i>business component name</i>	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 on page 128](#)
- [Guidelines for Creating a Text Control or List Column on page 128](#)

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.

- Left-align fields in a form or list.
- Right-align labels 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” on page 416](#)
- Avoid using a negative. For example, instead of Not Required, use Optional.
- In a form applet, right-align the labels and position them to the left of 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 a red asterisk to indicate that a field is required, as you can in a form applet. For more information, see [“Caution About Using Specialized Classes” on page 74](#).
- You cannot apply a background color to a list column.

7

About Views, Screens, and Applications

This chapter describes views, screens, and applications. It includes the following topics:

- [About the Siebel Client Navigation Model on page 129](#)
- [About Views on page 130](#)
- [About Screens on page 133](#)
- [Options to Create a View or Screen on page 138](#)
- [About Applications on page 142](#)

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 below 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 below 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 on page 130](#)
- [About Master-Detail Views on page 131](#)

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” on page 26](#).

About List-Form Views

A *list-form view* 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 above 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 above 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

Figure 25 describes how Siebel CRM creates a master-detail view.

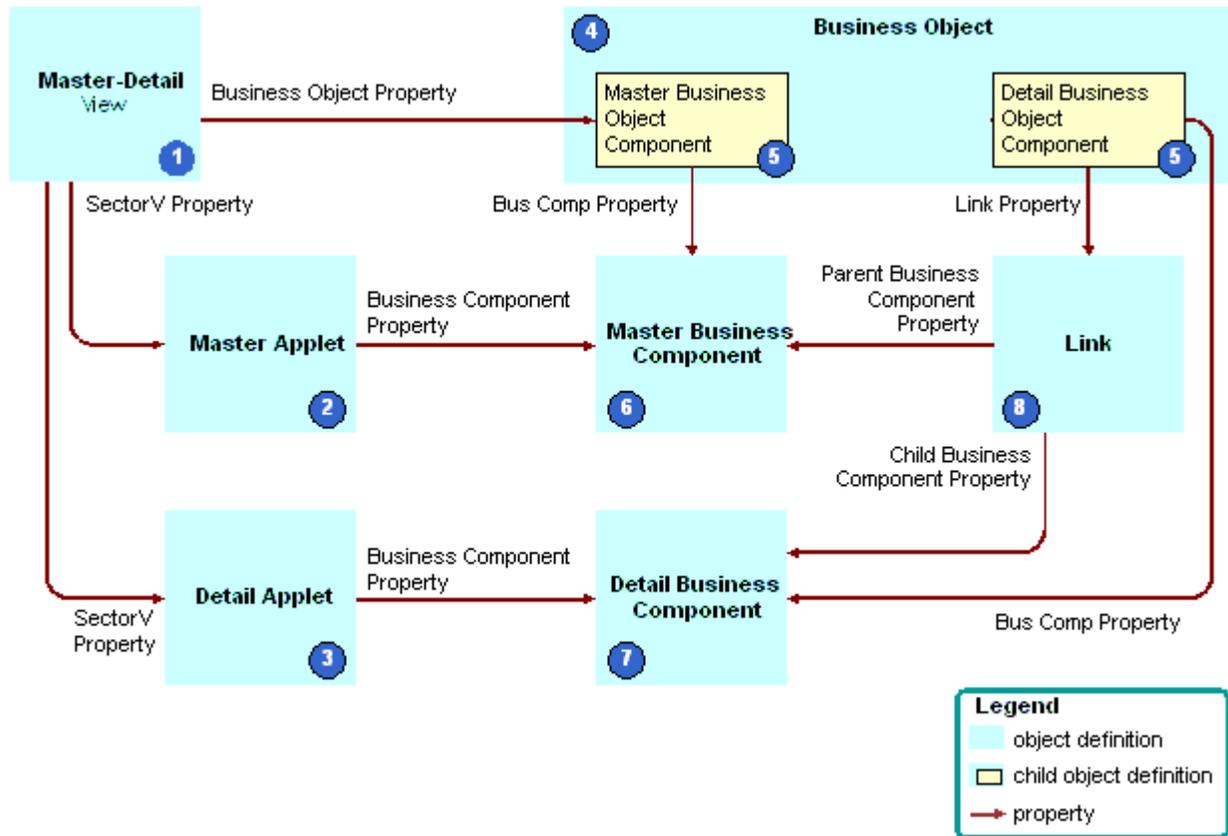


Figure 25. How Siebel CRM Creates a Master-Detail View

Explanation of Callouts

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 on page 69](#)
- [About Links on page 104](#)
- [About Business Objects on page 105](#)

About Screens

This topic describes screens. It includes the following information:

- [About Screen Views on page 133](#)
- [Guidelines for Creating a View on page 136](#)

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” on page 129](#) and [“Process of Creating a Screen” on page 281](#).

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” on page 283](#).

Table 22 describes types of screen views.

Table 22. 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 below screen tabs.
Aggregate View	Siebel CRM displays an Aggregate View as follows: <ul style="list-style-type: none"> ■ If no value is defined for the Parent Category property, then Siebel CRM displays the screen view as a link in the link bar below 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: <ul style="list-style-type: none"> ■ 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 below 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. <p>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.</p>

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

Table 23 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”](#) on page 129 and [“Process of Creating a Screen”](#) on page 281.

Table 23. Type of Screen View That Is Defined for Each Navigation Level

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” on page 282.
2	Links in the link bar directly below screen tabs. These links refer to groups of views.	Screen view with the following properties: <ul style="list-style-type: none"> ■ Type is Aggregate Category ■ Parent Category is empty
	Links in the link bar directly below screen tabs These links refer to a single view.	Screen view with the following properties: <ul style="list-style-type: none"> ■ Type is Aggregate View ■ Parent Category is empty
	Links in the view list in an applet header	Screen view with the following properties: <ul style="list-style-type: none"> ■ 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: <ul style="list-style-type: none"> ■ 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: <ul style="list-style-type: none"> ■ Type is Detail View ■ Parent Category is an Aggregate Category
4	Links in the link bar below view tabs, or in an alternate location depending on the web template.	Screen view with the following properties: <ul style="list-style-type: none"> ■ Type is Detail View ■ Parent Category is Detail Category

Example of a Screen View Hierarchy

Table 24 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.

Table 24. Example of a Screen View Hierarchy

Level	Siebel Client Location	Type	Category Name	Category Default View	View	Parent Category
2	Link in link bar below screen tabs	Aggregate View	Not applicable	Not applicable	Account Screen HomePage View	Not applicable
	Link in link bar below 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 below 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” on page 271](#).

- 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 siebns.dat file. The Server Manager displays these views in the Siebel client. Any modification that you make to one of these views must be stored in siebns.dat file. You cannot store this information in siebns.dat. 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 a special character, such as an ampersand (&).

Guidelines for Naming a View According to the Type of View

Table 25 describes guidelines for naming a view according to the type of view.

Table 25. Guidelines for Naming a View According to the Type of View

Type of View	Name Format	Example
List-form view	<i>business component</i> List View	Account List View
Master-detail view	<i>detail business component</i> Detail - <i>master business component</i> View	Opportunity Detail - Contacts View
Explorer view	<i>business component</i> Explorer View	Account Explorer View
Chart view	<i>master business component</i> Chart View - <i>detail business component</i> Analysis	Account Chart View - State Analysis

Guidelines for Naming a View According to the Type of Aggregate View

Table 26 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” on page 189.

Table 26. Guidelines for Naming a View According to the Type of Aggregate View

Type of Aggregate View	Example
Personal	My Personal <i>Contacts</i>
Sales Rep	My <i>Contacts</i>
Manager	My Team's <i>Contacts</i>
Organization	All <i>Contacts</i>
Sub Organizations	All <i>Accounts</i> Across My Organizations
All	All <i>Contacts</i> Across Organizations
Group	User <i>Catalog List View</i>
Catalog	<i>Products</i> Across Catalogs
Admin Mode	<i>Contacts</i> 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 on page 138](#)
- [Options to Toggle Between Applets in a View on page 141](#)

For more information, see the following topics:

- [Chapter 14, “Configuring Views, Screens, and Applications”](#)
- [Guidelines for Reusing a Predefined View on page 210](#)

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” on page 118.

- 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.

Figure 26 describes how Siebel CRM creates a static drilldown.

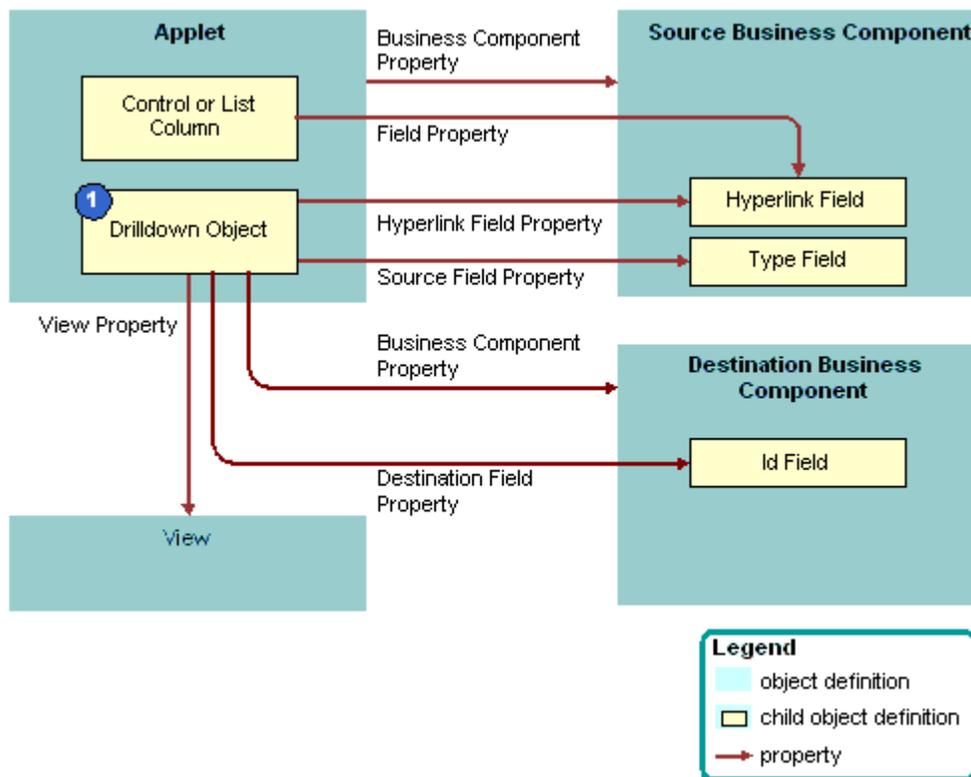


Figure 26. How Siebel CRM Creates a Static Drilldown

Explanation of Callouts

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.

Figure 27 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.

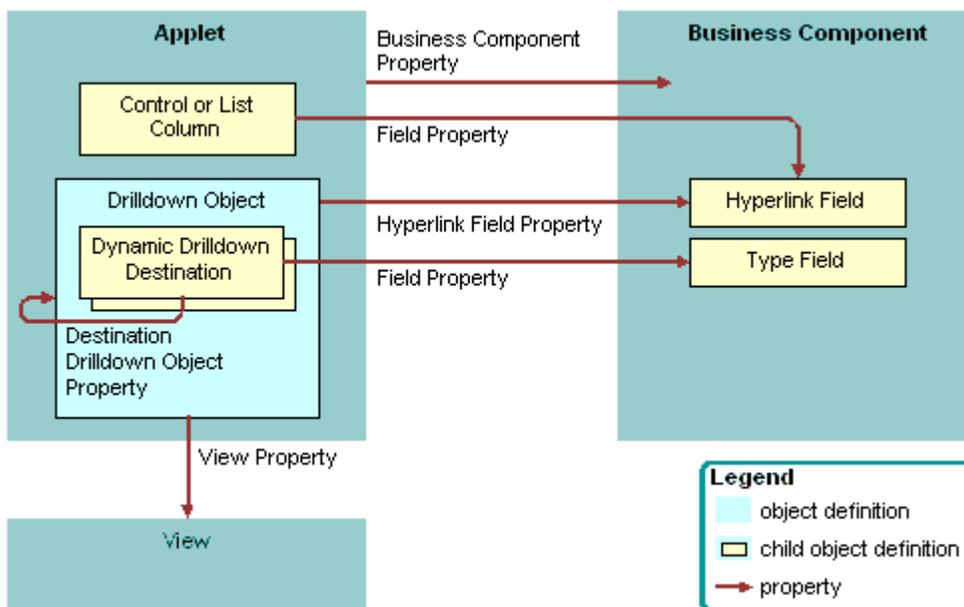


Figure 27. How Siebel CRM Creates a Dynamic Drilldown

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” on page 118](#).
- 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.
- 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” on page 133](#).

- 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” on page 273](#) and [“Improving Performance When Using Applet Toggles” on page 508](#).

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” on page 27](#).
- **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” on page 133](#).
- **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” on page 451](#).

For more information, see [“Creating and Deploying an Application” on page 297](#) and [“How Siebel CRM References Web Pages” on page 149](#).

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

This chapter describes Siebel Web Templates (SWT) and Siebel Web Engine (SWE) tags. It includes the following topics:

- [About Siebel Web Templates on page 145](#)
- [About View Web Templates on page 152](#)
- [About Applet Web Templates on page 154](#)
- [About Siebel Tags on page 168](#)
- [Guidelines for Configuring Siebel Web Templates and Siebel Tags on page 178](#)

For more information, see [“How the Siebel Web Engine Creates a Siebel Application” on page 35](#), and [Chapter 21, “Configuring Siebel Web Templates and Siebel Tags.”](#)

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 on page 145](#)
- [How Siebel CRM References Web Pages on page 149](#)
- [How Siebel CRM Uses HTML Frames in the Container Page on page 150](#)

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” on page 25](#).

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 the following text:

```
swe
```

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” on page 116](#).
- **Format template.** Allows you to create custom HTML types, such as specialized controls, list items, and page items. These templates use the SWF (Siebel Web Format) extension. For more information about SWF files, see [“Configuring an HTML Control Type” on page 483](#) and [“Caution About Using Specialized Classes” on page 74](#).

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

Figure 28 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.

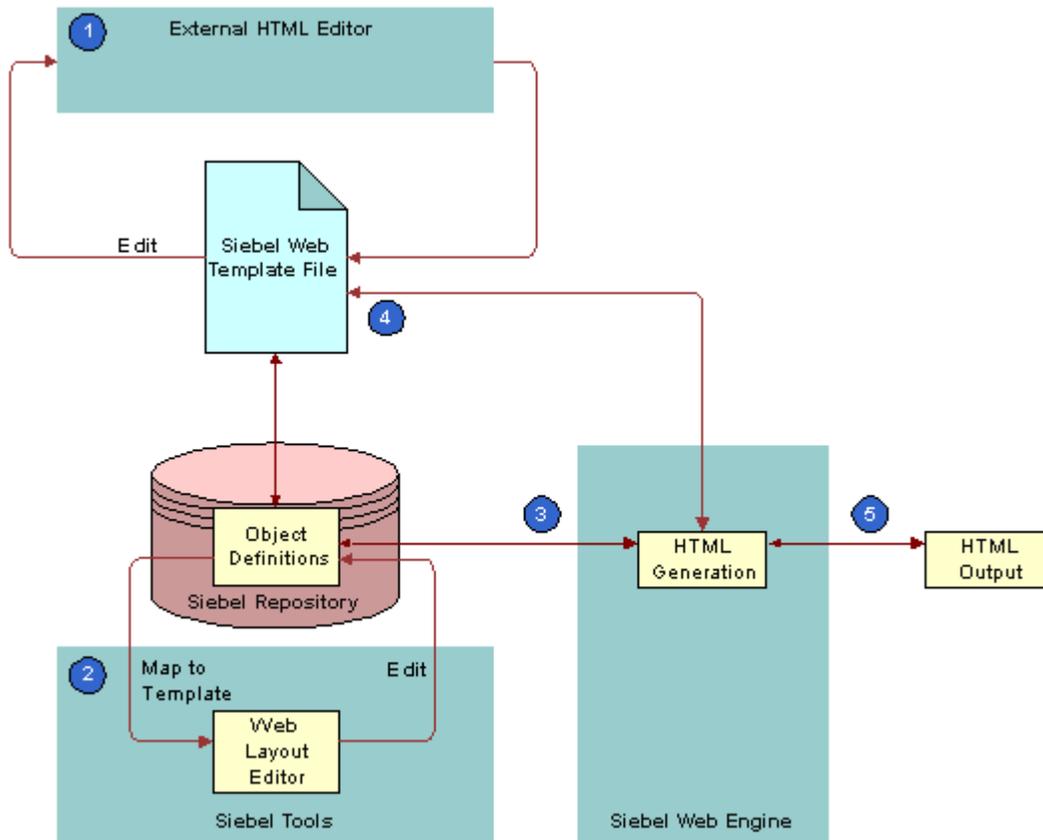


Figure 28. Development Environment You Use to Create HTML Output

Explanation of Callouts

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 SRF file. 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 file 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, then you must use Siebel Tools to define browser script.

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 on page 142](#)
- [Creating and Deploying an Application on page 297](#)
- [Properties of an Application on page 607](#)

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.swt file. 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” on page 180](#).
- **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:
 - swe:screenbar
 - swe:screenlink
 - swe:now-control
- **View bar.** The Siebel Web Engine logic that is associated with the following tags loads the view bar:
 - swe:viewbar
 - swe:viewlink
 - swe:now-control

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” on page 150](#).

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 [“Example Frameset Code From a Container Page” on page 151](#).

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 swe:frameset tag and the swe:frame tag to do the following:

- Create attributes for HTML frames.
- Allow the Siebel Web Engine to control how it targets and refreshes URLs.

SWE Tag That Defines the Set of Frames in a Document

The swe:frameset 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 swe:frame tags.

The swe:frameset tag uses the following format:

```
<swe: frameset htmlAttr="xxx"> ... </swe: frameset>
```

The swe:frameset 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, *' "
```

SWE Tag That Marks the Beginning and End of Content in a Frame

The swe:frame 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 swe:frameset tag.

The swe:frame tag uses the following format:

```
<swe: frame type="xxx" name="yyy"> . . . . </swe: frame>
```

The swe:frame 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 swe:frame tag contains a swe:frameset tag, and the type attribute of the outer swe:frame tag is the following:

page

Example Frameset Code From a Container Page

The following swe:frameset code is from the CCPageContainer.swt container page:

```
<swe: frameset html Attr="rows=' 60, 21, 25, *' border=' 0' frameborder=' No' ">
  <swe: frame type="Page" html Attr="margi nhei ght=' 0' margi nwi dth=' 0' noresi ze
    scrol ling=' No' ">
    <swe: i ncl ude fi le="CCFrameBanner. swt"/>
  </swe: frame>
  <swe: frame type="Screenbar" html Attr="margi nhei ght=' 0' margi nwi dth=' 0' noresi ze
    scrol ling=' No' ">
```

```

    <swe: include file="CCFrameScreenbar.swt" />
</swe: frame>

<swe: frame type="Viewbar" htmlAttr="marginheight='0' marginwidth='0' noresize
scrolling='No' ">
    <swe: include file="CCFrameViewbar.swt" />
</swe: frame>

<swe: frame type="View" htmlAttr="marginheight='0' marginwidth='0' noresize
scrolling='Auto' ">
    <swe: current-view/>
</swe: frame>
</swe: frameset>

```

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 swe:applet 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 code is from an example view web template:

```

<!-- Template Start: CCViewBasic.swt -->
<!------- Page Title ----->
<title>
<swe: this property="Title"/>
</title>

<!------- Salutation applet and Search Applet, table 3.1 ----->
<table border="0" cellpadding="0" cellspacing="1" width="100%">
  <tr>
    <td width="66%"><swe: applet id="101"/>&nbsp;  </td>
    <td width="33%"><swe: applet id="201"/>&nbsp;  </td>
  </tr>
</table>
<!------- End Salutation applet and Search Applet, table 3.1 ----->

```

```

<! ----- Regular Applet(s) ----->
<swe: for-each count=5 iteratorName="currentId" startValue="1">
    <swe: applet id="swe: currentId" />
</swe: for-each>
<! ----- Special Applet(s) ----->
<swe: for-each count=3 iteratorName="currentId" startValue="11">
    <swe: applet id="swe: currentId" />
</swe: for-each>
<!-- Template End: CCViewBasic.swt -->

```

Applet ID Tags

Each `swe:applet id=x` 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 `swe:applet` tags:

- **Tags with IDs of 101 and 201.** Displays the salutation and search applets that Siebel CRM displays at the top 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 bottom 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 on the left and the corresponding list applet occupies the frame on the right in an explorer view.
- If the user does a search. Siebel CRM requires a search frame and a results frame in the right portion of the content area.

For more information, see [“Guidelines for Naming a Siebel Web Template” on page 178](#).

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.swt start -->
<swe: frameset htmlAttr="cols=' 33%, 66%' ' border=' 1' frameborder=' Yes' ">

```

```

<!-- Column 1 Applets -->
<swe: frame type="Applet" htmlAttr="marginheight='0' marginwidth='0'
scrolling='Auto' ">
<swe: for-each count=10 iteratorName="currentId" startValue="101">
  <swe: applet id="swe: currentId" hintText="Applet" var="Parent">
    <!--start applet-->
    <swe: this property="FormattedHtml "/>
    <!--end applet-->
  </swe: applet>
</swe: for-each>
</swe: frame>
<!-- Column 2 Applets -->
<swe: frame type="Applet" htmlAttr="marginheight='0' marginwidth='0'
scrolling='Auto' ">
<swe: for-each count=10 iteratorName="currentId" startValue="201">
  <swe: applet id="swe: currentId" hintText="Applet" var="Parent">
    <!--start applet-->
    <swe: this property="FormattedHtml "/>
    <!--end applet-->
  </swe: applet>
</swe: for-each>
</swe: frame>
</swe: frameset>
<!-- CCView_33_66_Frame.swt end --> </HTML>

```

About Applet Web Templates

This topic describes applet Web templates. It includes the following information:

- [Overview of Applet Web Templates on page 155](#)
- [About Grid Form Applet Templates on page 156](#)
- [About Nongrid Form Applet Templates on page 157](#)
- [About List Applet Templates on page 159](#)

- [About Tree Applet Templates on page 162](#)
- [About Catalog List Applets and Rich List Templates on page 165](#)

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” on page 111](#) and [“Adding a Web Template to an Applet” on page 345](#).

Properties of the Applet Web Template

[Table 27](#) describes properties of the applet web template.

Table 27. Properties of an 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” on page 116 .
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. [Table 28](#) describes properties of the applet web template item.

Table 28. Properties of an 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.

Table 28. Properties of an Applet Web Template Item

Property	Description
Item Identifier	<p>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.</p> <p>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.</p>
Type	<p>Indicates the type of control that the applet web template item defines. You can choose one of the following values:</p> <ul style="list-style-type: none"> ■ Control ■ List Item ■ Web Control

About Grid Form Applet Templates

A grid template simplifies the work of creating the layout of a form applet. Siebel Tools displays a grid layout template in a graphic interface that includes objects that you drag and drop from a palette to a work space. The grid layout applet web template, Siebel tag, and other features in the Web Layout 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” on page 315](#).

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 Siebel 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 Web Layout 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:

- swe:form-applet-layout
- /swe:form-applet-layout

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 Web Layout Editor to configure the layout of a form applet. You must use the Web Layout Editor to modify the layout of an applet that uses a grid applet web template.

About the Body, Header, and Footer

A grid layout applet web template includes a body, header, and footer:

- The swe:form-applet-layout 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 Web Layout 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 Web Layout Editor in Siebel Tools to map controls to any available placeholder. You cannot use the Web Layout 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 swe:form tag. For more information, see [“Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data” on page 116](#).

```
<swe:control id="1100">
  <div class=CmdTxt>
    <swe: this property="FormattedHtml" hintText="Outside Applet Help Text"/>
  </div>
</swe:control >
<table class="AppletStyle1" width="100%" align="center">
  <swe: form>
```

```

<tr>
  <td colspan="2">
    <swe:include file="CCTitle.swt"/>
  </td>
</tr>
<tr>
  <td>
    <swe:error>
      <swe:this property="FormattedHtml"/>
    </swe:error>
  </td>
</tr>
<swe:for-each startValue="1301" count="10" iteratorName="currentId">
  <swe:control id="swe:currentId" hintMapType="FormItem">
    <tr valign="top">
      <td class="scLabelRight">&nbsp;
        <swe:this property="RequiredIndicator"
          hintText="Required"/>
        <swe:this property="DisplayName" hintText="Label"/>
      </td>
      <td class="scField">
        <swe:this property="FormattedHtml" hintText="Field"/>&nbsp;
      </td>
    </tr>
  </swe:control>
</swe:for-each>
</swe:form>
</table>

```

Tags Included in a Nongrid Form Applet Template

This topic describes the some of the tags that a nongrid form applet template includes.

SWE Tag That Accepts User Input

The `swe:form` 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.

SWE Tag That Specifies Placeholders for Controls

The `swe:control` tag specifies placeholders for controls. This tag includes the following attributes:

- **Id.** References the control for the placeholder.
- **Property.** Specifies the property 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.

SWE Tag That Handles Errors

For more information, see [“Configuring How Siebel CRM Displays an Error That Occurs on the Siebel Server” on page 475](#).

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” on page 116](#).

Example Code of a List Applet Template

The following is example code of a list applet template:

```
<table width="100%" cellpadding="0" cellspacing="0" border="0" align="center">
<swe:form>
...
<swe:list>
<!-- List Header Section Start>
<swe:control id="147">
```

```

        <td class="Header" align="center">
        <swe: this property="DisplayName"/>
        </td>
    </swe: control >
    <swe: select-row>
        <td width="42" align="center" class="Header">&nbsp;
        </td>
    </swe: select-row>
<swe: for-each startValue="501" count="20" iteratorName="currentId">
    <swe: control id="swe: currentId">
        <td align="swe: this.TextAlignment" class="Header">
        <swe: this property="ListHeader"/>
        </td>
    </swe: control >
</swe: for-each>
    <swe: control id="142">
        <td class="Header" align="center">
        <swe: this property="DisplayName"/>
        </td>
    </swe: control >
<!-- List Header Section End>
<!------- Loop for all 7 records, List Body ----->
<swe: for-each-row count="7">
<tr class="swe: this.RowStyle">
    <swe: control id="147">
        <td width="42" align="center" class="Row">
        <swe: this property="FormattedHtml" hintMapType="Control"/>
        </td>
    </swe: control >
</swe: select-row>

```

```

        <td width="42" align="center" class="Row">
        <swe: this property="FormattedHtml " />
        </td>
    </swe: select-row>
<!-- ----- List Field Values (501-520) ----->
<swe: for-each startValue="501" count="40" iteratorName="currentId">
    <swe: control id="swe: currentId">
        <td align="swe: this.TextAlignment" class="Row">
        <swe: this property="FormattedHtml "
        hintText="Field"/>
        </td>
    </swe: control >
</swe: for-each>
<!-- ----- Per-record Control Buttons ----->
    <swe: control id="142">
        <td align="center" class="Row">
        <swe: this property="FormattedHtml " hintMapType="Control " />
        </td>
    </swe: control >
</tr>
</swe: for-each-row>
<!-- ----- End Loop, List Body ----->
</swe: list>
...
</swe: form>
</table>

```

Tags That a List Applet Template Includes

This topic describes some of the tags included in a list applet template.

SWE Tag That Encloses an Editable Section

The swe:form tag encloses an editable section. You use it for an editable list applet.

SWE Tag That Encloses the List Header and Body

The swe:list 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 swe:list tags with the specialized list control that supports some capabilities, such as resizing columns. For more information, see [“Caution About Using Specialized Classes” on page 74](#).

SWE Tag That Defines a Placeholder for a List Column

The swe:control 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 swe:control 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="swe: this.TextAlignment">
```

SWE Tag That Repeats for Each List Row

The swe:for-each-row 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” on page 363](#).

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 swe:for-each-node tag.

Siebel CRM uses the swe:for-each-indent 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 `swe:indent-img` 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” on page 467](#).

Example Code of a Tree Applet Template

In this example, if the user clicks or expands a tree, then the `swe:applet-tree-list` 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 on the left and list applet on the right-->
<table border="0" cell spacing="0" cell padding="1" width="100%">
  <tr>
    <!-- Begin Tree Applet -->
    <td>
      <swe:applet id="1" hintText="Tree Applet"/>
    </td>
    <!-- Begin List Applet -->
    <td>
      <swe: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 `swe:node` tag that includes a `DisplayName` type.
- For tree nodes, it ignores the `swe:node` 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>
  <swe:for-each-node>
```

```

<TR VALI GN=top>
<TD NOWRAP>
<swe: for-each-i ndent>
    <swe: i ndent-i mg/>
</swe: for-each-i ndent>
<swe: node type="Di spl ayName">
    <swe: thi s property="FormattedHtml " />
</swe: node>
<swe: node type="Fi el dVal ue">
    <swe: thi s property="FormattedHtml " />
</swe: node>
</TD>
</TR>
</swe: for-each-node>
</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.

SWE Tag That Displays Tree Nodes and Field Values

Siebel CRM uses the `swe:for-each-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 `swe:for-each-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.

SWE Tag That Indents Tree Items

Siebel CRM uses the `swe:for-each-indent` to indent tree items. It iterates through each level of a tree item. The `swe:for-each-indent` tag does not include any attributes.

SWE Tag That Provides a Placeholder for a GIF Image

Siebel CRM uses the swe:indent-img 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 swe:indent-img tag to identify the GIF file. The GIF image can include an empty space or a vertical bar. The swe:indent-img tag does not include any attributes.

SWE Tag That Provides a Placeholder for a Tree Item

Siebel CRM uses the swe:node 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 swe:node tag must use the swe:this tag as a child tag.

The swe:node 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

Figure 29 describes an example of an applet that interweaves records.



Figure 29. Example of an Applet That Interweaves Records

Note how Siebel CRM interweaves records in this example:

- The parent applet provides data for the bullet items under Portable Music.
- The child applet provides data for values that Siebel CRM displays below 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” on page 369](#).

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 swe:applet 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 swe:for-each-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 swe:for-each-child tag uses the following format:

```
<swe: for-each-child> . . . </swe: for-each-child>
```

For more information about the item identifier, see [“Properties of the Applet Web Template Item” on page 155](#).

SWE Tag That Places the Child Applet in the Parent Applet

The swe:child-applet 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 swe:child-applet tag uses the following format:

```
<swe: 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

[Table 29](#) 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” on page 155](#).

Table 29. Properties of the View 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>
<swe: for-each-row>
<tr>
<td>
<swe: control id = "5001" /> <!-- field value like "Small Business" -->
</td>
<td>
<swe: for-each-child>
```

```
<swe: child-applet> <!-- Show the child applet -->
</swe: for-each-child>
</td>
</tr>
</swe: for-each-row>
</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>
<swe: for-each-row>
<td>
<swe: control id="5001"/> <!-- field value like "Desktop" -->
</td>
</swe: for-each-row>
</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 on page 169](#)
- [About Singleton and Multipart Tags, and the This Tag on page 170](#)
- [About Iterator Tags on page 171](#)
- [About Search and Find Tags on page 172](#)
- [About Siebel Conditional Tags on page 175](#)

For more information, see ["Siebel Tag" on page 25](#).

Overview of How Siebel CRM Uses Siebel Tags

Siebel CRM maps a Siebel object to an Id in a web template. The Siebel Web template files do not include references to controls in the Siebel repository. Instead, it includes 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:

```
<swe: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 compiled SRF file 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” on page 155](#).
- 2 To replace the placeholder in the template file, it displays the Formatted HTML representation of this repository object.

Figure 30 describes how the mapping between controls and IDs work to display an image as a link to add a new contact. 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.

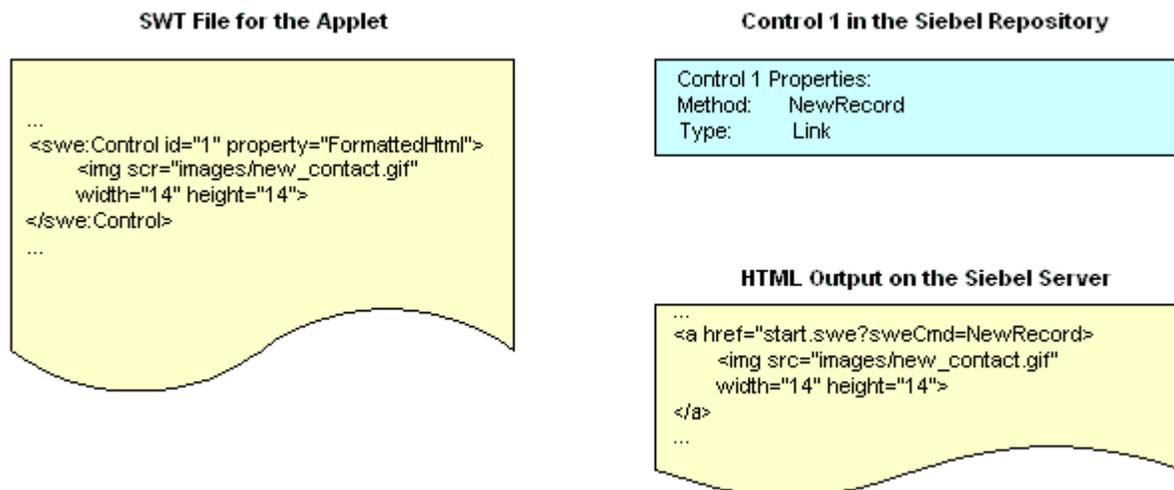


Figure 30. Example of Mappings Between Controls and IDs

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:

```
">
```

You cannot nest some Siebel tags. For example, the following is not valid. It creates an error:

```
<swe: control id="1">  
  <swe: control id="2" property="formattedHTML"/>  
  <swe: this property="formattedHTML"/>  
</swe: control >  
</swe: control >
```

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:

```
<swe: pageitem name="value"/>
```

The following is an example of a *multipart* tag. It does not include a slash at the end of the tag:

```
<swe: control id="1" property="formattedHTML">  
  ...HTML here...  
</swe: control >
```

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:

```
<swe: control id="1">  
  ...HTML here...  
  <swe: this property="formattedHTML"/>  
</swe: control >
```

The `swe:this` tag is an alias for the nearest enclosing Siebel context. You can enclose the `swe:xxx` element to create this context. For example, Siebel CRM commonly includes the `swe:this` tag in a multipart `swe:control` element. In this situation, the `swe:this` 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 `swe:this` element in an applet template file, and if the `swe:this` tag is not in any `swe: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 `swe:for-each` tag to reduce the size of the template files:

```
<swe:for-each count="x" iteratorName="yyyy" startValue="z"/>
```

You can use the following iterator tags:

- `swe:for-each`
- `swe:for-each-row`
- `swe:for-each-child`
- `swe:for-each-node`
- `swe:for-each-indent`
- `swe:for-each-value`

The `swe:for-each` tag includes the following attributes:

- **count.** Specifies the number of times the `swe:for-each` tag must iterate the contents of the `swe:for-each` 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: `swe:iteratorName`.

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 `swe:for-each` 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:

```
swe:CurrentID
```

You can use the swe:CurrentID+x tag to reference a value that is an increment over the current value. The following code is an example of this usage:

```
<swe: for-each startValue="2301" count="50" iteratorName="currentId">
  <swe: control id="swe: currentId">
    .
    .
  </swe: control >
  <swe: control id="swe: currentId+100" />
</swe: for-each>
```

About Search and Find Tags

The Siebel client 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:

- swe:srchCategoryList
- swe:srchCategory
- swe:srchCategoryText
- swe:srchCategoryControl

Example Code for Search and Find Applet Tags

The following code is an example of using search and find applet tags:

```
<swe: srchCateogryLi st>
  <swe: srchCategory>
    <td><swe: srchCategoryText/></td>
    <td><swe: srchCategoryControl /></td>
  </swe: srchCategory>
</swe: srchCategoryLi st>
```

Search Result Applet Tags

You can use the following search results tags to display search and find results in a list applet:

- swe: srchResultFieldList
- swe: srchResultField
- swe: this

Siebel CRM displays search results tags in the Search_ListBodySearchResults.swt template.

Example Code for Search Result Applet Tags

The following code includes an example that uses search result applet tags:

```
<swe: srchResul tFi el dLi st>
    <swe: srchResul tFi el d><td align="swe: thi s. TextAl i gnment" cl ass="Row"><swe: thi s
        property="FormattedHtml " />&nbsp; </td>
    </swe: srchResul tFi el d>
</swe: srchResul tFi el dLi st>
```

Summary of Search, Find, and Search Result Tags

Table 30 describes search, find, and search result tags.

Table 30. Summary of Search, Find, and Search Result Tags

Tag Name	Description
swe: srchCategoryList	<p>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:</p> <ul style="list-style-type: none"> ■ swe: srchCategory ■ swe: srchCategoryText ■ swe: srchCategoryControl <p>It uses the following format:</p> <pre><swe: srchCategoryLi st> ... </swe: srchCategoryLi st></pre>
swe: srchCategory	<p>Search tag that represents a search category object. It encloses the following tags:</p> <ul style="list-style-type: none"> ■ swe: srchCategoryText ■ swe: srchCategoryControl <p>It uses the following format:</p> <pre><swe: srchCategory> ... </swe: srchCategory></pre>

Table 30. Summary of Search, Find, and Search Result Tags

Tag Name	Description
swe:srchCategoryControl	<p>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.</p> <p>It uses the following format:</p> <pre data-bbox="597 562 976 594"><swe: srchCategoryControl /></pre>
swe:srchCategoryText	<p>Search tag that displays the display name of the search category. It must be called in the context of the srchCategory tag.</p> <p>It uses the following format:</p> <pre data-bbox="597 741 935 772"><swe: srchCategoryText/></pre>
swe:srchResultFieldList	<p>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.</p> <p>This tag creates a context and encloses the following tags:</p> <ul style="list-style-type: none"> ■ swe:srchResultField ■ swe:this <p>It uses the following format:</p> <pre data-bbox="597 1129 1382 1161"><swe: srchResul tFi el dLi st> ... </swe: srchResul tFi el dLi st></pre>
swe:srchResultField	<p>Search result tag that represents a result field object. Siebel CRM must call it in the context of the srchResultFieldList tag. It encloses the swe:this tag.</p> <p>It uses the following format:</p> <pre data-bbox="597 1339 1263 1371"><swe: srchResul tFi el d> ... </swe: srchResul tFi el d></pre>
swe:this	<p>A search result tag. Depending on the value of the property attribute, it does one of the following:</p> <ul style="list-style-type: none"> ■ Property is TextAlignment. Gets the text alignment property for the result field from the following object: <pre data-bbox="646 1549 1133 1581">Search Definition - Custom result Field</pre> ■ 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. <p>It uses the following format:</p> <pre data-bbox="597 1759 760 1791"><swe: thi s/></pre>

About Siebel Conditional Tags

This topic describes Siebel Web Engine conditional tags. For more information, see *Siebel Developer's Reference*.

If Conditional Tag

The swe:if tag provides a simple conditional branching capability. It uses the following format:

```
<swe:if condition="xxx"> ... </swe:if>
```

The swe:if tag includes the Condition attribute. Siebel CRM does the following:

- If the condition is TRUE, then Siebel CRM processes the body of the swe:if tag.
- If the condition is FALSE, then Siebel CRM skips the body of the swe:if tag.

The swe:if tag does not provide an else capability. To implement an else condition, you can use some combination of the swe:switch, swe:case, and swe: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:

- swe:switch
- swe:case
- swe:default

The swe:switch tag is a container tag for the swe:case and swe:default tags.

Format for the Switch, Case, and Default Conditional Tags

The swe:switch, swe:case, and swe:default tags use the following format:

```
<swe:switch>
  <swe:case condition="xxx">
    ...
  </swe:case>
  <swe:case condition="yyy">
    ...
  </swe:case>
  <swe:default>
    ...
</swe:switch>
```

</swe: switch>

Attributes for the Switch, Case, and Default Conditional Tags

The swe:case tag includes the Condition attribute. The swe:switch and swe:default tags include no attributes. To process these tags, Siebel CRM does the following:

- Ignores any tags that exist in the body of the swe:switch tag that are not the swe:case tag or swe:default tag.
- Examines the swe:case tags, starting with the first swe:case tag, and then does one of the following:
 - If any of the swe:case tags satisfy the condition, then Siebel CRM skips any other swe:case tags and swe:default tags, and then processes the body of the swe:case tag that satisfies the condition.
 - If none of the swe:case tags satisfy their conditions, then Siebel CRM processes the body of the swe:default tag. You must make sure that the body of a swe:switch tag contains only a swe:default tag.

Variable Conditional Tag

An applet template includes the swe:if-var tag. It conditionally express the body of the swe:if-var tag as determined by a variable that is set in a parent view web template. For the purposes of the swe:if-var 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 swe:if-var tag in the child applet template can evaluate.

The expression in the swe:if-var tag returns a value of true or false depending on if the variable it evaluates is a property of the swe:applet 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.

Figure 31 describes object relationships that Siebel CRM uses with the variable conditional tag.

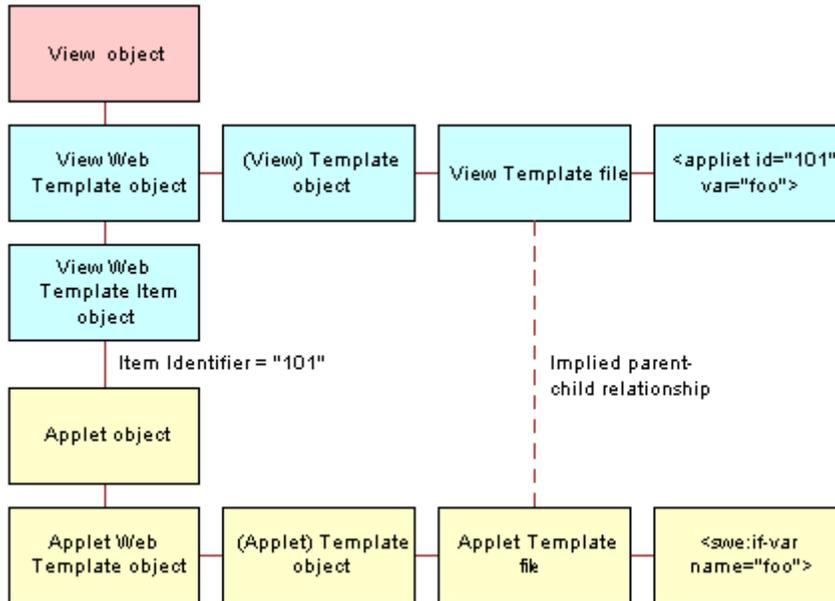


Figure 31. Object Relationships Siebel CRM Uses with the Variable Conditional Tag

Example Code That Uses the Variable Conditional Tag

In this example, a view uses a template that contains the following tags:

```
<swe:applet hi ntMapType="Appl et" i d="1" property="FormattedHtml " hi ntText="Appl et"
var="Parent"/>
```

```
<swe:applet hi ntMapType="Appl et" i d="2" property="FormattedHtml "
hi ntText="Appl et" var="Chi ld"/>
```

The view object references an applet through a view web template item. The template for this applet includes the following tags:

```
<swe:if-var name="Parent">
  <td val i gn="mi ddl e" nowrap>
    <swe:menu type="Button" bi tmap="MenuBttn" wi dth="38" hei ght="15"
    bgcol or="gray" fgcol or="bl ue"/>
  </td>
</swe:if-var>
<swe:if-var name="Chi ld">
  <td val i gn="mi ddl e" nowrap>
    <swe:menu type="Button" bi tmap="MenuBttn" wi dth="38" hei ght="15"
    bgcol or="gray" fgcol or="red"/>
  </td>
</swe:if-var>
```

```
</td>  
  
</swe:if-var>
```

If you drag and drop the applet into the placeholder in the view web template, and if the applet Id for this placeholder is:

- 1. The first swe:if-var condition returns TRUE and the second condition returns FALSE. This occurs because the var property of the swe:applet 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 Naming a Siebel Web Template

The file name of a Siebel Web template uses the SWT extension. For example, CCPageContainer.swt, CCHomePageView.swt, and so on. You are not required to use this format. The Siebel Web Engine recognizes and interprets the file correctly regardless of how you name it. It is recommended that you use swt for the file extension to help develop and maintain Siebel CRM. The swt extension allows you to immediately recognize how Siebel CRM uses the file.

Siebel CRM typically stores Siebel Web template files in the Web Template directory in your Tools installation directory. The Filename property references a web template.

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” on page 116.](#)

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 `swe:applet` tag in the frame to an applet in the Siebel repository. Otherwise, empty frames will occur.
- If a `swe:frame` block contains a `swe:applet` 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 `swe:include` tag to define the contents of a frame. You can place the contents directly into the body of the `swe:frame` tag.
- Make sure the contents of the `swe:frame` 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 `swe:frame` tag contains only the `swe:current-view/` 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 Web Server Extension installation directory:

`SWSE_ROOT\PUBLIC\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” on page 26](#) and *Siebel Developer’s Reference*.

Guidelines for Modifying a Predefined Query

If you modify a predefined query, then use the following guidelines:

- The swe:pdqbar 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 swe:pdqbar tag
 - The Favorites label that Siebel CRM displays to the left of the swe:pdqbar 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 swe:pdqbar tag in the view frame.
- If Siebel CRM uses HTML frames, then you must place the swe:pdqbar 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.

Table 31 describes the query management commands that you can use.

Table 31. Query Management Commands

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.

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

This chapter describes an overview of configuring a Siebel application. It includes the following topics:

- [About Configuring a Siebel Application on page 183](#)
- [Roadmap for Configuring a Siebel Application on page 184](#)
- [Developing an Implementation Plan on page 184](#)
- [Using Development Tools and Setting Up the Development Environment on page 187](#)

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.

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 on page 184](#)
- 2 [Using Development Tools and Setting Up the Development Environment on page 187](#)
In particular, see [Setting Up a Developer with a Local Database on page 193](#).
- 3 [Process of Determining Whether You Can Reuse a Predefined Object on page 213](#)
- 4 [Optional. Process of Creating and Binding an Entity Relationship Diagram on page 223](#)
- 5 [Configuring Tables on page 233](#)
- 6 [Configuring Views, Screens, and Applications on page 267](#)
- 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:

- [Chapter 17, “Configuring Special-Purpose Applets”](#)
 - [Chapter 21, “Configuring Siebel Web Templates and Siebel Tags”](#)
 - [Chapter 25, “Localizing Siebel Business Applications”](#)
- 8 [Improving the Performance of Siebel Business Applications on page 489](#)

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 on page 185](#)
- [Developing a Plan to Control File Versions for the Physical User Interface Layer on page 187](#)

This task is a step in [“Roadmap for Configuring a Siebel Application” on page 184](#).

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” on page 221](#).
 - 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” on page 36](#).
 - 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” on page 189](#).
 - 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” on page 187](#).
 - 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” on page 185](#).
- 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 [Chapter 10, “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 bottom 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 [Figure 1 on page 24](#).

- 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.

Use a web template. It 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” on page 25](#).

The description in this topic is appropriate for most projects. For more information, see [“Getting Help From Oracle” on page 23](#).

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 Web Server environment.
- 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” on page 184](#).

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 on page 188](#)
- [Guidelines for Developing a Siebel Application on page 189](#)
- [Setting Up the Configuration File for Siebel Tools on page 191](#)
- [Displaying Object Types You Use to Configure Siebel CRM on page 192](#)

- [Setting Up a Developer with a Local Database on page 193](#)
- [Creating Scripts to Configure Siebel CRM on page 197](#)

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” on page 187](#) 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.
For example, the system and database environment and developer workstations. For more information, see [“Setting Up a Developer with a Local Database” on page 193](#).
- 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** Compile your Siebel application 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 [Chapter 25, “Localizing Siebel Business Applications.”](#)
- 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 [Chapter 25, “Localizing Siebel Business Applications.”](#)

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.

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 on page 42](#)
- [Guidelines for Naming a Business Component on page 76](#)
- [Guidelines for Naming an Applet on page 126](#)
- [Guidelines for Creating a Join on page 94](#)
- [Guidelines for Creating a Business Object on page 109](#)

- [Guidelines for Creating an Applet on page 125](#)
- [Guidelines for Naming a View on page 137](#)
- [How Siebel CRM References Web Pages on page 149](#)
- [Reusing Predefined Objects on page 203](#)

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” on page 204](#).

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 Technical Services. Any file that contains one of the following extensions and is part of the Siebel installation is a Siebel configuration file:

- .cfg

- css
- gif
- swt
- htm
- xsl
- sws
- swf
- sw
- ctl
- sql
- ucf
- rox
- rod
- srf
- 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 tool s.cfg file in a text editor.
- 2 Set the EnableToolsConstrain parameter to FALSE.
For more information, see [“How the EnableToolsConstrain Parameter Affects Text Strings” on page 192](#).
- 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

Table 32 describes how the EnableToolsConstrain parameter affects text strings. For more information, see *Using Siebel Tools*.

Table 32. How the EnableToolsConstrain Parameter Affects Text Strings

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” on page 253](#).
- 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” on page 338](#).

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 black check mark with a white background for the tree.

- 6 Repeat [Step 4](#) 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.

Setting Up a Developer with a Local Database

After you install Siebel Tools, the Siebel Server, and any other software in the development environment, you must set up each developer with a local database. Doing this provides the developer with a local copy of the Siebel database, including the Siebel repository, on the developer's local computer. The developer can check out objects from the Siebel repository, configure and test their work on their local computer, and then check objects back into the Siebel repository. (If you use the Workspaces feature of Siebel Tools, then the local database is not used.)

NOTE: For Siebel Innovation Pack 2016, the local database for Siebel Tools and for Siebel Remote uses Oracle Database XE. SAP SQL Anywhere is no longer available.

Oracle Database XE is always installed with Siebel Tools. The data source for the local database in either case is LOCAL_XE. During installation, you must also specify the location of the 32-bit Oracle Database Client. As described in the *Siebel Installation Guide* for the operating system you are using, you have two main choices with respect to the local database when you install Siebel Tools:

- You can install Siebel Tools without the local database and then set up the developer as a remote user. If the database has been extracted for this user, then the developer can initialize it. This is the traditional method of setting up a developer with a local database. For more information, see ["Setting Up the Developer as a Remote User" on page 194](#).
- You can install Siebel Tools with the local database, and then run a script to set up the local database. This local database cannot be used with Siebel Mobile Web Client. This is a new method of setting up a developer with a local database. For more information, see ["Installing and Setting Up the Local Database" on page 195](#). (Alternatively, you can install Siebel Tools without the local database and install it later.)

For more information about:

- Installing Siebel Tools, see the *Siebel Installation Guide* for the operating system you are using. This guide also provides information about installing Oracle Database XE with Siebel Tools, about requirements such as an existing installation of the 32-bit Oracle Database Client, and about validating the installation of Siebel Tools and Oracle Database XE.
- Extracting a local database and setting up a remote user, see *Siebel Remote and Replication Manager Administration Guide*.
- Checking in and checking out projects, or using the Workspaces feature, see *Using Siebel Tools*.

Setting Up the Developer as a Remote User

This topic describes setting up the developer as a remote user. This procedure uses the traditional method of setting up a developer with a local database. Alternatively, you can use the method described in [“Installing and Setting Up the Local Database” on page 195](#).

This topic is part of [“Setting Up a Developer with a Local Database” on page 193](#).

To set up a developer as a remote user

- 1 Install Siebel Tools on the computer that the developer uses.

You must install Siebel Tools in a directory that is separate from the Siebel Mobile Web Client. For example, if you install the Siebel client in `c:\siebel\client`, then install Siebel Tools in `c:\siebel\tools`. This task does the following:

- Makes sure that the development environment and the Siebel client environment are distinct.
- Makes sure that, if you use Siebel Remote in these environments, then the two installations do not conflict.

For more information, see the *Siebel Installation Guide* for the operating system you are using.

NOTE: For this procedure, you do not install the local database when you install Siebel Tools. For [“Installing and Setting Up the Local Database” on page 195](#), however, you do install the local database with Siebel Tools.

- 2 Verify that each developer possesses a valid user name and password for the Siebel development database server.

In most situations, Siebel CRM uses the employee login ID and password for the database server user name and password.

- 3 Using a Siebel client connected to the development server database, create an Employee record and a Mobile User record for each developer.

Use the first and last name of the developer for the employee first and last names. For the login ID, use a consistent naming format, such as first initial and last name, that will simplify identifying who locked a project.

NOTE: Password encryption interferes with project check in and check out. If you check projects in and out, then you must disable password encryption in the client or in the configuration file if you run Siebel Tools. For more information about password encryption, see *Siebel Security Guide*.

For more information about creating an employee record, see *Siebel Security Guide*.

- 4 In the Siebel database, grant each developer the Developer and Siebel administrator responsibilities.

To prevent unintended modifications to important system preferences, you can create a responsibility that contains access to all views except the System, Service, and Marketing Administration views. You can use a common position for all developers. For testing purposes, you can set up an organization structure that models the structure of your business.

If you do not grant the user the Developer responsibility, then Siebel Tools does not activate drilldowns in the Siebel Tools client.

For more information about setting up an employee, see *Siebel Applications Administration Guide*. For more information about setting up responsibilities and positions, see *Siebel Security Guide*.

- 5 On the Siebel Server, run the Generate New Database component to create the database, and then use the Database Extract server component to extract the local database for each developer, as described in *Siebel Remote and Replication Manager Administration Guide*.

Database Extract creates a template for the local database that includes only business data, not repository data. This server component extracts enterprise data into this template, together with any data that possesses limited visibility, such as contacts, accounts, opportunities, and so on. The developer must add repository data to the local database in [Step 9 on page 195](#).

- 6 To initialize the local database for the developer, open Siebel Tools and connect to the local database using the LOCAL_XE data source.

Siebel Tools displays a message that states that the local database is not found.

- 7 To start the initialization process, click Yes.

- 8 In the Siebel Remote Parameters dialog box, enter the Siebel developer login that you defined in [Step 3 on page 194](#) and an appropriate password.

After the local database has been initialized, the developer must add repository data to the local database, as described in the next step.

- 9 Do an initial get operation for all repository projects on the local database.

For more information about this task, see *Using Siebel Tools*. You have finished setting up the local database for this developer. The final step in this procedure applies only if you installed the local database with Siebel Tools.

Installing and Setting Up the Local Database

This topic describes installing the local database with Siebel Tools and setting up this database. This procedure uses a new method of setting up a developer with a local database, which is an alternative to the method described in [“Setting Up the Developer as a Remote User” on page 194](#).

This topic is part of [“Setting Up a Developer with a Local Database” on page 193](#).

To install and set up the local database

- 1 Follow [Step 1 on page 194](#) through [Step 4](#), in “[Setting Up the Developer as a Remote User](#)” on [page 194](#). However, in this procedure, you install the local database with Siebel Tools.

For more information, see the *Siebel Installation Guide* for the operating system you are using.

- 2 On the Siebel Tools computer, run the local database setup script, using a command similar to the following.

This command reads custom schema and data from the enterprise database and applies it to the existing local database that was installed with Siebel Tools. Enclose argument values in double quotes where the value has a space in it.

```
SIEBEL_TOOLS_ROOT\bin\LocalDbSetup.exe -s $SiebelTools -o $ODBCDataSource -t $TableOwner -u $username -p $password -r "Siebel Repository" -d $ODBCDataSource_XE -x $OracleXEInstallationPath -l $logfile -w $debugmode -y $TableTypes -z $StatisticsUpdation
```

where:

- \$SiebelTools is the path of the Siebel Tools installation.
- \$ODBCDataSource is the ODBC entry of the enterprise database.
- \$TableOwner is the table owner of the enterprise database.
- \$username is a user name for the enterprise database.
- \$password is the password for this user name.
- \$ODBCDataSource_XE is the ODBC entry for Oracle Database XE (LOCAL_XE).
- \$OracleXEInstallationPath is the Oracle Database XE installation path, such as *SIEBEL_TOOLS_ROOT\oracle*.
- \$logfile is the name of the log file.
- \$debugmode specifies the debug mode. The default is N. SQL and data files are not deleted after this script runs. You can set this to Y to analyze the issue in case of any error.
- \$TableTypes are the types of tables for which data is to be applied. The default value is RS, which applies repository and seed data. Supported values are: (D)ata|(D)ictionary|(R)epository|(S)eed|(L)og|(C)omposer|(E)xtension. Use one or more values as appropriate for data tables, dictionary tables, repository tables, seed data tables, logic tables, tables for Siebel Composer, or extension tables.
- \$StatisticsUpdation specifies whether statistics are updated. The default is Y.

After the script finishes running, you have finished setting up the local database for this developer.

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 on page 197](#)
- [Scripts That You Write for the Browser on page 198](#)
- [Creating and Deploying a Browser Script on page 200](#)

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.
- Develop a routine that opens an external application on the Siebel Server in reply to a Siebel event or to send start-up parameters. This capability is valid for browser script only.

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 Layout 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.

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

Figure 32 describes the hierarchy of relationships between object types that Siebel CRM can use with a 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.

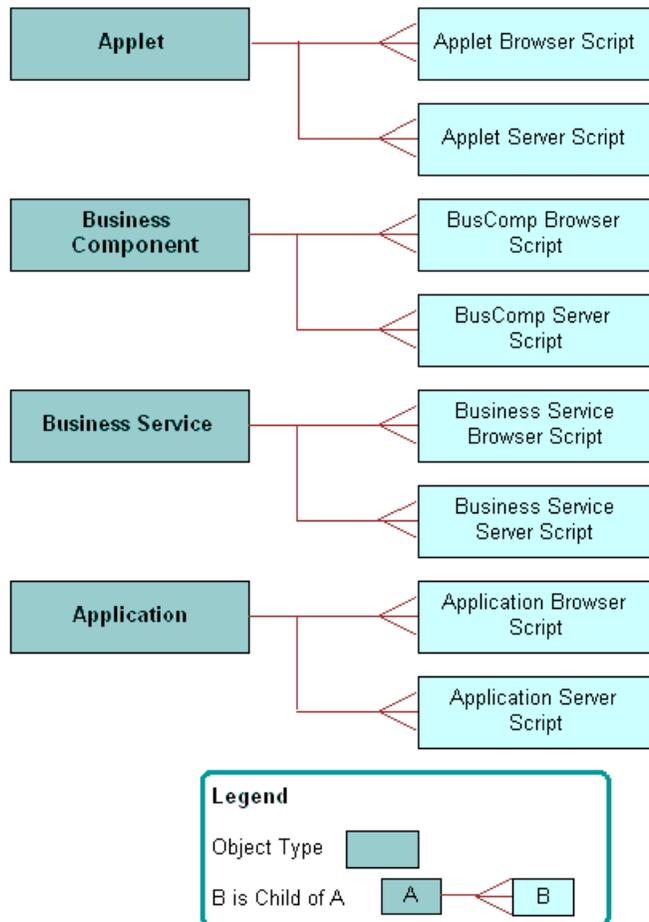


Figure 32. Hierarchy of Relationships Between Object Types That Siebel CRM Uses with a Script

Figure 33 includes an example of how you can use a browser script to validate the field in a form that Siebel CRM displays in the Siebel client. This example uses browser script on the BusComp_PreSetFieldValue event handler in the Account business component.

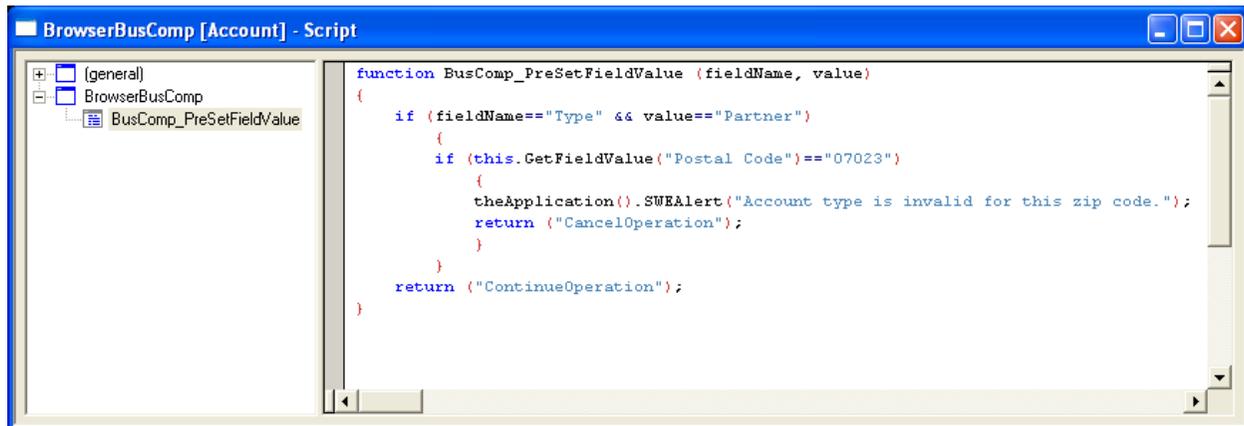


Figure 33. Example of Browser Script to Validate a Field

Creating and Deploying a Browser Script

You can create and deploy a JavaScript file (.js).

To create and deploy a browser script

1 Use one of the following configurations to create the browser script:

- In Siebel Tools, compile objects to a repository file.

If you compile objects to a repository file, then Siebel Tools only creates browser scripts for compiled objects. Siebel Tools places them in the directory that you define in the Scripting tab of the Development Tools Options dialog box. To view this dialog box, choose the View menu, and then the Options menu item.

If you do not create a directory, then Siebel Tools stores the browser script in the following directory:

```
SIEBEL_TOOLS_ROOT\publi c\l anguage_code\srf_ timestamp\bscripts\al l
```

where:

- *SIEBEL_TOOLS_ROOT* is the root directory where you installed Siebel Tools.
- *language_code* is the language code, such as ENU.
- *srf_timestamp* is the timestamp when the Siebel Repository File (SRF) was saved.

For more information, see *Using Siebel Tools*.

- In the command-line interface, run the `genbscript.exe` utility.

If you run the `genbscript.exe` utility, then it creates all browser scripts that exist in the Siebel repository and places them in a directory that you define. You run this utility from the `SIEBEL_TOOLS_ROOT/bin` directory. Use the following format to run `genbscript`:

```
genbscript config_file destination_directory language_code
```

where:

- *config_file* is the name of the configuration file.
- *destination_directory* is the destination directory where `genbscript` stores the script files.
- *language_code* is the language code, such as ENU. The language code parameter is optional for ENU, but you must define it for other languages.

For example:

```
genbscript c:\siebel\client\bin\enu\agent.cfg c:\siebel\client\publi c\enu  
enu
```

- 2 Stop, and then restart, the Web server.

This configuration loads the scripts into the Siebel Web Server Extension (SWSE) and avoids an Object Not Found error message.

- 3 Deploy the browser script to the following directory on the SWSE:

```
SWSE_ROOT\publi c\language_code
```

- 4 Deploy the browser script to the following directory on the Siebel Mobile Web Client, as necessary:

```
SIEBEL_CLIENT_ROOT\publi c\language_code
```

- 5 If you migrate scripts from one location to another location, then you must copy the following directories to the correct location (from `SIEBEL_TOOLS_ROOT\publi c\language_code`, for example):

```
srf_timestamp\bscripts\all
```

If you create a browser script, then the Siebel software creates a directory path and names it according to the version of the Siebel repository file. The system appends it to the path that you create as the destination directory. For example, after you compile or migrate browser scripts to the SWSE, the Siebel software uses the following path to the browser script files:

```
SWSE_ROOT\publi c\language_code\srf_timestamp\bscripts\all
```

- 6 If you compile on a Siebel Server that runs in the Windows operating system, and then migrate browser scripts to a Siebel Server that runs on an Oracle Solaris or AIX operating system, then you must FTP the directories to the correct location on the Oracle Solaris or AIX computer.

10 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 on page 203](#)
- [Guidelines for Reusing a Predefined Object on page 204](#)
- [Process of Determining Whether You Can Reuse a Predefined Object on page 213](#)

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 on page 204](#)
- [Guidelines for Reusing a Predefined Table on page 205](#)
- [Guidelines for Reusing a Predefined Business Object on page 209](#)
- [Guidelines for Reusing an Applet on page 209](#)
- [Guidelines for Reusing a Predefined View on page 210](#)
- [Guidelines for Reusing a Predefined User Interface Object on page 210](#)
- [Reasons Not to Reuse a Predefined Object on page 211](#)

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.
- The Siebel Repository File is smaller and compiles faster.
- 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 [Chapter 24, "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" on page 118.](#)

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 [Chapter 10, "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 displaying 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” on page 118](#).

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” on page 55](#).

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. Rare instances exist when you must create a new one-to-one table, such as to add a LONG column because you cannot add it to a base table and because Siebel CRM supports only one LONG column on a table. For more information, see [“Creating a LONG Column on an Extension Table” on page 237](#).

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” on page 43](#).

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 [Chapter 10, “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” on page 43](#).

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” on page 58](#).

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” on page 76](#).

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” on page 118](#).

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” on page 190](#).

For more information, see the following topics:

- [About Applets, Controls and List Columns on page 111](#)
- [Guidelines for Creating an Applet on page 125](#)

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” on page 190](#).

For more information, see the following topics:

- [About Views on page 130](#)
- [Guidelines for Creating a View on page 136](#)

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” on page 190](#).

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” on page 118](#).
- 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” on page 74](#).

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” on page 118](#) and [“Caution About Using Specialized Classes” on page 74](#).
- 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"](#) on page 184.

To determine whether you can or cannot reuse a predefined object, do the following tasks:

- 1 [Determining Functional Fit for Reusing a Predefined Object](#) on page 213
- 2 [Determining Technical Fit for Reusing a Predefined Object](#) on page 214
- 3 (Conditional) [Determining Whether You Can Reuse a Predefined Table Column](#) on page 215
- 4 (Conditional) [Determining Whether You Can Reuse a Predefined Business Component Field](#) on page 217
- 5 (Conditional) [Determining Whether You Can Reuse a Predefined Business Component](#) on page 219

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"](#) on page 59.

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"](#) on page 213.

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"](#) on page 203.
- 2 Determine whether you must copy the object rather than reuse the object.
For more information, see ["Guidelines for Reusing a Predefined Object"](#) on page 204.
- 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”](#) on page 213.

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 [Chapter 24, “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” on page 213](#).

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” on page 59](#).

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 [Chapter 24, “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” on page 192](#).

If you are in doubt about how a column might affect Siebel Remote, then see [“Getting Help From Oracle” on page 23](#). 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” on page 416](#).

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” on page 217](#).

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” on page 98](#).
 - 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” on page 60](#).

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”](#) on page 57.

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”](#) on page 213.

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”](#) on page 213.

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”](#) on page 98.

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” on page 219](#).

- 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” on page 504](#).

- 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” on page 74](#).

- 4 Use a user key field.

For more information, see [Step 8 on page 216](#).

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 page 74](#).
- 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” on page 118](#).

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” on page 77](#) and [“Caution About Using Specialized Classes” on page 74](#).

- 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

This chapter describes how to use the Entity Relationship Designer. It includes the following topics:

- [About the Entity Relationship Designer on page 221](#)
- [Process of Creating and Binding an Entity Relationship Diagram on page 223](#)
- [Opening or Modifying an Entity Relationship Diagram on page 226](#)
- [Modifying Shapes and Lines in the Entity Relationship Designer on page 228](#)

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:

- A drag-and-drop environment to create an ERD.
- 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 diagramming 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

Figure 34 describes an example entity relationship diagram that contains two entities and one relationship.



Figure 34. Example of an Entity Relationship Diagram

Table 33 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.

Table 33. Example of How the Entity Relationship Designer Filters Business Components

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	<p>The following business components are available for binding:</p> <ul style="list-style-type: none"> ■ 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	<p>The following business components are available for binding:</p> <ul style="list-style-type: none"> ■ 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

Figure 34 on page 221 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.

Table 34 describes the links and joins that are available to bind to relationship AC.

Table 34. Example of How the Entity Relationship Designer Filters Links and Joins

Relationship AC	Objects That Are Available to Bind
one-to-many	<p>The following objects are available to bind:</p> <ul style="list-style-type: none"> ■ 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	<p>The following objects are available to bind:</p> <ul style="list-style-type: none"> ■ 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”](#) on page 184.

To create and bind an ERD, perform the following tasks:

- 1 [Creating an Entity Relationship Diagram on page 224](#)
- 2 [Binding an Entity to a Business Component on page 224](#)
- 3 [Associating an Entity Attribute with a Business Component Field on page 225](#)
- 4 [Binding a Relationship to a Link or Join on page 225](#)

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”](#) on page 223.

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”](#) on page 192.
- 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 Drag and drop 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 Drag and drop 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 on the left and the subsequent entity placed on the right. 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”](#) on page 223.

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”](#) on page 221.

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” on page 245](#).

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” on page 223](#).

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” on page 223](#).

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” on page 222](#), and [“About Links” on page 104](#).

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 on page 226](#)
- [Viewing the Entities and Relations Lists of an ERD on page 226](#)
- [Modifying the Properties of a Relationship on page 227](#)
- [Copying the Drawing of an Entity Relationship Diagram on page 228](#)

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 drag 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” on page 226](#).

- 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 on page 228](#)
- [Modifying Relationships in the Entity Relationship Designer on page 229](#)
- [Moving Shapes in the Entity Relationship Designer on page 230](#)
- [Resizing Shapes in the Entity Relationship Designer on page 231](#)
- [Zooming, Displaying, and Snapping the Grid on page 231](#)

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 [Step 2](#).
 - 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 [Step 2](#).
 - 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 drag it to a new position on the canvas.
You can modify the shape of a relationship. For example, you can add a right 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 drag and drop 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:
 - Drag 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 drag 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 drag 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

This chapter describes tasks you perform to configure tables. It includes the following topics:

- [Using the New Table Wizard to Create a New Table on page 233](#)
- [Creating a Custom Index on page 235](#)
- [Adding an Extension Column to a Base Table on page 236](#)
- [Configuring Objects to Use a One-To-Many Extension Table on page 237](#)
- [Configuring an Extension Table on page 237](#)
- [Applying a Custom Table to the Local Database on page 240](#)
- [Applying a Data Layer Customization to the Server Database on page 242](#)
- [Downloading a Data Layer Customization to Remote Users on page 243](#)

For more information, see [“Options to Configure the Data Objects Layer” on page 59](#).

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” on page 63](#).

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 Compile and test your modifications.

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” on page 64](#).

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” on page 57](#) 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” on page 57](#).

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” on page 63](#).

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” on page 511](#).

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.

For more information, see [“Applying a Custom Table to the Local Database” on page 240](#).

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” on page 118](#).

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:

- [Creating a LONG Column on an Extension Table on page 237](#)
- [Manually Creating a One-to-One Extension Table on page 238](#)
- [Modifying a Custom Extension Table or Column on page 238](#)
- [Deleting a Custom Extension Table or Column on page 239](#)

Creating a LONG Column on an Extension Table

You can create a LONG extension column. For more information, see [“Guidelines for Creating a LONG Column” on page 64](#).

To create a LONG column on an extension table

- 1 Locate an appropriate one-to-one extension table that corresponds to the base table that requires the LONG column.
The S_EVT_ACT_X table is an example of a one-to-one extension table for the S_EVT_ACT table.
- 2 Create a column in the table.
- 3 Set the Physical Type property of the column to Long and the Length property 0.
For more information, see [“Adding an Extension Column to a Base Table” on page 236](#).
- 4 Apply the modifications to your local database.
For more information, see [“Applying a Custom Table to the Local Database” on page 240](#).

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” on page 66](#) and [“Using the New Table Wizard to Create a New Table” on page 233](#).

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” on page 43](#).

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 in [Step a](#).
- 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 Custom Table to the Local Database

After you configure a table you must apply your configurations to the local database. Note that Siebel CRM does not support a custom database trigger. If you create a custom trigger on a Siebel base table, then you must disable it before you update the logical database schema. You must redefine the trigger after you finish the update.

This topic describes how to apply a custom table to the local database for databases other than IBM DB2 for z/OS. If you use IBM DB2 for z/OS, then see *Implementing Siebel Business Applications on DB2 for z/OS*.

To apply a custom table to the local database

- 1 Make sure no antivirus software is running on the computer that you use to run Siebel Tools.
If antivirus software is running, and if you click the Apply/DDDL button, then Siebel Tools might stop responding.
- 2 In the Object Explorer, click Table.
- 3 In the Tables list, locate the table that includes your customization.

- 4 In the Tables list, click Apply/DDL.

Siebel Tools disables the Apply/DDL button for tables whose Type property is External. For more information, see *Overview: Siebel Enterprise Application Integration*.

- 5 In the Choose Option dialog box, choose one of the following, and then click OK:

- **Apply.**
- **Generate DDL.** Creates a DDL file instead of applying the customization.

- 6 In the Warning dialog box, click OK.

- 7 In the Apply Schema dialog box, complete the fields using values from the following table.

Field	Description
Tables	Choose one of the following options: <ul style="list-style-type: none"> ■ All. Update the local database to reflect all modifications that Siebel Tools makes to the dictionary. If you choose this option, then Siebel Tools compares each database object with the data dictionary and updates the dictionary, if necessary. ■ Current Query. Update the local database to reflect modifications Siebel Tools makes to the tables in the current query only. ■ Current Row. Update the local database to reflect modifications Siebel Tools makes to the table in the current row only.
Table space	Leave each of these fields empty. These fields are specific to IBM DB2 for z/OS.
16K table space	
32K table space	
Index space	
Storage control file	Optional. The DBA provides this file. It is specific to your database.
Database user	Enter the database user ID of the table owner, which is typically SIEBEL. Siebel Tools reads the table owner from the configuration file for Siebel Tools.

Field	Description
Database user password	<p>Enter your user password for the local database.</p> <p>The password is case sensitive. All characters of the password must be in uppercase. For example, assume the remote user JSMITH initialized the local database and used ROver12 as the password. If this user applies a schema modification to the local database, then this user must use SIEBEL as the Privileged User Id and ROver12 as the Password.</p> <p>When Siebel CRM initializes the local database for a remote client, Siebel CRM modifies the table owner password to the password of the remote user. In this situation, use the password of the remote user in the password field.</p>
ODBC data source	<p>Verify that the ODBC connection in the ODBC Data Source text box is correct for your environment.</p> <p>You cannot apply a schema modification to any database other than the database to which you are currently connected. You cannot define the ODBC name of a different database.</p>

- 8 Click Apply.
Your custom extension table and columns are now available to use in your configuration.
- 9 Optional. To activate a customization on an Enterprise Integration Manager table, choose the appropriate table, and then click Activate.
- 10 Test the customization.

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 on page 241](#).
 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” on page 243](#).
 - 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: For Siebel Innovation Pack 2016, the local database for Siebel Tools or for Siebel Mobile Web Client uses Oracle Database XE. SAP SQL Anywhere is no longer available or supported.

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*.
- 7 If you do not use Siebel Anywhere, then manually reextract and reinitialize all remote user databases.

13

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 on page 245](#)
- [Configuring a Business Component Field on page 251](#)
- [Configuring a Link on page 261](#)
- [Creating a Business Object on page 266](#)

Configuring a Business Component

This chapter describes how to configure a business component. It includes the following information:

- [Creating a New Business Component on page 245](#)
- [Determining How a Business Component Sorts Records on page 246](#)
- [Determining How a Business Component Sequences Records on page 246](#)
- [Defining Read-Only Behavior for a Business Component on page 248](#)
- [Creating a Recursive Join on a Business Component on page 249](#)
- [Configuring a Business Component to Copy Child Records If the User Copies the Parent Record on page 250](#)
- [Allowing the User to Set a Primary Team Member on page 251](#)

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 [Chapter 10, "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" on page 219](#).
- 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” on page 599](#).

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” on page 74](#).

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” on page 77](#).
- 4 Compile and test your modifications.

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” on page 82](#).

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” on page 23](#).

- 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.
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: <i>name of the numbered business component.name of the sequence value field (Sequence)</i>

- 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” on page 104](#).

- 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 Compile and test your modifications.

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” on page 85](#).

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.

Property	Value
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 Compile and test your modifications.
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” on page 426](#).

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" on page 423](#).
- 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 Compile and test your modifications.

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 on page 252](#)
- [Activating a Multi-Value Field on page 252](#)
- [Validating Data That the User Enters in a Business Component Field on page 253](#)
- [Creating a Business Component Field That Displays More Than One Currency on page 254](#)
- [Configuring Client-Side Import to Update a Business Component Field on page 257](#)
- [Creating a Joined Business Component Field on page 258](#)
- [Creating a Predefault Value for a Joined Business Component Field on page 260](#)

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 on page 217](#)
- [Guidelines for Creating a Business Component on page 76](#)

- 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 Compile and test your modifications.

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” on page 98](#) and [“Creating Multi-Value Groups and Multi-Value Group Applets” on page 423](#).

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 Compile and test your modifications.

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” on page 191](#).

- 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 Compile and test your modifications.

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. Instead, Siebel CRM uses one of the following specialized classes in the Action business component to perform this validation:

- CSSBCActivity for Siebel CRM version 8 of Siebel Business Applications.
- CSSBCFINSActivity Siebel CRM version 8 of Siebel Industry Applications. The parent class is CSSBCActivity.

For more information, see [“Caution About Using Specialized Classes” on page 74](#).

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” on page 74](#).

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
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 Step 4 .
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”](#) on page 256.

- 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 Compile the Oppty and Oppty (SSE) projects.
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” on page 74](#), 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” on page 192](#).
- 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 Compile locked projects, and then test your modifications.

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	<p>Primary key column in the joined table. For example, ROW_ID.</p> <p>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.</p> <p>For a join to a party table, the destination column must reference the PAR_ROW_ID column in the joined table.</p>
Source Field	<p>Foreign key field in the business component. For example, Primary Address Id.</p> <p>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.</p>

- 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	<p>The search specification. For example:</p> <p style="text-align: center;"><code>GetProfileAttr("Primary Address Id")</code></p> <p>For more information, see "Options to Filter Data That Siebel CRM Displays in an Applet" on page 118.</p>

- 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.

Property	Value
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 Compile and test your modifications.

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 Compile all locked projects.
- 5 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.

- 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* '

- 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'

- Set the Link Specification property of the Name and Sales Stage fields in the parent business component to TRUE.
- Compile all locked projects.
- Restart the Siebel client.
- 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 on page 261](#)
- [Configuring a Link That Creates a One-to-Many Relationship on page 263](#)
- [Configuring Two Links That Create a Many-to-Many Relationship on page 263](#)
- [Creating an Association Between One Parent and Multiple Child Records on page 264](#)

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	<p>If the user deletes the parent record, then Siebel CRM deletes all child records.</p> <p>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.</p> <p>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.</p>
Clear	<p>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.</p>
None	<p>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.</p>

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”](#) on page 50.

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” on page 50.](#)

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” on page 265.](#)

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-many link to create a many-to-many 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. [Table 35](#) describes the properties of the ACCNT_NAME column that Siebel CRM uses in the S_ACCNT_POSTN table.

Table 35. Example of Table That Includes Denormalized Columns

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 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

Property	Description
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” on page 109](#).

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](#).

14 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 on page 267](#)
- [Configuring a View on page 270](#)
- [Process of Creating a Screen on page 281](#)
- [Process of Creating a Screen Home Page View on page 285](#)
- [Creating and Deploying an Application on page 297](#)

Process of Creating a View

To create a view, do the following tasks:

- 1 [Creating a View on page 267](#)
- 2 [Editing the Layout of a View on page 268](#)
- 3 [Registering and Associating a View with a Responsibility on page 269](#)

Creating a View

This task is a step in [“Process of Creating a View” on page 267](#).

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.

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” on page 191](#).
- 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" on page 145](#) 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" on page 268](#).

Editing the Layout of a View

This task is a step in ["Process of Creating a View" on page 267](#).

You edit the layout of a view in the Web Layout Editor. This editor allows you to edit the mapping between applets in the view and placeholders in the template.

To edit the layout of a view

- 1 In Siebel Tools, in the Object Explorer, click View.
- 2 Locate the view you must modify in the Views list.
- 3 Right-click the view in the Views list, and then click Edit Web Layout.

If a template is associated with the view, then Siebel Tools displays the Web Layout Editor. The Web Layout Editor displays mapped and unmapped placeholders from the underlying view web template.

If Siebel 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 Web layout, drag an applet from the Applets window and drop it onto an applet placeholder in the template.

The Applets window displays all applets that reference business components in the business object of the view. When you add an applet to a placeholder, Siebel Tools displays the applet in the position that Siebel CRM displays it in the Siebel client.
 - To delete an applet from the layout, click the applet, and then press the DELETE key.
 - To preview the view, right-click the Web Layout Editor, and then click Preview.

In preview, Siebel Tools removes unmapped placeholders and simulates how Siebel CRM displays the view in the Siebel client. This preview is not an exact representation of the final HTML output, but it does provide a close approximation of how Siebel CRM displays the view in the Siebel client.

- To export the preview to an HTML file:
 - Click the File menu, and then click Export.
 - Choose a file name and location in the Save As dialog box.
- To modify the web template, click Change Template that Siebel Tools displays next to the Template text box in the Controls/Columns window.

This configuration might result in an invalid mapping if the corresponding placeholder ID does not exist in the new template. To test for an invalid mapping, right-click, and then click Check Mapping.

- 5 Save your modifications.

Registering and Associating a View with a Responsibility

This task is a step in [“Process of Creating a View” on page 267](#).

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

- 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.

- 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 on page 270](#)
- [Configuring the Thread Bar on page 271](#)
- [Defining the Drilldown Sequence to Configure Search for an Account on page 272](#)
- [Creating an Applet Toggle on page 273](#)
- [Defining Whether a View Is Cachable on page 276](#)
- [Controlling How the User Can Modify View Layouts on page 276](#)
- [Creating a Secure View on page 278](#)
- [Creating a View That Requires an Explicit User Login on page 278](#)
- [Restricting Access to Records in a View on page 279](#)
- [Troubleshooting a View That Siebel CRM Does Not Display in the Siebel Client on page 279](#)

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” on page 267](#).

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.
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.

Property	Description
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 below 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” on page 481](#).

Figure 35 displays the following example of the thread bar:

Opportunity: 2013 4700 Desktops - 2500 units > Contact: Carlson > Activity

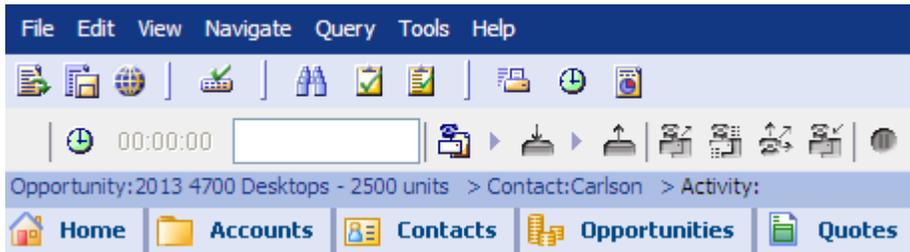


Figure 35. Example of the Thread Bar

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.
Thread Field	The name of the field that Siebel Tools displays to the right of 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 to the left of 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 [Step 2](#) for each view that requires a thread bar.
- 4 Compile and test your modifications.

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 Compile and test your modifications.

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.

Table 36 describes the data requirements for each payment method.

Table 36. Payment Methods and Data Requirements for the Applet Toggle Example

Payment Method	Data Requirement
Cash	No special data requirements exist. The default payment method is Cash.
Credit Card	The following data is required: <ul style="list-style-type: none"> ■ Credit Card Type ■ Credit Card Number ■ Expiration Date
Check	The following data is required: <ul style="list-style-type: none"> ■ 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” on page 141](#).

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” on page 351](#).

- 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 Compile and test your modifications.

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” on page 192](#).

- 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 Compile and test your modifications.

For more information, see *Using Siebel Tools*.

Controlling How the User Can Modify View Layouts

Some views in Siebel CRM, such as the home page view, allow the user to modify the layout of the view. For example, the user can do the following in some views:

- Reorder applets
- Collapse or expand applets
- Show or hide applets

For more information, see *Siebel Fundamentals*.

To control how the user can modify view layouts

- 1 In Siebel Tools, click View in the Object Explorer.
- 2 In the Views list, locate the view you must modify.
- 3 In the Object Explorer, expand the View tree, and then click View Web Template.
- 4 In the View Web Templates list, set the User Layout property to TRUE.
- 5 In the Object Explorer, expand the View Web Template tree, and then click View Web Template Item.

- 6 In the View Web Template Items list, set properties using values from the following table.

Property	Description
Display Size	Determines if Siebel CRM minimizes or maximizes the applet. If you set Display Size to Always Maximized, then the user cannot minimize the applet.
Display Visibility	Determines if Siebel CRM displays or hides the applet. If you set Display Visibility to Always Show, then the user cannot hide the applet.
Move Range	<p>Defines the range where the user can move the applet. For example, on an application home page that contains two columns, applets require a move range of Column1 or Column2. Any applet with a move range of Column1 is movable only in the left-side column. Any applet with a move range of Column2 is movable only in the right-side column.</p> <p>If this property is not defined, then the user cannot move the applet.</p> <p>Siebel CRM fixes the location of an applet in the view. For example, a move range is typically defined for the salutation applet on the home page.</p>

This step defines the default layout of the view.

- 7 In the View Web Template Items list, add the Layout Controls Applet.

- 8 Add the following controls to the appropriate applet in the view:

- ButtonMoveAppletUp
- ButtonMoveAppletDown
- ButtonHideApplet
- ButtonShowApplet
- ButtonMinimizeApplet
- ButtonMaximizeApplet

These view layout controls use invoke methods to manipulate the user preferences of the view layout.

- 9 For each control you added in [Step 8](#), add a corresponding applet web template item, and then map the control to the appropriate placeholder in the web template.

For more information, see [“Properties of the Applet Web Template Item” on page 155](#).

- 10 Compile and test your modifications.

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 Web Server 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

- 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.
- If the Siebel Mobile Web Client is connected to the server database, then it makes the view read only.
- If the Siebel Mobile Web Client is connected to the local database, then it makes the view not read only.

For more information, see Article ID 484433.1 on My Oracle Support.

To restrict access to records in a view

- 1 In the Siebel Mobile Web Client, 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 [Table 37](#).

Table 37. Problems with a View That Siebel CRM Does Not Display in the Siebel Client

Diagnostic Steps or Cause	Solution
The view does not exist in the SRF file. The spelling of the view name does not match the view name in the SRF file.	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 compile the SRF file using the All Projects option.
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.

Table 37. Problems with a View That Siebel CRM Does Not Display in the Siebel Client

Diagnostic Steps or Cause	Solution
<p>The view is not included in one of the responsibilities for the user who is currently logged in.</p>	<p>Use the Administration - User screen, Employees view to identify the responsibilities that Siebel CRM assigns to the user.</p> <p>Use the Administration - Application screen, Responsibilities view to determine to include or not include the view.</p>
<p>The view is hidden using personalization rules.</p>	<p>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” on page 191.</p>
<p>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.</p>	<p>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.</p> <p>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.</p> <p>Make sure the Visibility Applet and Visibility Applet Type properties of the view are set correctly. For more information, see <i>Siebel Security Guide</i>.</p>
<p>The view does not belong to the same business object as the default view for the screen.</p>	<p>Make sure the view references the same business object.</p>
<p>Siebel CRM does not translate the screen menu item or page tab into the appropriate language.</p>	<p>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.</p> <p>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.</p> <p>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.</p> <p>For more information, see Chapter 25, “Localizing Siebel Business Applications.”</p>

Table 37. Problems with a View That Siebel CRM Does Not Display in the Siebel Client

Diagnostic Steps or Cause	Solution
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 <code>DBSRVR\DB_PLATFORM</code> 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. For more information, see Article ID 475818.1 on My Oracle Support. This document was previously published as Siebel Alert 41.

Process of Creating a Screen

To create a screen, perform the following tasks:

- 1 [Creating a Screen on page 281](#)
- 2 [Creating a Page Tab on page 282](#)
- 3 [Creating a Screen Menu Item on page 282](#)
- 4 [Creating a Screen View on page 283](#)
- 5 [Defining the Sequence That Siebel CRM Uses to Display Screen Views on page 283](#)

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" on page 281](#).

You create a new screen in the Screens list in Siebel Tools.

To create a screen

- 1 In the Object Explorer, click Screen.

- 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” on page 281](#).

For more information, see [“Page Tab” on page 27](#) and [“Troubleshooting a View That Siebel CRM Does Not Display in the Siebel Client” on page 279](#).

To create a page tab

- In the Object Explorer, expand the Application tree, and then click Page Tab.
- 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” on page 338 .

Creating a Screen Menu Item

This task is a step in [“Process of Creating a Screen” on page 281](#).

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” on page 27](#).

To create a screen menu item

- 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” on page 191](#).

- 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 Step 2 on page 282 .

Creating a Screen View

This task is a step in [“Process of Creating a Screen” on page 281](#).

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” on page 283](#).

For more information, see [“About Screen Views” on page 133](#).

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” on page 606](#).

Defining the Sequence That Siebel CRM Uses to Display Screen Views

This task is a step in [“Process of Creating a Screen” on page 281](#).

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 below 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 Compile and test your modifications.

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 on page 285](#)
- 2 [Creating Links to Frequently Accessed Data on page 288](#)
- 3 [Determining How Siebel CRM Displays Recent Records on page 289](#)
- 4 [Defining the Business Object for the Screen Home Page View on page 291](#)
- 5 [Creating Simplified Screen Home Page Applets on page 292](#)
- 6 [Creating a Screen Home Page View on page 294](#)
- 7 [Adding the Screen View to the Screen on page 296](#)

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” on page 285](#).

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” on page 69](#).

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: <i>business component name</i> Home Search Virtual 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 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 target business object. For example, Account.

- 3 Create the Home Add Virtual business components:
 - a Repeat [Step a on page 286](#) through [Step c on page 286](#). 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 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 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 <i>field name</i> For example, Mirror Field Account.
Value	Pick, <i>target field, mirror pick Id field</i> 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 f on page 286](#). In the Business Component User Props list, create a new record using values from the following table.

Property	Value
Name	Mirror Add <i>mirror pick Id field name</i> 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” on page 285.

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” on page 192.
- 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: <i>name of the screen home page</i> 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 <i>name of the screen home page</i> 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"](#) on page 285.

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- <i>screen home page business object name</i> 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.

- 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. 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” on page 285](#).

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” on page 105](#).

To define the business object for the screen home page view

- In Siebel Tools, create a new business object using information from the following table.

Property	Value
Name	<i>name of the target business component</i> Home For example, Account Home.
Project	ScreenHomePage

- In the Object Explorer, expand the Business Object tree, and then click Business Object Component.
- 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" on page 285](#).

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" on page 111](#).

To create simplified screen home page applets

- 1 Copy a predefined banner applet.
- 2 Define properties for the new applet you created in [Step 1](#). Use values from the following table.

Property	Value
Name	<i>Name of the target business component Home Screen Homepage Banner</i> 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 above 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 in [Step 3](#). Use values from the following table.

Property	Value
Name	<i>Name of the target business component</i> Home Search Virtual Form Applet For example, Account Home Search Virtual Form Applet.
Business Component	<i>Name of the target business component</i> Home Search Virtual For example, Account Home Search Virtual. You created this business component in “Defining Business Components for the Screen Home Page View” on page 285.

- 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 you created in [Step 5](#). Use values from the following table.

Property	Value
Name	<i>Name of the target business component</i> Home Add Virtual Form Applet For example, Account Home Add Virtual Form Applet.
Business Component	<i>Name of the target business component</i> Home Add Virtual For example, Account Home Add Virtual. You created this business component in “Defining Business Components for the Screen Home Page View” on page 285.

- 7 Do the following for each applet you defined in [Step 3](#) and [Step 5](#):
 - 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 in [Step 3](#) and [Step 5](#):
 - 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 [Step b](#) 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” on page 116](#).

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” on page 155](#).

- 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” on page 138](#).
 - 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” on page 285](#).

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” on page 116](#) and see [“Creating a View” on page 267](#).

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: <ul style="list-style-type: none">■ <i>Name of the target business component</i> Screen Homepage View For example, Account Screen Homepage View.
Title	Use the following format: <ul style="list-style-type: none">■ <i>Name of the target business component</i> Home For example, Account Home.
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” on page 291 .

- 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” on page 292](#).
- 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	Left top
Rapid Add	Left bottom
Homepage Banner	Center To
Public and Private View Link	Center bottom
Screen Home Task Assistant List Applet	Right top
Recent Record	Right bottom

- 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

For more information, see [“Properties of the Applet Web Template Item”](#) on page 155.

Adding the Screen View to the Screen

This task is a step in [“Process of Creating a Screen Home Page View”](#) on page 285.

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
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”](#) on page 283.

- 6 Compile and test your modifications.

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 on page 297](#)
- [Configuring the Sort Order for Siebel CRM on page 298](#)
- [Configuring Keyboard Shortcuts for an Application or Applet on page 300](#)

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” on page 142](#) and [Chapter 10, “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” on page 143](#).

- 4 Add a page container template to the application for your home page.

For more information, see [“About the Container Page” on page 149](#).

- 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” on page 149](#).

- 6 Associate the application with screens.

Configuring the Sort Order for Siebel CRM

To configure the Sort Order dialog box for Siebel CRM, you set the Sort Web Page property of the application object to use the template file for the Sort Order dialog box. You can use the Sort Dialog to display one or more instances of a list of fields to sort and the sort order to use. You can use a control that calls the SortOrder method to call the Sort Dialog. You can use this control only in a base template of a list applets. For information about other sort capabilities, see [“How a Business Component Sorts Records” on page 74](#).

You use the swe:sort-field tag to display the list of sortable fields and the sort order options. This sequence attribute is a required attribute of the swe:sort-field tag that specifies the order that Siebel CRM uses to sort the chosen columns. For information how to configure tags and web templates, see [“About Siebel Web Templates” on page 145](#).

The following code is an example of the format for the swe:sort-field tag:

```
<swe:sort-field sequence="1"/>
```

This tag displays the following HTML select lists:

- A list that includes the fields that the user can sort. Siebel CRM maps these fields to the swe:control tags in the base template for the applet.
- A list that includes Ascending and Descending that determines the sort order.

You can include as many swe:sort-field tags in the sort web page as you require.

To configure the sort order for Siebel CRM

- 1 Open Siebel Tools.
- 2 In the Object Explorer, click Web Page.
- 3 In the Web Pages list, locate the web page that you will use for the sort web page, and then note the web template that this page references.
- 4 Add your custom code to the web template that the sort web page references.
For an example, see [“Code from an Example Sort Web Page” on page 299](#).
- 5 In the Object Explorer, click Application.
- 6 In the Applications list, locate the Siebel application you must modify.
- 7 Set the Sort Web Page property to the web page you located in [Step 3](#).
- 8 To create the link or button that runs the sort, create a web page item that calls the ExecuteSort method. It is not necessary to define the View and Applet parameters for this method because these parameters default to the currently active view and applet.
- 9 Compile and test your modifications.

For more information, see *Using Siebel Tools*.

Code from an Example Sort Web Page

The following code is a fragment from an example sort web page:

```

<swe: form>
<table width=100% bgcolor="#EEEEEE" border=0 cellpadding=0 cellspacing=3>
<tr>
  <td><swe: pageitem id="1" property="DisplayName"/></td>
  <!-- "Sort By" Label -->
</tr>
<tr>
  <td><swe: sort-field sequence="1"/></td>
  <!-- First column to sort on -->
</tr>
<tr>
  <td><swe: pageitem id="2" property="DisplayName"/></td>
  <!-- "Then By" Label -->
</tr>
<tr>
  <td><swe: sort-field sequence="2" /></td>
  <!-- Second column to sort on -->
</tr>
<tr>
  <td><swe: pageitem id="2" property="DisplayName" /> </td>
  <!-- "Then By" Label -->
</tr>
<tr>
  <td><swe: sort-field sequence="3"/></td>
  <!-- Third column to sort on --></tr>
<tr>
  <td><swe: pageitem id="5" property="FormattedHtml" /></td>
  <!-- Execute Sort -->

```

```

</tr>
</table>
</swe: form>

```

Configuring Keyboard Shortcuts for an Application or Applet

This topic describes how to configure keyboard shortcuts. It includes the following information:

- [About Keyboard Shortcuts on page 300](#)
- [Guidelines for Creating Keyboard Shortcuts on page 300](#)
- [Creating a Keyboard Shortcut on page 301](#)
- [Modifying or Hiding the Key Sequence on page 302](#)

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” on page 455](#).

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 accidentally press Ctrl+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 Repository File, 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” on page 192](#).
- 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” on page 455](#).
- 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+0.

Property	Description
Display Name	Enter the display name.
Browser Platform	Choose one of the following values: <ul style="list-style-type: none"> ■ Extended. For extended mode only. ■ Basic. For basic mode only. ■ All. For Extended and Basic modes.

- 8 Compile and test your modifications.

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” on page 300](#).

- 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 Compile and test your modifications.

For more information, see *Using Siebel Tools*.

15 Configuring Applet Layouts

This chapter describes how to use the Applet Layout Editor to configure an applet. It includes the following topics:

- [Process of Using the Applet Layout Editor on page 303](#)
- [Options for Configuring an Applet Layout on page 308](#)
- [Using Grid Layout for an Applet on page 315](#)

Process of Using the Applet Layout Editor

To use the applet layout editor, perform the following tasks:

- 1 [Setting the Language Mode of the Applet Layout Editor on page 303](#)
- 2 [Setting the Configuration Context on page 304](#)
- 3 [Defining the Applet Mode on page 305](#)
- 4 [Adding a Control or List Column to an Applet Layout on page 305](#)
- 5 [Previewing the Applet Layout on page 307](#)
- 6 Optional. [Exporting an Applet Preview to an HTML File on page 308](#)

The *Applet Layout 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” on page 191](#).

Setting the Language Mode of the Applet Layout Editor

This task is a step in [“Process of Using the Applet Layout Editor” on page 303](#).

Siebel Tools displays the current language in the lower right corner 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 SRF file. You can set the language mode. For more information, see [Overview of Localizing a Siebel Application on page 539](#).

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*.

Setting the Configuration Context

This task is a step in ["Process of Using the Applet Layout Editor" on page 303](#).

Capabilities vary between browsers. For example, some browsers support frames and JavaScript. Before you edit the layout of an applet, you set the context so that the Web Layout Editor displays objects that the browser supports.

To set the configuration context

- 1 In Siebel Tools, click the View menu, Toolbars, and then make sure the Configuration Context menu item contains a check mark.

The Configuration Context toolbar includes the Target Browser list. This list allows you to specify the target browser.

- 2 In the Configuration Context toolbar, choose the Target Browser list, and then choose Target Browser Config.
- 3 In the Available browsers window of the Target Browser Configuration dialog box, choose a browser you must map, and then click the right arrow.

You can view capability information about the browser in the Capability Name and Value sections of the dialog box.

- 4 Repeat [Step 3](#), and then click OK.

The target browser determines how Siebel Tools handles conditional template tags in the Web Layout Editor. If you choose more than one browser, then the group of target browsers determines how Siebel Tools handles these tags.

If you do not choose a browser in the Target Browser field, then Siebel Tools displays an error message when you open the Applet Layout Editor.

CAUTION: Do not modify the Configuration Context after you start to modify an applet layout.

- 5 In the Object Explorer, click Applet.
- 6 In the Applets list, locate the applet you must modify.

- 7 Right-click, and then click Edit Web Layout.

The Web Layout Editor displays information differently depending on the target browser you choose.

It might be necessary for you to define different applet layouts to support different browsers.

If the applet you chose does not reference an applet web template, then Siebel CRM displays a dialog box that allows you to open the Applet Wizard so that you can reference a web template.

Defining the Applet Mode

This task is a step in [“Process of Using the Applet Layout Editor” on page 303](#).

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” on page 123](#).

- 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 Layout Editor” on page 303](#).

You can use the Applet Layout 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” on page 113](#).

To add a control or list column to an applet layout

- 1 In Siebel Tools, click Applet in the Object Explorer.
- 2 In the Applets list, locate the applet you must modify, right-click, and then click Edit Web Layout.
- 3 Make sure Siebel Tools displays the Controls/Columns window.

To display the Controls/Columns window, click the View menu, Windows, and then click the Controls Window menu item.

- 4 Drag and drop the control or list column you must add from the Controls/Columns window to the canvas.

Siebel Tools displays the control or list column in the canvas and creates the corresponding web template item. If this is a custom control, then Siebel Tools creates an object definition for the control. If you add a control, then the Applet Layout Editor adds the corresponding child objects to the applet, including the control or list column, and the applet web template item. For more information, see [“Properties of the Applet Web Template Item” on page 155](#).

- 5 Optional. Use the Properties window to define the properties for the control.

If the Properties window is not open, then click the View menu, Windows, and then click the Properties Window menu item.

- 6 To add a control or list column in a layout that does not use a grid, drag and drop the control or list column from the Controls/Columns window to any empty placeholder in the canvas.

The predefined placeholders in the web template determines the locations that are available for a control in an applet that references a web template that does not use a grid. For example, with a list applet. For these applets, you drag and drop a control onto an empty placeholder in the canvas. For more information, see [“About Nongrid Form Applet Templates” on page 157](#).

- 7 Save your modifications.

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” on page 315](#).

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” on page 541](#).

To delete a control or list column

- 1 In Siebel Tools, click Applet in the Object Explorer.
- 2 In the Applets list, locate the applet you must modify, right-click, and then click Edit Web Layout.
- 3 In the canvas, choose the control you must delete.

To choose multiple controls, hold down the Shift key, and then choose the controls you must delete.

- 4 Do one of the following:
 - Click the Edit menu, and then click Cut.
 - Press CTRL+X.
 - Right-click, and then click Delete.

Siebel Tools removes the item from the canvas and deletes the corresponding applet web template object definitions from the Siebel repository.

Previewing the Applet Layout

This task is a step in [“Process of Using the Applet Layout Editor” on page 303](#).

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:

- To view the layout in full view, hide any docked windows, such as the Object Explorer or the Properties window.
- Siebel Tools displays a grid in the preview mode that allows you to estimate the width of the applet layout. The default grid includes cells that measure 100 pixels by 100 pixels. A red bar in the preview mode indicates the right edge of the layout, beyond which the user must scroll horizontally. The bar is two grid cells wide. For the default grid, Siebel CRM displays the bar at 969 pixels, which is optimized for a resolution of 1024 pixels.

You can modify the background grid of the preview to optimize the layout for different monitor settings. For more information, see *Using Siebel Tools*.

- 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 Siebel Tools, click Applet in the Object Explorer.
- 2 In the Applets list, locate the applet you must modify, right-click, and then click Edit Web Layout.
- 3 In the Mode list of the Controls/Columns window, choose the applet mode that you must edit.
- 4 Right-click the canvas, and then click Preview.

Siebel Tools displays an approximation of how Siebel CRM displays the applet in the Siebel client.

Exporting an Applet Preview to an HTML File

This task is a step in [“Process of Using the Applet Layout Editor” on page 303](#).

You can export a preview to an HTML file for later viewing.

To export an applet preview to an HTML file

- 1 While in Preview mode, click the File menu, and then click Export.
- 2 In the Save As dialog box, choose a file name and locate the following directory:

`SIEBEL_TOOLS_ROOT\publ i c`

You must choose this directory so that Siebel Tools correctly displays image files that are referenced in the HTML file, such as buttons.

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 on page 309](#)
- [Displaying a Parent Applet Field in the Title of a Detail Applet on page 309](#)
- [Displaying a Subset of Fields or CRM Records on page 310](#)
- [Displaying a Field Only If the User Chooses Show More on page 311](#)
- [Setting the Tab Order for Fields in an Applet on page 312](#)
- [Setting the Input Method Editor Mode on a Control or List Column on page 312](#)
- [Copying Controls and Labels from an Applet to a Web Template on page 313](#)
- [Verifying the Map Between a Control or List Column and a Placeholder on page 314](#)

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”](#) on page 191.

To configure the display name for a control caption or list column

- 1 In Siebel Tools, click Applet in the Object Explorer.
- 2 In the Applets list, locate the applet you must modify, right-click, and then click Edit Web Layout.
- 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.

Siebel 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 Siebel Tools finds an exact match, then Siebel 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, Siebel Tools updates the Display Name property for the control or list column.
 - If Siebel Tools does not find an exact match, or if the match is not unique to a single symbolic string, then Siebel 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 in the Controls list.
 - Define the Display Name property in the List Columns list.

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 Siebel Tools, click Applet in the Object Explorer.
- 2 In the Applets list, locate the detail applet you must modify, right-click, and then click Edit Web Layout.
- 3 Drag and drop a text control into the placeholder for the title that is positioned on the canvas.
- 4 Give the control a useful name, such as *business component name* Title.

- 5 Modify the HTML Type property of the control to PlainText.
- 6 In the Field property of the control, choose the parent business component field whose value you must display, for example Name.
- 7 Compile and test your modifications.

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” on page 311](#).

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” on page 327](#).

To display a subset of fields or CRM records

- 1 In Siebel Tools, click Applet in the Object Explorer.
- 2 In the Applets list, locate the detail applet you must modify.
- 3 Right-click the applet, and then click Edit Web Layout.
- 4 In the Applet Layout Editor, drag a Link control from the Palette window, and then drop it in the placeholder at the upper right portion of the applet.
- 5 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” on page 348 .
HTML Icon Map	Use one of the following values: <ul style="list-style-type: none">■ For a form applet, use ToggleLayout.■ For a list applet, use ToggleListRowCount.
HTML Type	Link

Property	Value
Method Invoked	Use one of the following values: <ul style="list-style-type: none"> ■ For a form applet, use ToggleLayout. ■ For a list applet, use ToggleListRowCount.
Name	Use one of the following values: <ul style="list-style-type: none"> ■ For a form applet, use ToggleLayout. ■ For a list applet, use ToggleListRowCount.
Read Only	FALSE
Runtime	FALSE
Show Popup	FALSE
Sort	FALSE
Visible	TRUE

- 6 Close the Applet Layout Editor, then step off the applet record to save your modifications.
- 7 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” on page 155](#).
- 8 Compile and test your modifications.
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 Siebel Tools, click Applet in the Object Explorer.
- 2 In the Applets list, locate the detail applet you must modify, right-click, and then click Edit Web Layout.
- 3 In the canvas, right-click the control, and then click More.
Siebel Tools displays the control in the canvas with a down arrow. In the Siebel client, Siebel CRM only displays this control after the user chooses Show More.

For more information, see [“Displaying a Subset of Fields or CRM Records” on page 310](#).

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 In Siebel Tools, click Applet in the Object Explorer.
- 2 In the Applets list, locate the applet you must modify, right-click, and then click Edit Web Layout.
- 3 Click the Format menu, choose Set Tab Order.

Siebel 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.

- 4 To create the tab order, click each control in the same sequence that the user must use to navigate through the controls.

Siebel Tools assigns a sequence number to each control when you click the control.

- 5 After you assign all the controls, click the canvas.

Siebel Tools returns the Applet Layout Editor to normal edit mode. If necessary, you can start at [Step 1](#) to reset the tab order.

- 6 Save your modifications.
- 7 Click the Format menu, and then click Set Tab Order.

Siebel Tools returns the Applet Layout Editor to normal edit mode.

- 8 Repeat steps [Step 1](#) through [Step 7](#) for each applet web template mode.

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” on page 192](#).

- 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 Compile and test your modifications. For more information, see *Using Siebel Tools*.

Copying Controls and Labels from an Applet to a Web Template

You can copy controls and labels from an applet to a web template.

To copy controls and labels from an applet to a web template

- 1 In Siebel Tools, click Applet in the Object Explorer.
- 2 In the Applets list, locate the applet you must modify, right-click, and then click Edit Web Layout.
- 3 In the canvas, hold down the Shift key, and then choose the items you must copy.
- 4 Click the Edit menu, and then click Copy.
- 5 In the Object Explorer, navigate to the web template where you must copy the items.
- 6 Click the Edit menu, and then click Paste.
 - If you paste into the same applet web template that the applet references, or if you paste into a different applet, then Siebel Tools creates new controls and applet web template items.
 - If you paste into the same applet but to another mode, then Siebel Tools creates only the applet web template items.

For more information, see [“Properties of the Applet Web Template Item” on page 155](#).

Undoing Modifications or Deleting an Applet Web Template Item

If you choose Undo from the Edit menu or delete an applet web template item from a layout, then Siebel Tools does not automatically delete controls. If you must delete the controls, then you must delete them manually. For important caution information, see [“Deleting a Control or List Column While in Language Override Mode” on page 541](#).

Verifying the Map Between a Control or List Column and a Placeholder

In some situations, the map between a control or list column and a placeholder in a web template can become invalid. This might occur if Siebel CRM deactivates or deletes the object definition for a control or list column from the Siebel repository but that Siebel CRM still displays on the web template. It might occur if you reference a new web template from an applet, and if the predefined placeholder ID for the control or list column does not exist in the new template.

You cannot assign the same placeholder ID to more than one object.

To verify the mapping between a control or list column and a placeholder

- 1 In Siebel Tools, click Applet in the Object Explorer.
- 2 In the Applets list, locate the applet you must verify, right-click, and then click Edit Web Layout.
- 3 Right-click the canvas of the Applet Layout Editor, and then click Check Mappings.

If a map between a control or list column and a placeholder is invalid, then Siebel Tools prompts you to delete the placeholder from the web template.

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 on page 316](#)
- [Using the Conversion Wizard to Convert a Form Applet to Grid Layout on page 316](#)
- [Modifying the Web Template to Convert a Form Applet to Grid Layout on page 318](#)
- [Identifying an Applet or Applet Web Template That You Cannot Convert to a Grid Layout on page 319](#)
- [Modifying the Background Color of an Applet on page 319](#)
- [Troubleshooting a Grid Layout Conversion Problem on page 320](#)
- [Guidelines for Working with Grid Layout on page 321](#)

Grid layout is a design technology in the Applet Layout 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 Layout Editor. This editor helps you to control the layout of the form applet.

For more information, see [“About Grid Form Applet Templates” on page 156](#).

To use grid layout for an applet

- 1 In Siebel Tools, click the View menu, Toolbars, and then make sure the Format Toolbar menu item contains a check mark.
- 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 click Applet Web Template.
- 5 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” on page 316](#).

- 6 In the Applets list, right-click, and then click Edit Web Layout.
- 7 Add and delete controls, and then arrange controls, as necessary.
- 8 For more information, see [“Guidelines for Arranging Controls in Grid Layout” on page 321](#).

Applet Web Templates That Support Grid Layout

Table 38 describes the applet web templates that support grid layout. For more information, see “About Grid Form Applet Templates” on page 156.

Table 38. Applet Web Templates That Support Grid Layout

Web Template	File Name	Description
Applet Form Grid Layout	CCAppletFormGridLayout.swt	Use with all modes of form applets. This template includes buttons in the applet header.
Applet Popup Form Grid Layout	CCAppletPopupFormGridLayout.swt	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" on page 319](#).
- 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" on page 318](#).
- 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" on page 540](#).
- 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 [Step 7](#).
- 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 Layout Editor" on page 303](#).
- 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 ["Troubleshooting a Grid Layout Conversion Problem" on page 320](#).

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” on page 320](#).

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” on page 319](#).
- 2 In the Object Explorer, click the Applet object type.
- 3 In the Applets list, right-click the applet that you must convert to a grid layout, and then click Edit Web Layout.
- 4 In the Controls/Columns window, click Change Template.
If the Controls/Columns window is not visible, then click the View menu, Windows, and then click Controls Window.
- 5 In the Choose Template dialog box, choose the appropriate template.
For more information, see [“Applet Web Templates That Support Grid Layout” on page 316](#).
- 6 Repeat [Step 5](#) for each applet mode.
After you reference a grid layout template, you can use the Applet Layout Editor to edit the applet. For more information, see [“Process of Using the Applet Layout Editor” on page 303](#) and [“Guidelines for Working with Grid Layout” on page 321](#).

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 top 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 top 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 [Table 39](#).

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.

Table 39. Problems That Occur During Conversion to Grid Layout

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” on page 155.	Use the Applet Layout Editor to map a control to the applet layout. For more information, see “Adding a Control or List Column to an Applet Layout” on page 305.
Cannot convert an applet.	The following items might cause the problem: <ul style="list-style-type: none"> ■ 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” on page 319. 	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” on page 318.	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” on page 316.

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 the right to view the right-most 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 bottom of the applet. The grid layout applet does not compress empty space in the applet.
- In the Applet Layout 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 Layout 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” on page 155](#).
- If you save an applet layout, then the Applet Layout 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.

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 Layout 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 Layout 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 Layout 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.

16 Configuring Applets

This chapter describes how to configure an applet. It includes the following topics:

- [Creating an Applet on page 323](#)
- [Configuring Pop-Up Applets and Windows on page 327](#)
- [Configuring Applet Buttons, Controls, and List Columns on page 333](#)
- [Configuring How Siebel CRM Displays Data in an Applet on page 344](#)
- [Process of Configuring Drilldown from the Calendar Applet on page 352](#)

Creating an Applet

This topic describes how to create an applet. It includes the following information:

- [Creating a List Applet on page 323](#)
- [Creating a Form Applet on page 325](#)

For more information, see the following topics:

- [Chapter 6, “About Applets, Controls and List Columns”](#)
- [Configuring Applet Layouts on page 303](#)
- [Configuring Special-Purpose Applets on page 363](#)
- [Configuring Lists and Pick Applets on page 389](#)
- [Configuring Menus and Toolbars on page 455](#)

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” on page 191.](#)
- 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” on page 126.
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” on page 127.
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” on page 116.](#)

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 [Step 4](#). 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” on page 325.](#)

- 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 Layout Editor and displays the layout of the new list applet ready for you to edit. For more information, see [“Process of Using the Applet Layout Editor” on page 303](#).

- 9 Compile and test your modifications.

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” on page 155](#).

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” on page 191](#).
- 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.
Applet Name	New Account List Applet	Enter a unique name for the applet. For more information, see “Guidelines for Naming an Applet” on page 126 .
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” on page 127 .
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” on page 315 .

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” on page 116](#).

- 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” on page 325](#).

- 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 Layout Editor and displays the layout of the new list applet ready for you to edit.

- 9 Compile and test your modifications.

For more information, see *Using Siebel Tools*.

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 on page 327](#)
- [Creating a Pop-Up Control in an Applet on page 328](#)
- [Creating a Pop-Up Applet That Siebel CRM Opens from an Applet on page 328](#)
- [Creating a Pop-Up Applet That Siebel CRM Opens from a Menu Item on page 330](#)
- [Creating a Pop-Up Wizard on page 331](#)

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 `CSSSWFramePopup` 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” on page 310](#).

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.

Table 40 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 that Table 40 describes, then Siebel CRM pops up a list in the Siebel client instead of a calculator or calendar.

Table 40. How the Data Type of a Field Affects the Pop-Up Control That Siebel CRM Displays

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 Compile and test your modifications.

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” on page 192.](#)
- 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” on page 330.

- 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. 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” on page 116.
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” on page 389](#), and the topics about the radio button and radio button group in *Siebel Business Process Framework: Task UI Guide*.

- 10 Compile and test your modifications.

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 in [Step 1](#). Make sure this code does the following:
 - Runs the specialized behavior. For more information, see [“Caution About Using Specialized Classes” on page 74](#).
 - 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” on page 328](#). 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” on page 192.](#)
- 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	<p>Applet=<i>name of pop-up applet</i>, ShowMode=<i>mode of pop-up applet</i> where:</p> <ul style="list-style-type: none"> ■ <i>name of pop-up applet</i> is the name of the pop-up applet. Required. ■ <i>mode of pop-up applet</i> 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. <p>For example:</p> <p style="padding-left: 40px;">Applet=Product Popup Applet, ShowMode=Edit List</p> <p>For more information, see “Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data” on page 116.</p>
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 Compile and test your modifications.
For more information, see *Using Siebel Tools.*

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” on page 328.](#)

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" on page 332](#).
- 6 On the last template in the sequence, create a control named Finish that closes the applet, and then updates the parent applet.
- 7 Compile and test your modifications.
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 [Step 2 on page 332](#).
- 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 on page 334](#)
- [Calling a Method from a Button in an Applet on page 336](#)
- [Identifying the Controls and List Columns That Siebel CRM Displays in the Siebel Client on page 337](#)
- [Modifying the Text Style of a Control or List Column in an Applet on page 338](#)
- [Displaying Totals for a List Column in an Applet on page 338](#)
- [Defining the Properties of a Control or List Column If HTML Type Is Text on page 341](#)
- [Using a Control to Allow the User to Click a Link to Activate a Record on page 343](#)
- [Displaying the Save Button on page 343](#)

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

To configure a spell check button

1 In Siebel 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” on page 192](#).

2 Create a check spelling button in the applet that contains the field where Siebel CRM must do the spell check:

- a In Siebel 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: <ul style="list-style-type: none"> ■ 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 If Siebel 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, right-click the applet you modified in [Step 2](#), and then click the Edit Web Layout menu item.
 - 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 then drag and drop it onto a placeholder in the Applet Web Template editor.
 - e Right-click in the Applet Web Template editor, and then click the Preview menu item.
Siebel Tools displays an approximation of how it will display the Spell Check button in the Siebel client.
- 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 [Step 2](#) through [Step 4](#) for each additional field.
You must create a button for each field.
- 9 Compile and test your modifications.
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.

- In the Applet User Properties list, add a new record using information from the following table.

Property	Value
Name	CanInvokeMethod: <i>Name of method</i> where: ■ <i>Name of method</i> 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.

- Compile and test your modifications.
 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” on page 155](#).

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, wtmt.*
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 wtmt
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 wtmt.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” on page 191](#).

- 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 Compile and test your modifications.

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 swe:control tag to Total.

For example, use one of the following code:

```
<swe: control id="XXX" property="Total " />
```

or

```
<swe: control id="XXX">
  <swe: this property="Total " />
</swe: control >
```

If the property attribute in the swe:control tag or in the swe:this 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 swe:control tag to a list column. For more information, see [About List Applet Templates on page 159](#).

- 11 Compile and test your modifications.

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” on page 192](#).
- 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 swe:control tag in the template file to Total.
For more information, see [Step 10 on page 339](#).
- 12 Compile and test your modifications.
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 below 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 below 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 Compile and test your modifications.

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” on page 342](#).

- 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” on page 98](#), and [“How the Runtime Property Determines the Icon That Siebel CRM Displays with a Text Box” on page 342](#).

- 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” on page 342](#).

- 9 Compile and test your modifications.

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 to the right of 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 [Step 5 on page 341](#).
 - 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” on page 604](#).
- 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 Compile and test your modifications.

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 on page 344](#)
- [Controlling Query Behavior If the User Presses CTRL+ENTER on page 345](#)
- [Filtering Data That Siebel CRM Displays in an Applet on page 346](#)
- [Displaying HTML Content in an Applet on page 346](#)
- [Displaying a System Field in an Applet on page 351](#)
- [Avoiding Losing Context During a Drilldown on page 352](#)
- [Configuring Quick Fill for a Custom Applet on page 352](#)

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” on page 116](#).

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” on page 345](#).
- 2 Enter text into the Name property to match one of the applet modes described in [Table 18 on page 116](#).
- 3 Set the Type property to one of the applet modes described in [Table 18 on page 116](#).

- 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” on page 154](#).

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” on page 116 .
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” on page 118.](#)

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” on page 348.](#)
- 4 Compile your modifications.
- 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" on page 349](#).
- 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" on page 349](#).
- 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

Table 41 describes the properties of a control that you must set to display HTML content.

Table 41. Properties of a Control That Displays 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	<p>Determines the type of HTML that Siebel CRM displays in the field. You can choose one of the following values:</p> <ul style="list-style-type: none"> ■ 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 “Setting the Field Retrieval Type Property to Service” on page 348. ■ 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 <i>Siebel Portal Framework Guide</i>. ■ 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	<p>Set the HTML Display Mode so that the HTML content displays properly in the browser. You can choose one of the following values:</p> <ul style="list-style-type: none"> ■ 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 <i>Siebel Security Hardening Guide</i>.

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” on page 349](#).

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” on page 192](#).
- 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 Compile and test your modifications.

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” on page 89](#).

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. For more information, see Article ID 1131190.1 on My Oracle Support.

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
NewFromLastTemplate	New From Last Template	SBL_NEW_FROM_LAST_TEMPLATE-1004224721-0CC	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 Compile and test your modifications.

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 on page 353.](#)
- 2 [Defining Fields in the Business Component on page 353.](#)
- 3 [Defining the Applet User Properties on page 356.](#)
- 4 [Creating the Drilldown Objects and Controls on page 358.](#)
- 5 Optional. [Configuring a Different Icon for the Dynamic Drilldown on page 360.](#)
- 6 Optional. [Configuring a Different Destination for the Dynamic Drilldown on page 361.](#)

Preparing Siebel Tools to Configure Drilldowns

This task is a step in [“Process of Configuring Drilldown from the Calendar Applet” on page 352.](#)

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” on page 192.](#)

- 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” on page 352.](#)

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, "", "")

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
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 [Step 6 on page 357](#). 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 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.
- In the Fields list, locate the Primary Contact Icon field.
 - 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.
 - If a predefined join with this definition does not exist, then create a new join using values in [Step b](#) through [Step e](#).
 - 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”](#) on page 352.

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.

- 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.

- 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](#).

- 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](#).

- 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](#).

- 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

- 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”](#) on page 352.

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

- Make sure Activity HI Calendar Applet is still chosen in the Applets list.
- In the Object Explorer, in the Applet tree, click Drilldown Object.
- 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
Source Field	Id
Business Component	Action

- 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

- 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

- Make sure Action - Detail is chosen in the Drilldown Objects list.
- In the Object Explorer, expand the Drilldown Object tree, and then click Dynamic Drilldown Destination.
- 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 [Step 4 on page 354](#) before you can add the dynamic drilldown.

- 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 Compile and test your modifications.

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” on page 352](#).

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

Property	Value
Join	S_CONTACT. As an alternative, you can use the name of the join specification that you use in Step e on page 356 .
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", "", ""))
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” on page 352](#).

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

This chapter describes how to configure special-purpose applets such as tree, attachment, and pop-up applets. It includes the following topics:

- [Configuring a Tree Applet on page 363](#)
- [Configuring a Hierarchical List Applet on page 375](#)
- [Configuring a File Attachment Applet on page 382](#)
- [Configuring an Organization Analysis Applet on page 387](#)

Configuring a Tree Applet

This topic describes how to configure a tree applet. It includes the following information:

- [Overview of Configuring a Tree Applet on page 363](#)
- [Using the Tree Applet Wizard to Create a Tree Applet on page 367](#)
- [Configuring a Tree Node on page 368](#)
- [Using the Applet Layout Editor to Add a Tree Control on page 370](#)
- [Configuring a Recursive Tree Applet on page 371](#)
- [Configuring the Graphic Elements of a Tree Applet on page 372](#)

For more information, see [“About Tree Applet Templates” on page 162](#), and [“Configuring Icons in a Tree Applet” on page 467](#).

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 on the left side of the applet. It displays detailed information for a chosen tree node in the details applet in a frame to the right. 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 `swe:node` tag specifies the placeholder for a tree item. For more information, see [About Siebel Tags on page 168](#).

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 on the left side of the interface and the Service Request List Applet in a frame on the right side of 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 ["About Business Objects" on page 105](#).

Figure 36 describes the relationships and objects that the Service Request business object contains.

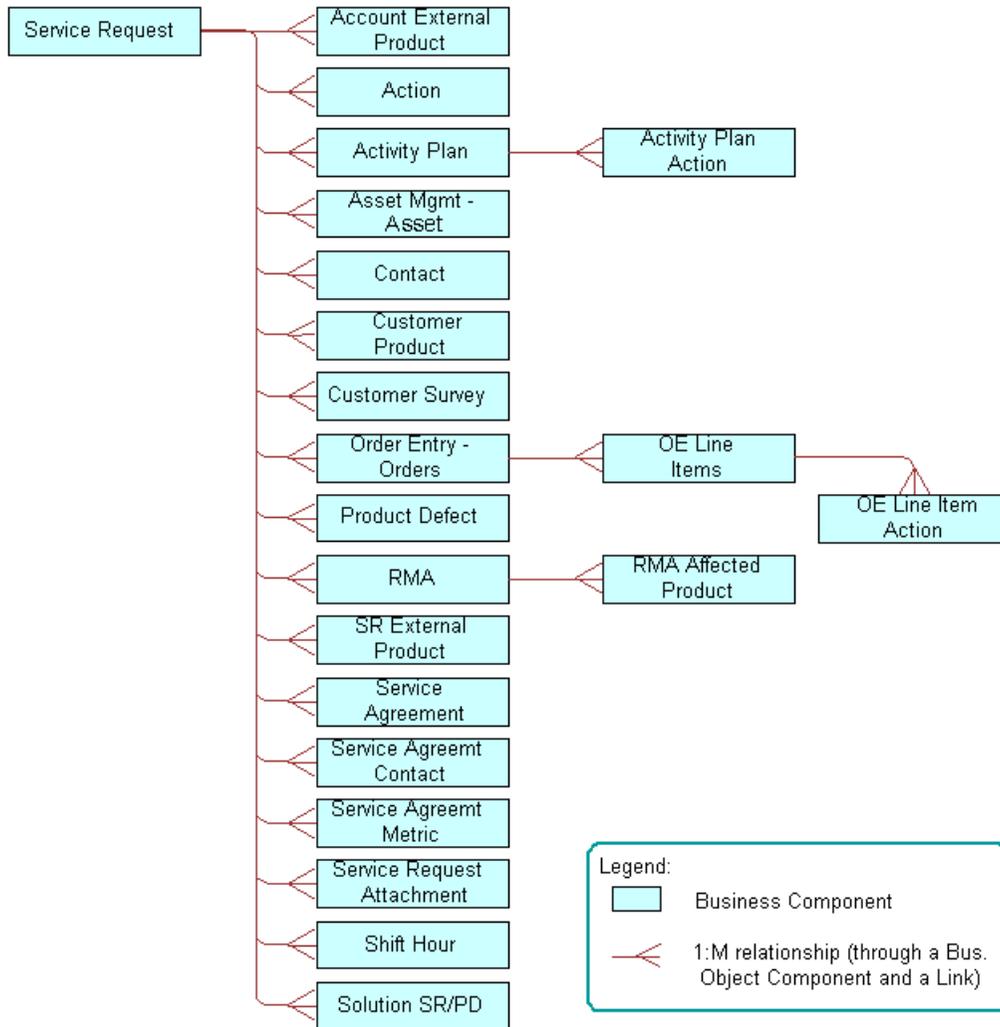


Figure 36. Relationships and Objects in the Service Request Business Object

Figure 37 describes relationships and objects in the Service Request business object that Siebel CRM uses in the Service Request Explorer View.

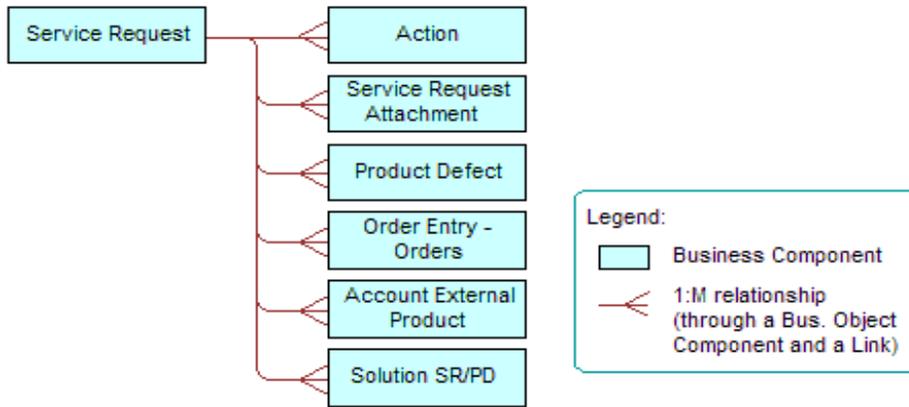


Figure 37. Relationships and Objects in the Service Request Business Object That Siebel CRM Uses in the Service Request Explorer View

Table 42 describes the relationship between business components in the Service Request business object and folder names in the tree applet.

Table 42. Relationship Between Business Component and Folder Name

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.

Table 43 describes properties of an applet that implement a tree applet.

Table 43. 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 top 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” on page 118](#).

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 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” on page 368](#).

Configuring a Tree Node

One tree node defines one folder icon. This includes the top 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 to the right of the folder icon.
Applet	Specify the applet that Siebel CRM displays in the right 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.

Property	Description
Position	<p>Do the following:</p> <ul style="list-style-type: none"> ■ 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. <p>For more information, see “Defining the Position Property of a Tree Node” on page 369.</p>
Business Component	<p>Specify the same business component that is defined for the applet that Siebel CRM displays in the right portion of the view.</p> <p>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.</p>
Label Field	<p>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:</p> <ul style="list-style-type: none"> ■ 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	<p>Specify the number 5. This number identifies the folder symbol.</p>

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 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 rightmost digit in a position typically specifies the order relative to other nodes that exist on the same level.

Using the Applet Layout Editor to Add a Tree Control

You can use the Web Layout Editor to add a tree control to a tree applet.

To use the Applet Layout 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 Drag a TreeControl control from the palette and drop it on 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 table 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 top 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: <ul style="list-style-type: none"> ■ Use the up and down arrow keys to move the tree node up or down a level. ■ Use the left and right 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 Web Layout Editor.

If you press the DELETE key when Siebel Tools displays the tree in the Applet Web Template Layout 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.
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	<p>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:</p> <p style="text-align: center;">[Parent xxx Id] is NULL</p> <p>where:</p> <ul style="list-style-type: none"> ■ xxx completes the name of the field. <p>For example:</p> <p style="text-align: center;">[Parent Account Id] is NULL</p> <p>For more information, see “Options to Filter Data That Siebel CRM Displays in an Applet” on page 118.</p>

Configuring the Graphic Elements of a Tree Applet

A tree control includes reusable graphic elements and text that Siebel CRM gets from a business component record, as defined in the tree and tree node. Siebel CRM defines the graphic elements in a tree applet, such as elbows, folder symbols, and so on, as parameters of the Application Object Manager. You can use these parameters to configure how Siebel CRM displays the folder and document symbols, expand icons, collapse icons, elbows, spacers, and so on. For more information about Application Object Manager, see *Siebel System Administration Guide*.

You can use an HTML hierarchy bitmap to configure some graphic elements. For more information, see [“Configuring Icons in a Tree Applet” on page 467](#).

To configure the graphic elements of a tree applet

- 1 In the Siebel client, navigate to the Administration - Server Configuration screen, and then the Servers view.
- 2 In the Siebel Servers list, locate the Siebel Server of interest.
- 3 Click the Components tab.
- 4 In the Components list, locate the Application Object Manager of interest.
For example, Call Center Object Manager (ENU).
- 5 Click the Parameters subview tab, and then click Hidden.
- 6 In the Component Parameters list, query the Parameter field for the parameter you must modify, and then set the values.

For more information, see [“Parameters You Can Modify That Determine How Siebel CRM Displays Graphic Elements in a Tree Applet” on page 373](#).

Parameters You Can Modify That Determine How Siebel CRM Displays Graphic Elements in a Tree Applet

Table 44 lists parameters you can modify that determine how Siebel CRM displays graphic elements in a tree applet.

Table 44. Parameters You Can Modify That Determine How Siebel CRM Displays Graphic Elements in a Tree Applet

Type of Tree Element	Application Object Manager Parameters
Elbows and Trees	<p>Siebel CRM supports the following parameters:</p> <ul style="list-style-type: none"> ■ <code>TreeNodeCollapseElbowCaption</code> ■ <code>TreeNodeCollapseTeeCaption</code> ■ <code>TreeNodeElbowCaption</code> ■ <code>TreeNodeExpandElbowCaption</code> ■ <code>TreeNodeExpandTeeCaption</code> ■ <code>TreeNodeTeeCaption</code>
Root, Leaf, Open Folder, and Closed Folder Icons	<p>Siebel CRM supports the following parameters:</p> <ul style="list-style-type: none"> ■ <code>TreeNodeCloseFolderCaption</code>. ■ <code>TreeNodeLeafCaption</code>. ■ <code>TreeNodeOpenFolderCaption</code>. Open folder with a dangling line. ■ <code>TreeNodeOpenFolder2Caption</code>. Open folder without a dangling line. ■ <code>TreeNodeRootCaption</code>. ■ <code>TreeNodeArrowDownCaption</code>. This icon indicates that more records exist that are not described below the caption. If the user clicks this icon, then Siebel CRM displays the next group. ■ <code>TreeNodeArrowUpCaption</code>. This icon indicates that more records exist that are not described above.
Indentation Graphics	<p>Siebel CRM supports the following parameters:</p> <ul style="list-style-type: none"> ■ <code>TreeNodeBarCaption</code> ■ <code>TreeNodeSpaceCaption</code>
Text Style Parameters	<p>Siebel CRM supports the following parameters:</p> <ul style="list-style-type: none"> ■ <code>TreeNodeFontStyle</code>. The default value is MS Sans Serif, Arial, and Helvetica. ■ <code>TreeNodeFontSize</code>. The default value is 1. ■ <code>TreeNodeSelectBgColor</code>. The default value is #000080. ■ <code>TreeNodeSelectFgColor</code>. The default value is #ffffff.m.

Using the Configuration File to Specify Parameters

You can use the configuration file to specify parameters that determine how Siebel CRM displays graphic elements in a tree applet.

To use the configuration file to specify parameters

- 1 Open the relevant configuration file in a text editor.
- 2 Add a separate line for each parameter you must specify.

Use the following format:

```
parameter_name = "<param1 param2>"
```

where:

- *param1* and *param2* are the names of the parameters.

For example:

```
TreeNodeCollapseCaption = "<img src='images/tree_collapse.gif' alt='-' border=0  
align=left vspace=0 hspace=0>"
```

You can use the alt parameter in the img tag to replace an image with text. This configuration is useful to support a browser that only displays text.

Configuring How Siebel CRM Displays Text From Field Values

You can configure how Siebel CRM displays text that it gets from field values.

To configure how Siebel CRM displays text from field values

- 1 Open the relevant configuration file in a text editor.
- 2 Add a separate line for each parameter you must specify, using the following format:

```
parameter_name = value
```

where:

- *parameter_name* is one of the following parameters:
 - `TreeNodeFontStyle`
 - `TreeNodeFontSize`
 - `TreeNodeSelectBgColor`
 - `TreeNodeSelectFgColor`

The term caption that Siebel CRM displays in the parameter refers to an icon or graphic. It displays the caption as an image and positions it in one of the following ways:

- Precedes the text that Siebel CRM creates from a field value
- Precedes another caption

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 on page 375](#)
- [Configuring Indentation and Order of a Hierarchical List Applet on page 376](#)
- [Limiting the Number of Records That Siebel CRM Returns in a Hierarchical List Applet on page 377](#)
- [Configuring a Hierarchical List Applet to Use External Data on page 377](#)

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” on page 363](#).

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 [Table 45 on page 376](#).

[Table 45](#) 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.

Table 45. Applet Method Menu Items That Control the Hierarchy of a Hierarchical List Applet

Item	Description
Indent	Moves the current record to a position that is indented from the peer record.
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 above the position of the peer record.
MoveDown	Moves the current record to a position that is under 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” on page 118](#).
 - Use a predefined query for the view. For more information, see [“Guidelines for Modifying a Predefined Query” on page 180](#).

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 on page 377](#)
- 2 [Creating the Business Service for the Hierarchical List Applet on page 378](#)
- 3 [Implementing the Customization on page 381](#)

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.

- In the Business Components list, create a new business component using values from the following table.

Property	Value
Hierarchy Parent Field	Id

- In Business Components list, right-click the record you created in [Step 2](#), and then click Edit Server Scripts.
- In the Scripting Language dialog box, choose eScript, and then click OK.
- In the BusComp Script window, expand the BusComp tree, and then click BusComp_PreInvokeMethod.
- 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);
}
```
- 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 + "()");
}
```
- In the Object Explorer, expand the Business Components tree, and then click Field.
- 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” on page 192.](#)

- 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: <pre> 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(Cancel Operation); } </pre>

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 “Script for the Query Method for Configuring the Hierarchical List Applet” on page 626.

- 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: <pre> 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); } </pre>

- 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 [“About Last Child Info” on page 381](#).

The code can be similar to the code that you added in [Step 7](#).

- 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 SRF file.
- 2 If you defined a new screen, then add the screen to the application screen object.
- 3 Compile your modifications.
- 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 on page 384](#)
- [Configuring an Attachment Table on page 386](#)

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” on page 74](#).

To configure a file attachment applet

- 1 If necessary, configure an attachment business component.
For more information, see [“Configuring an Attachment Business Component” on page 384](#).
- 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” on page 384 .
Class	Set to one of the following values: <ul style="list-style-type: none"> ■ CSSFrameListFile for an attachment list applet ■ 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	<i>prefix</i> FileName	TextBox
Local	Dock Status	CheckBox
Request	<i>prefix</i> FileDockReqFlg	CheckBox
Size	<i>prefix</i> FileSize	TextBox
Type	<i>prefix</i> FileExt	TextBox
Modified	<i>prefix</i> FileDate	TextBox
Auto Update	<i>prefix</i> FileAutoUpdFlg	CheckBox

For the prefix, enter the required prefix for the business component. For more information, see [“Prefix for the Field Name” on page 384](#).

These list columns or controls reference fields in the attachment business component. For more information, see [“Configuring an Attachment Business Component” on page 384](#).

- 4 Make sure the value in the Detail Applet property of each list column or text box control you added in [Step 3](#) 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” on page 386](#).
- 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” on page 384 .

- 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: <ul style="list-style-type: none"> ■ TRUE. If the file does not exist, then the user cannot enter the name of the file. TRUE is the typical value. ■ 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” on page 385](#).

Fields in an Attachment Business Component

Table 46 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.

Table 46. Fields in an Attachment Business Component

Name	Column	Type	Text Length
<i>prefix</i> FileAutoUpdFlg	FILE_AUTO_UPD_FLG	DTYPE_BOOL	1
<i>prefix</i> FileDate	FILE_DATE	DTYPE_DATETIME	7
<i>prefix</i> FileDeferFlg	FILE_DEFER_FLG	DTYPE_TEXT	1
<i>prefix</i> FileDockReqFlg	FILE_DOCK_REQ_FLG	DTYPE_TEXT	1
<i>prefix</i> FileDockStatFlg	FILE_DOCK_STAT_FLG	DTYPE_TEXT	1
<i>prefix</i> FileExt	FILE_EXT	DTYPE_TEXT	10
<i>prefix</i> FileName	FILE_NAME	DTYPE_TEXT	200
<i>prefix</i> FileRev	FILE_REV_NUM	DTYPE_ID	15
<i>prefix</i> FileSize	FILE_SIZE	DTYPE_NUMBER	22
<i>prefix</i> FileSrcPath	FILE_SRC_PATH	DTYPE_TEXT	220
<i>prefix</i> FileSrcType	FILE_SRC_TYPE	DTYPE_TEXT	30

Table 47 describes a field that the file engine does not supply. This field is typically present but is not required.

Table 47. Field in an Attachment Business Component That the File Engine Does Not Supply

Name	Column	Type	Calculation
Dock Status	(calculated)	DTYPE_BOOL	IIf ([AcctFileDockStatFlg] = "N" OR [AcctFileDockStatFlg] 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 drag and drop functionality or the browser dialog box in the corresponding file attachment applet to bring data into the empty table one file at a time.

Table 48 describes required file columns in an attachment table.

Table 48. 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 Table 48 for the Name property. You are not required to use the value described in Table 48 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*.

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.

Property	Description
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” on page 191 .

- 5 Repeat [Step 4](#) for each additional field that Siebel CRM must display in the applet.
- 6 Compile and test your modifications.

For more information, see *Using Siebel Tools*.

18 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 on page 389](#)
- [Configuring Lists and Pick Applets on page 407](#)
- [Creating a List of Values on page 416](#)
- [Associating an Organization with a List of Values on page 418](#)

About Lists and Pick Applets

This topic describes lists and pick applets. It includes the following information:

- [About Static Lists on page 389](#)
- [About Pick Applets on page 393](#)
- [About Dynamic Lists on page 395](#)
- [About Hierarchical Lists on page 405](#)

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” on page 416](#).
- **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 to the right of 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”](#) on page 407 and [“Creating a List of Values”](#) on page 416.

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

Figure 38 describes the relationships and objects that Siebel CRM uses to create a static list.

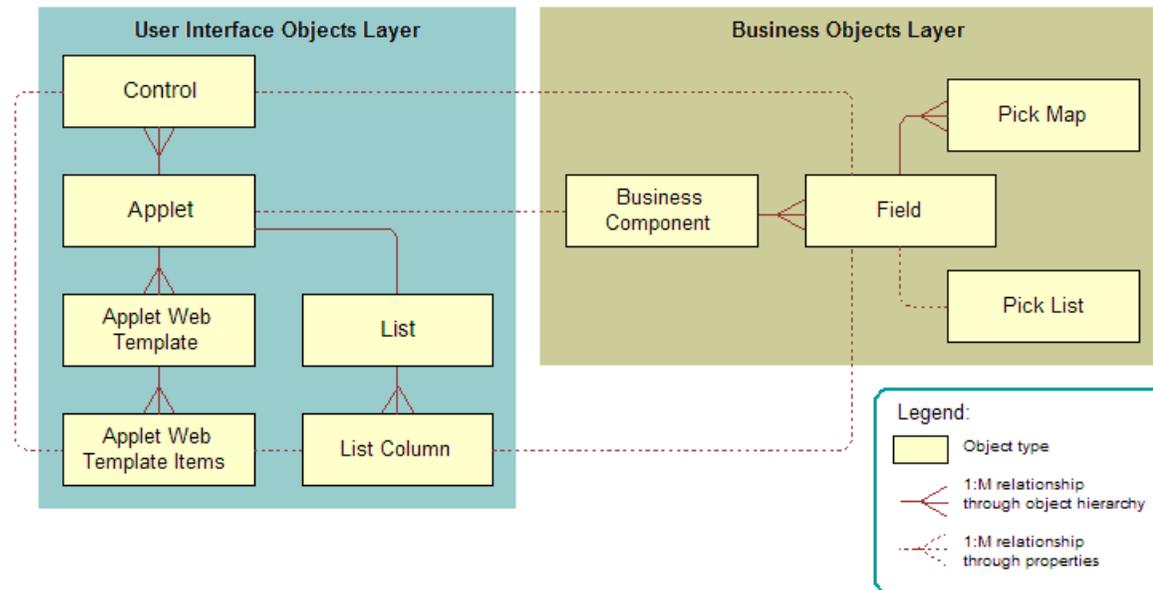


Figure 38. Relationships and Objects That Siebel CRM Uses to Create a Static List

Figure 39 includes an example of how Siebel CRM creates a static list. This example implements the Quality list that is described in Figure 38 on page 390.

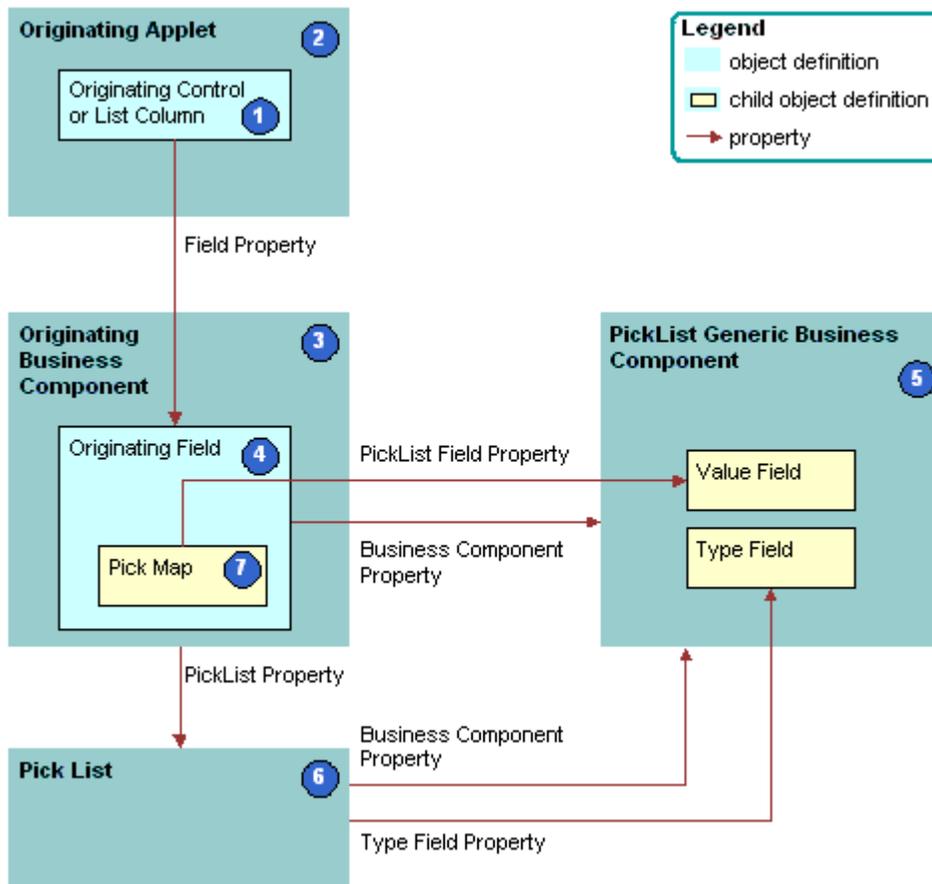


Figure 39. How Siebel CRM Creates a Static List

Explanation of Callouts

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” on page 392.](#)
- 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” on page 416.](#)
- **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” on page 74.](#)

[Table 49](#) lists example data in the Picklist Generic business component.

Table 49. Example of Data in the Picklist Generic Business Component

Type Field	Value Field
LEAD_QUALITY	Excellent
LEAD_QUALITY	Very Good
LEAD_QUALITY	High
LEAD_QUALITY	Fair

Table 49. Example of Data in the Picklist Generic Business Component

Type Field	Value Field
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” on page 395](#), and [“Creating a List of Values” on page 416](#).

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” on page 411](#).

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. 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

Figure 40 describes how Siebel CRM creates a pick applet.

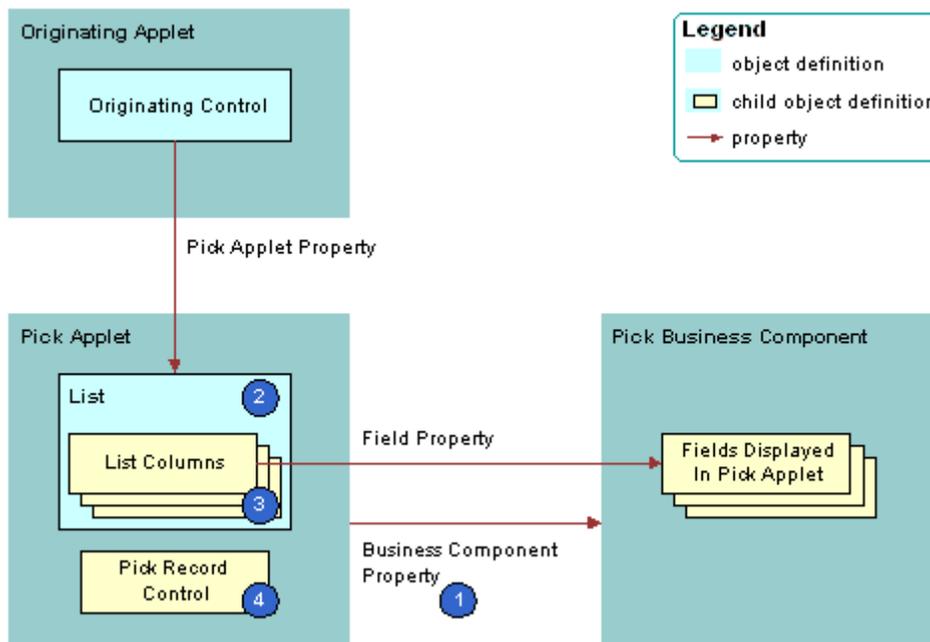


Figure 40. How Siebel CRM Creates a Pick Applet

Explanation of Callouts

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.

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” on page 116](#) and [“Creating a Hierarchical List” on page 414](#).

About Dynamic Lists

This topic describes the dynamic list. It includes the following information:

- [Example of How Data Flows in a Pick Applet on page 396](#)
- [How Siebel CRM Creates a Dynamic List on page 397](#)
- [Originating Applet of a Dynamic List on page 400](#)
- [Originating Business Component of a Dynamic List on page 401](#)
- [How Siebel CRM Constrains a Dynamic List on page 403](#)

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”](#) on page 412.

Example of How Data Flows in a Pick Applet

Figure 41 describes how data in the pick applet typically originates from a different business component than the business component that supplies data to the originating applet.

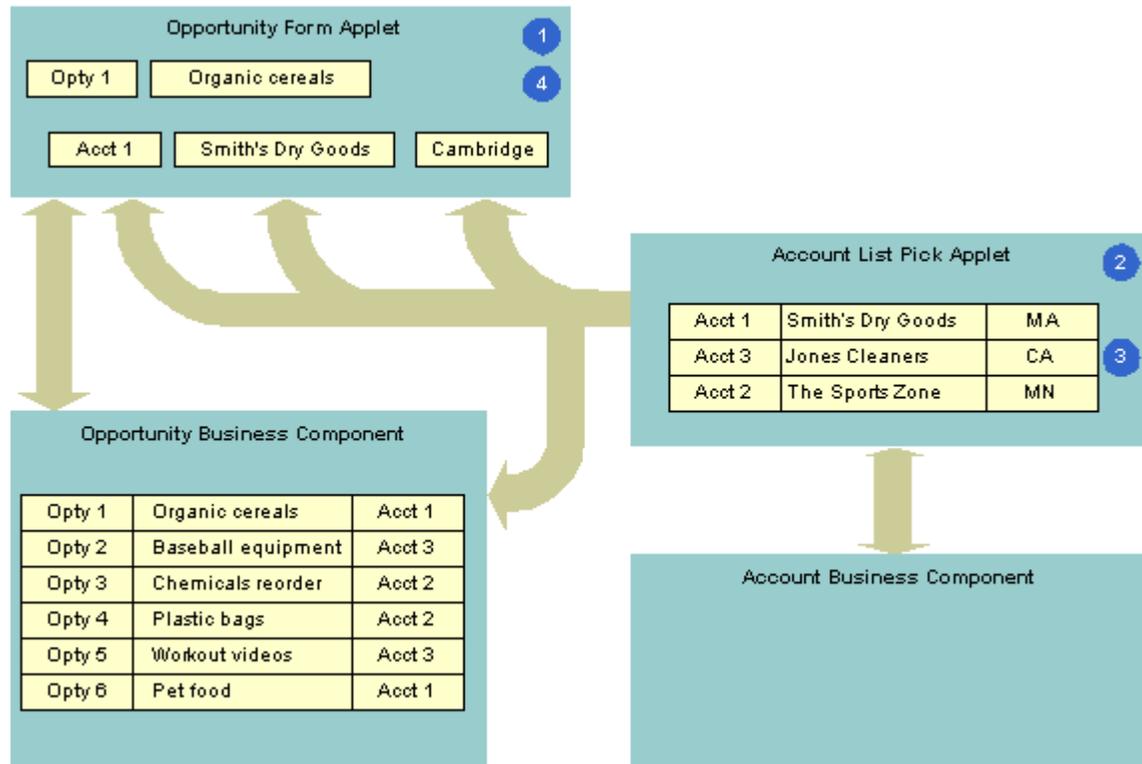


Figure 41. Example of How Data Flows in a Pick Applet

Explanation of Callouts

The following steps occur in this example:

- 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

Figure 42 describes the relationships between object types that Siebel CRM uses to create a dynamic list.

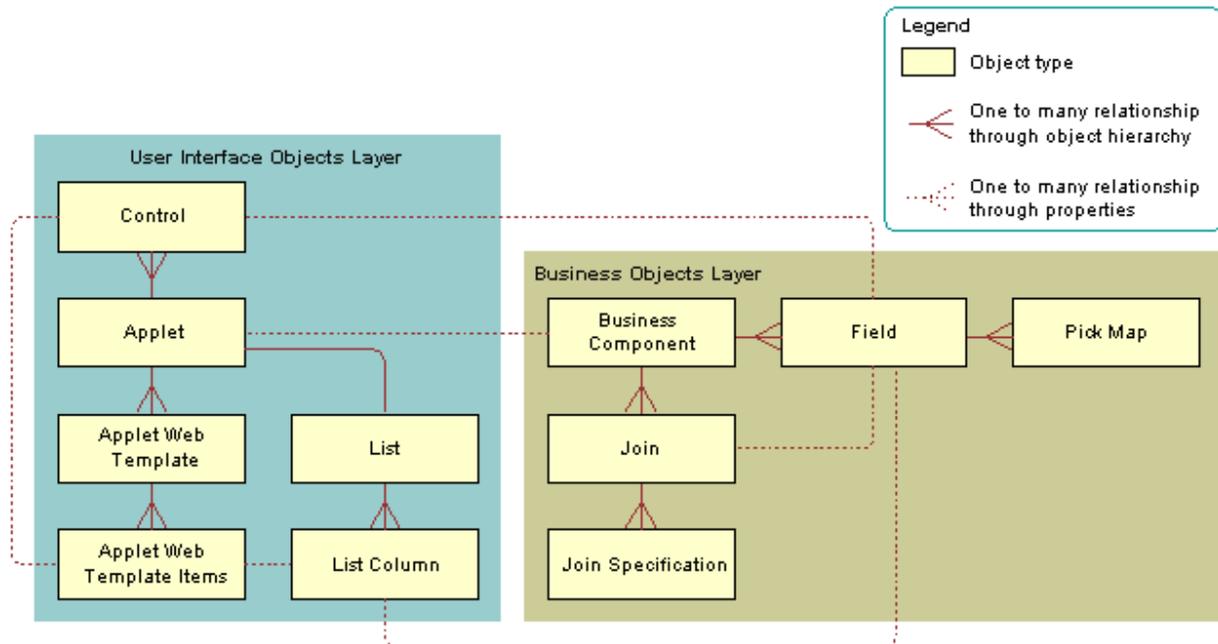


Figure 42. Relationships Between Object Types That Siebel CRM Uses to Create a Dynamic List

Figure 43 describes the objects that Siebel CRM uses to create a dynamic list.

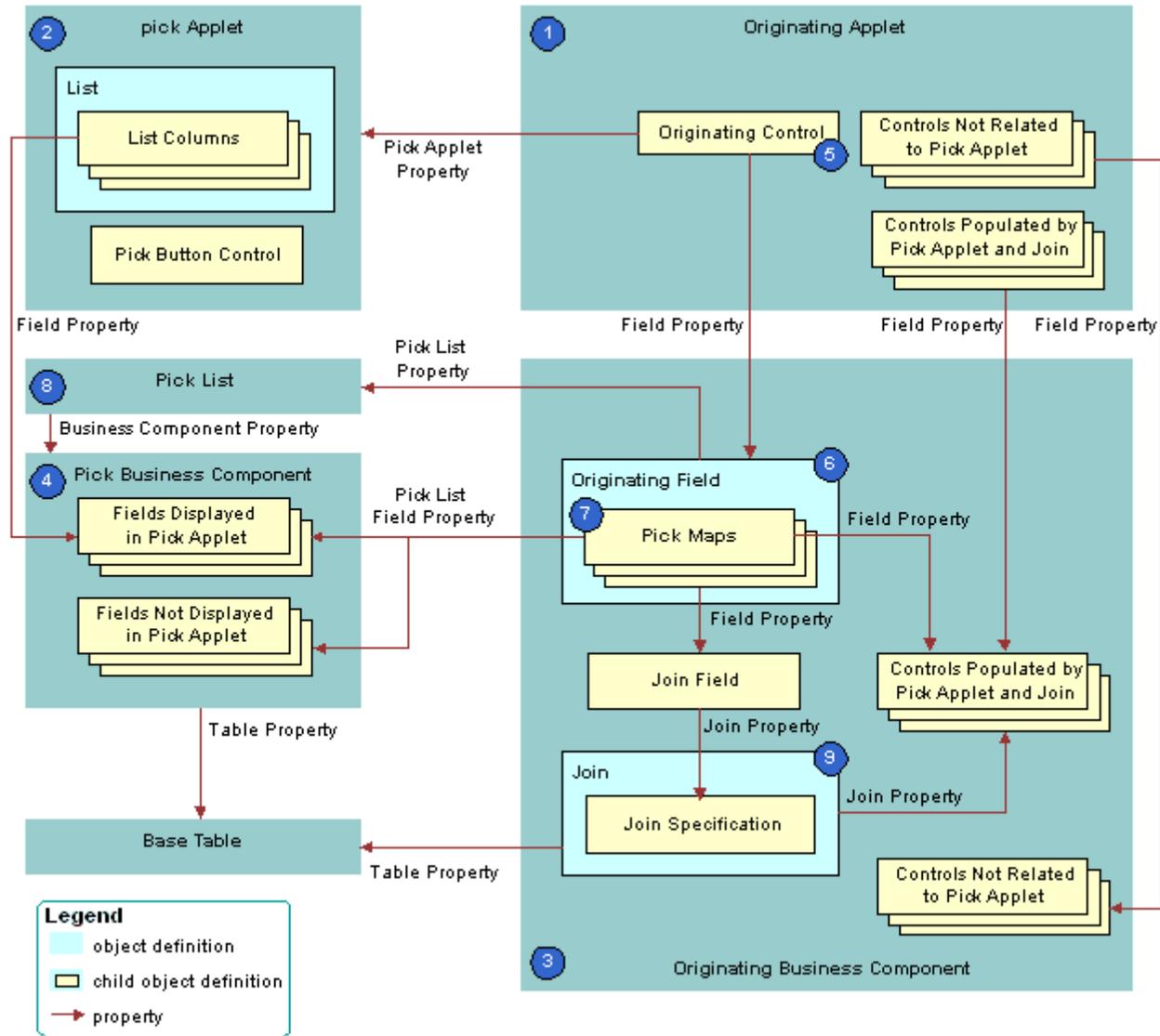


Figure 43. Objects That Siebel CRM Uses to Create a Dynamic List

Explanation of Callouts

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” on page 400](#).

- 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.

Figure 44 describes the details of the originating applet that Figure 42 on page 397 includes.

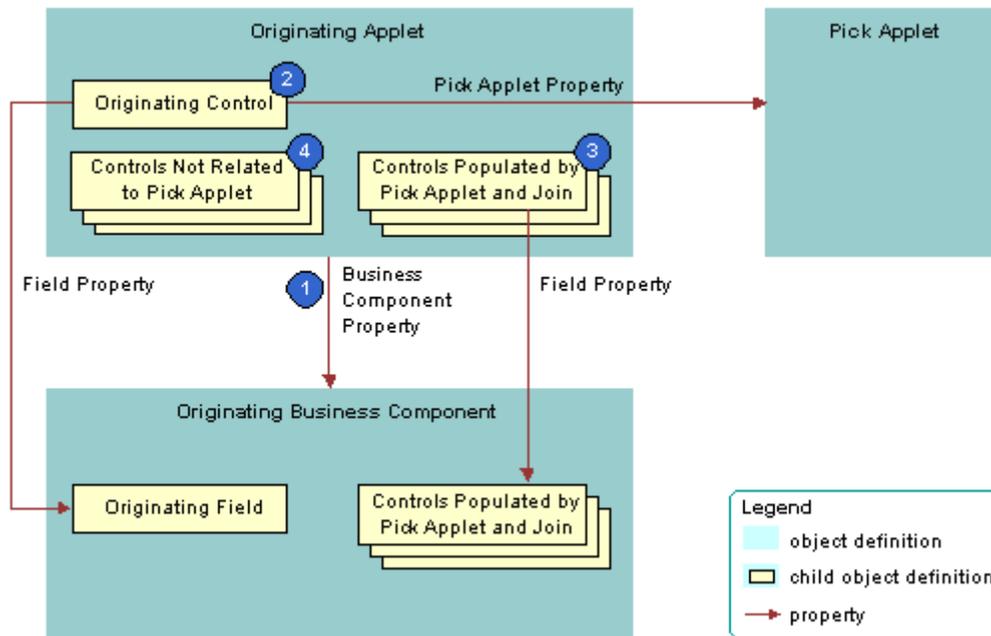


Figure 44. Originating Applet of a Dynamic List

Explanation of Callouts

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” on page 401](#).
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.

Figure 45 describes how Siebel CRM defines the originating business component of a dynamic list. This list is described in “Originating Applet of a Dynamic List” on page 400.

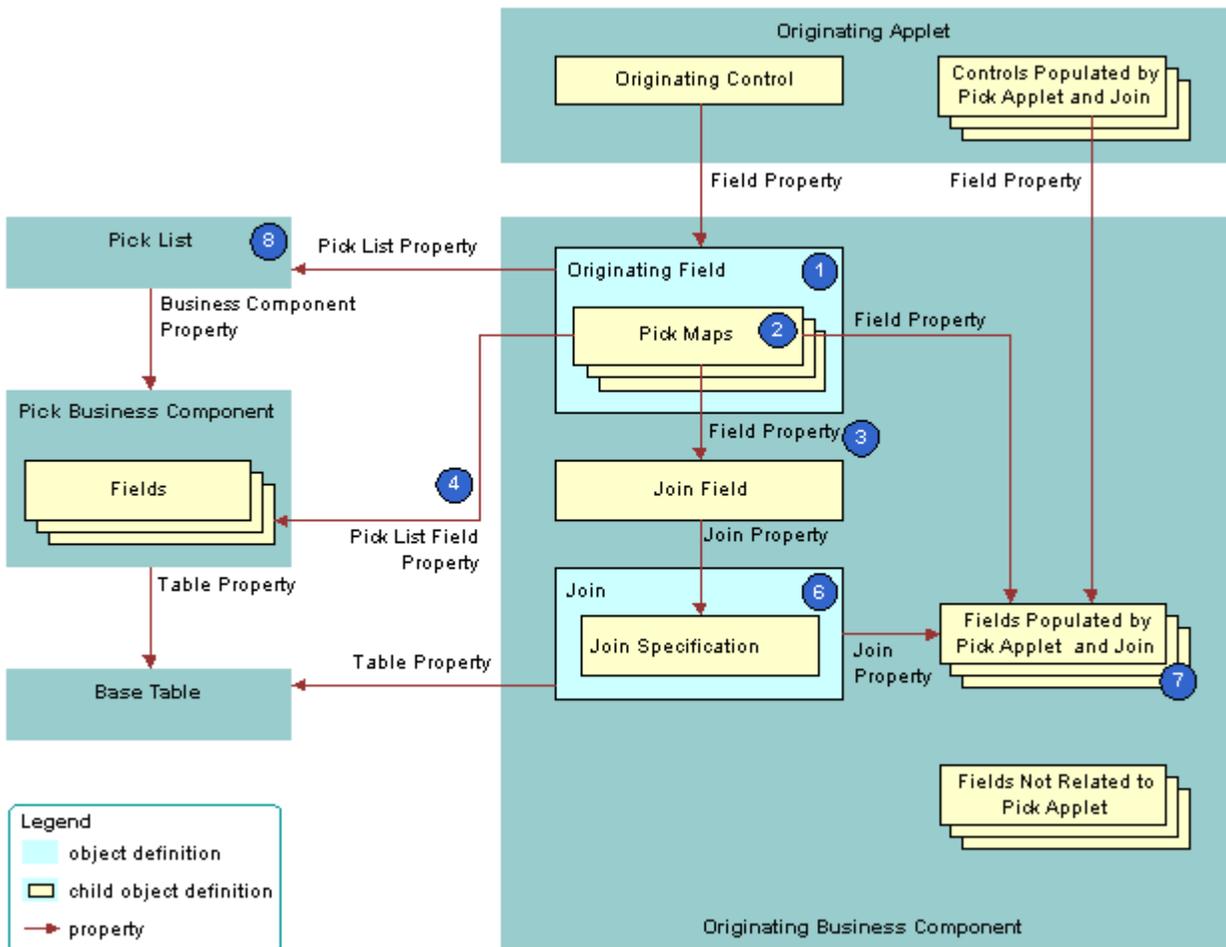


Figure 45. Originating Business Component of a Dynamic List

Explanation of Callouts

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” on page 98](#).

- 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” on page 426](#).

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” on page 414](#).

Figure 46 describes how Siebel CRM constrains a dynamic list. It includes details of the pick business component that Figure 42 on page 397 describes.

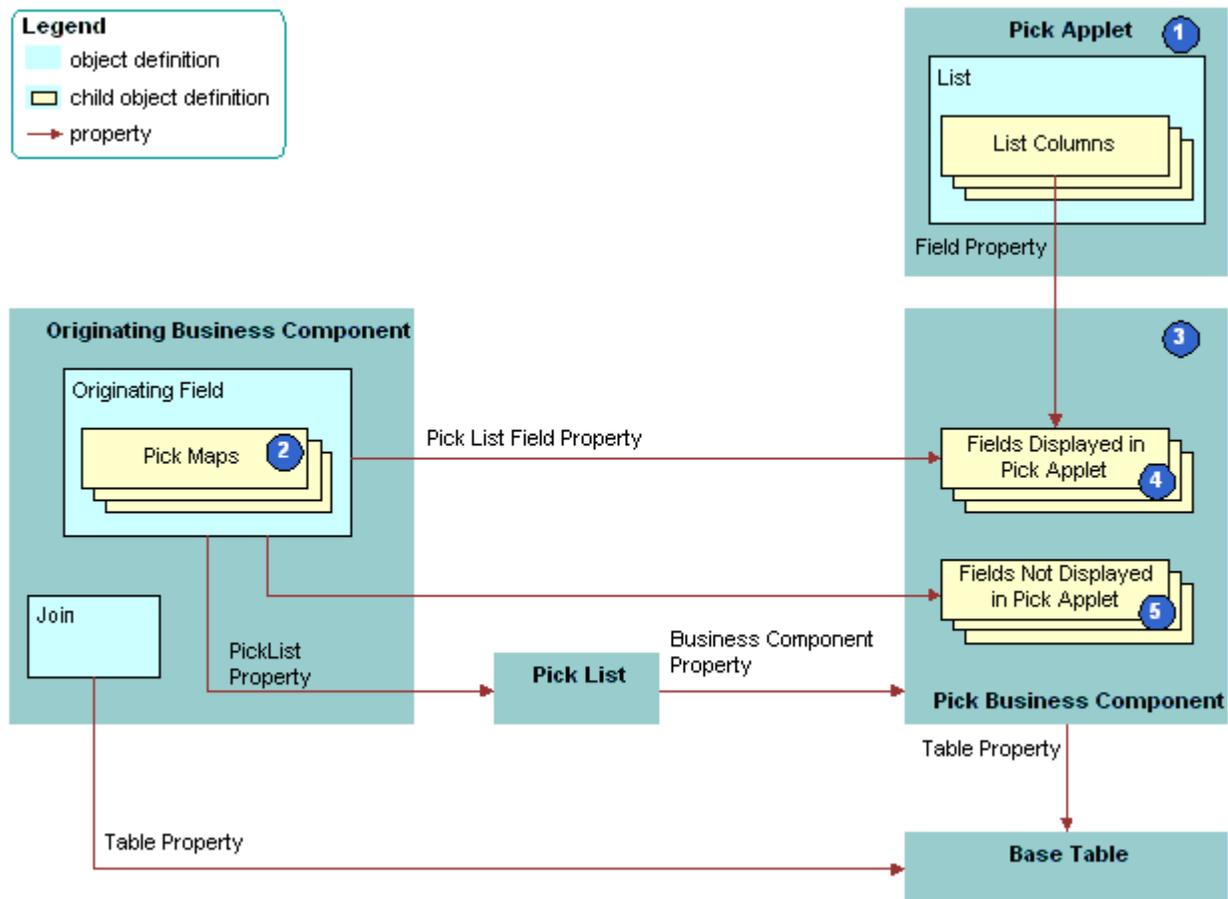


Figure 46. How Siebel CRM Constrains a Dynamic List

Explanation of Callouts

Siebel CRM uses the following objects to constrain a dynamic list:

- 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.
- 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” on page 401](#).

- **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 [“Constraint Pick Map Acts as a Predefault Value” on page 405](#).

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” on page 414](#).

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” on page 416](#).

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 [Table 50](#).

Table 50. Example List of Values for a Hierarchical List

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 [Table 50](#) 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” on page 416](#).

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.

Configuring Lists and Pick Applets

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 on page 407](#)
- [Creating a Static List Manually on page 408](#)
- [Using the Pick Applet Wizard to Create a Pick Applet on page 411](#)
- [Using the Pick List Wizard to Create a Dynamic List on page 412](#)
- [Constraining a Dynamic List on page 414](#)
- [Creating a Hierarchical List on page 414](#)

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” on page 408](#).
 - 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” on page 118](#).
 - 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” on page 416](#) 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” on page 407](#) and [“How Siebel CRM Creates a Static List” on page 390](#).

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.

Property	Description
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" on page 392 .
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.

Property	Description
Type Value	<p>Enter the relevant type for the list of values.</p> <p>For example, the values that display in the Lead Quality list in Table 49 on page 392 include a Type Value property whose value is LEAD_QUALITY.</p>
Search Specification	<p>In most situations, you can leave the Search Specification property empty. For more information, see “How Siebel CRM Handles a Hierarchy of Search Specifications” on page 120.</p>
Sort Specification	<p>Do one of the following:</p> <ul style="list-style-type: none"> ■ To use the sort specification that is defined for the business component, leave the Sort Specification property empty. ■ To override the sort specification that is defined for the business component, define a value in the Sort Specification property. <p>For more information, see “Creating a Sort Specification for a Static List” on page 410.</p>
No Insert	<p>Make sure the No Insert property contains a check mark.</p> <p>If the No Insert property does not contain a check mark, then Siebel CRM creates an error that is similar to the following:</p> <p style="text-align: center;">Unable to create list popup applet</p>

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” on page 74](#).

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 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” on page 116.](#)
- 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 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” on page 303](#).

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” on page 397](#).
- **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” on page 407](#).

- 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	<p>Enter a name for the new PickList object. Use the following format:</p> <p style="text-align: center;"><i>ABC PickList entity</i></p> <p>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.</p>
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” on page 118 .

- 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 Compile your modifications.

12 To designate the parent value, add LOV values to the Parent LIC column.

For more information, see [Table 50 on page 406](#) 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 [Table 51](#). 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 [Table 51](#) 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” on page 389](#).

Table 51 lists the values that belong to the LOV Type defined as ACCOUNT_STATUS.

Table 51. Example Values from the ACCOUNT_STATUS LOV Table

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. 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. If you must add a new set of values that you define as a new LOV Type, then you must use Siebel 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.
- 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” on page 118.
Translate	Do not modify this property. For more information, see “Modifying the Translate Property for a Predefined List of Values” on page 549.
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” on page 543.

- 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 for a Predefined List of Values” on page 549.
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”](#) on page 548. 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”](#) on page 389.
- 6 Compile and test your modifications.
Make sure you clear the cache. For more information, see [Step 7](#) on page 419.

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](#) on page 420
- [Guidelines for Using Script to Associate a List of Values with an Organization](#) on page 420
- [Creating a Value to Display for More Than One Organization](#) on page 421
- [Using the Organization Specifier Property to Display Custom Lists of Values](#) on page 421

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” on page 420](#). 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 [Step a](#) through [Step c](#) 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.

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

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 on page 423](#)
- [About Association Applets on page 433](#)
- [About Shuttle Applets on page 441](#)
- [Creating a Shuttle Applet on page 443](#)

For more information, see [“Creating an Applet” on page 323](#).

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 on page 423](#)
- [How Siebel CRM Creates a Multi-Value Group on page 426](#)
- [Guidelines for Creating Multi-value Group Applets and Pick Applets on page 430](#)
- [Creating a Multi-Value Group on page 430](#)
- [Creating a Multi-Value Group Applet on page 432](#)

For more information, see the following topics:

- [How Siebel CRM Sorts a Multi-Value Field on page 75](#)
- [About Links on page 104](#)
- [About Multi-Value Links on page 95](#)

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”](#) on page 30.

Viewing an Example of a Multi-Value Group Applet

You can view an example of a multi-value group applet.

To view an example of a multi-value group applet

- 1 In the Siebel client, choose the Account screen, and then the Accounts List.
- 2 In the Account Entry Applet, click the Select button that is located on the right side of the Address field.

Siebel CRM displays the Account Addresses multi-value group applet. This applet lists the detail Address records that Siebel CRM associates with the master account record. This dialog box lists the address information that is 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. You can view the list of addresses that Siebel CRM associates with the account while the Account Addresses multi-value group applet is open. 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”](#) on page 81.

Figure 47 describes the relationships and objects that Siebel CRM uses in a multi-value group applet.

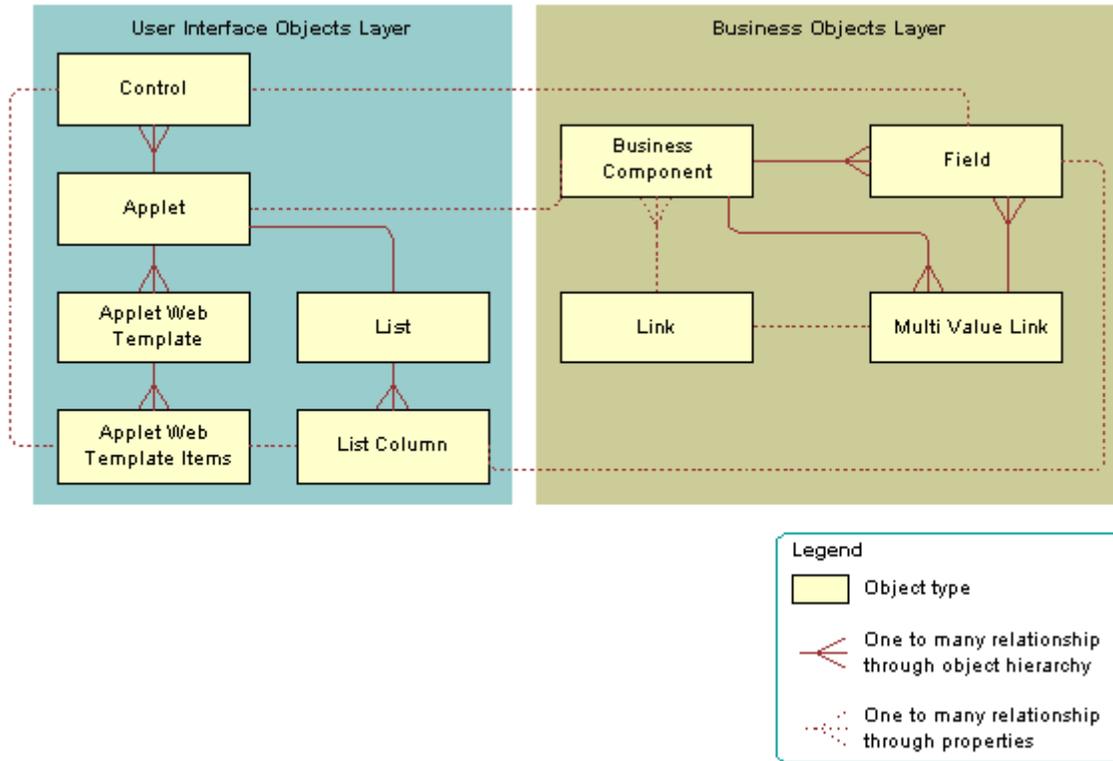


Figure 47. Relationships and Objects That Siebel CRM Uses in a View Multi-Value Group Applet

Properties of a Multi-Value Group Applet

Table 52 describes some of the properties of the multi-value group applet.

Table 52. 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

Figure 48 describes the objects and properties that Siebel CRM uses to create a multi-value group applet.

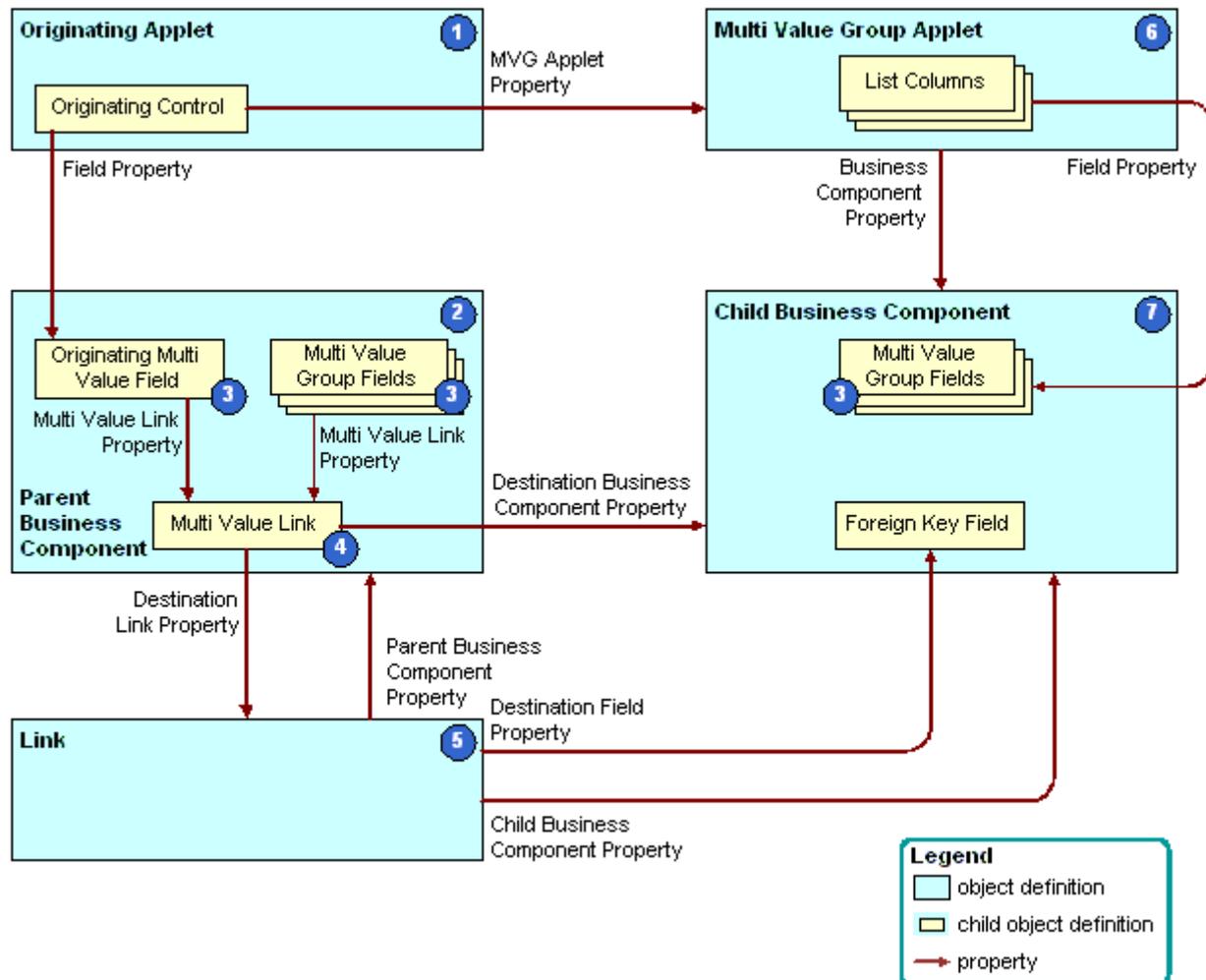


Figure 48. Objects and Properties Siebel CRM Uses to Create a Multi-Value Group Applet

Explanation of Callouts

Siebel CRM uses the following objects to create a multi-value group applet:

- 1 Originating applet.** Contains the control or list column that calls the multi-value group applet. For more information, see [“Originating Applet of a Multi-Value Group”](#) on page 428.
- 2 Parent business component.** Supplies data to the originating applet. For more information, see [“Parent Business Component of a Multi-Value Group”](#) on page 428.

- 3 **Multi-value fields.** Includes the fields that constitute a multi-value group. For more information, see [“About the Multi-Value Field” on page 98](#).

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 multi-value group references.
 - 5 **Link.** Specifies the parent-child relationship between the parent business component and the child business component that the multi-value group 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 multi-value group 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 multi-value group applet are the records of the child business component that the multi-value group references. For more information, see [“Child Business Component of a Multi-Value Group” on page 429](#).

Example of Objects Siebel CRM Uses to Create a Multi-Value Group Applet

Table 53 describes some of the objects that Siebel CRM uses to create a multi-value group applet for the Account Address Mvg Applet.

Table 53. Example of Objects Siebel CRM Uses to Create a Multi-Value Group Applet

Object	Name of Object Definition
Originating applet	Account Entry Applet
Parent business component	Account
Multi-value fields	<p>This example includes the following multi-value fields:</p> <ul style="list-style-type: none"> ■ Street Address ■ Address Id ■ City ■ Country ■ Fax Number ■ Postal Code ■ State

Table 53. Example of Objects Siebel CRM Uses to Create a Multi-Value Group Applet

Object	Name of Object Definition
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 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.

Table 54 describes some of the properties of the originating control or list column.

Table 54. Some of the Properties of the originating Control or List Column

Property	Description
Field	Identifies the originating field in the originating business component.
MVG Applet	Name of the multi-value group 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 references. The primary is the record where Siebel CRM gets these values.

The parent business component does not include any properties that Siebel CRM requires to define a multi-value group. The field and multi-value link 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 multi-value link. For more information, see [“About the Multi-Value Field” on page 98](#).

About the MVF 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 is a child of a multi-value field. Each pick map defines a relationship between a field in the child business component that the multi-value group references and one in the originating business component. If the user chooses 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 multi-value group business component.

Table 55 describes some of the properties of the MVF pick map.

Table 55. Some of the Properties of the MVF 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 multi-value group business component when it calls the PickRecord method.
Pick List Field	Identifies a field in the multi-value group business component that is 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 MVF pick map. The Account business component includes a multi-value link to the Business Address business component, where it gets address information.

For more information, see [“About Multi-Value Links” on page 95](#), and [“About Links” on page 104](#).

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” on page 252](#).

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” on page 154](#).

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” on page 30](#) and [“How Siebel CRM Creates a Multi-Value Group” on page 426](#).

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.

- d Repeat [Step a](#) through [Step c](#) 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” on page 30](#) and [“How Siebel CRM Creates a Multi-Value Group” on page 426](#).

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” on page 126 .
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” on page 127 .
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” on page 433](#).
- 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” on page 325](#).

- 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 *Siebel Release Notes* on My Oracle Support.

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” on page 116](#). 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.

About Association Applets

This topic describes association applets. It includes the following information:

- [Overview of Association Applets on page 433](#)
- [How Siebel CRM Creates an Association Applet on page 435](#)
- [How Siebel CRM Calls an Association Applet from a Master-Detail View on page 439](#)
- [Constraining an Association Applet on page 441](#)

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.

Figure 49 includes an example of how Siebel CRM implements a many-to-many relationship between two business components in the Siebel schema.

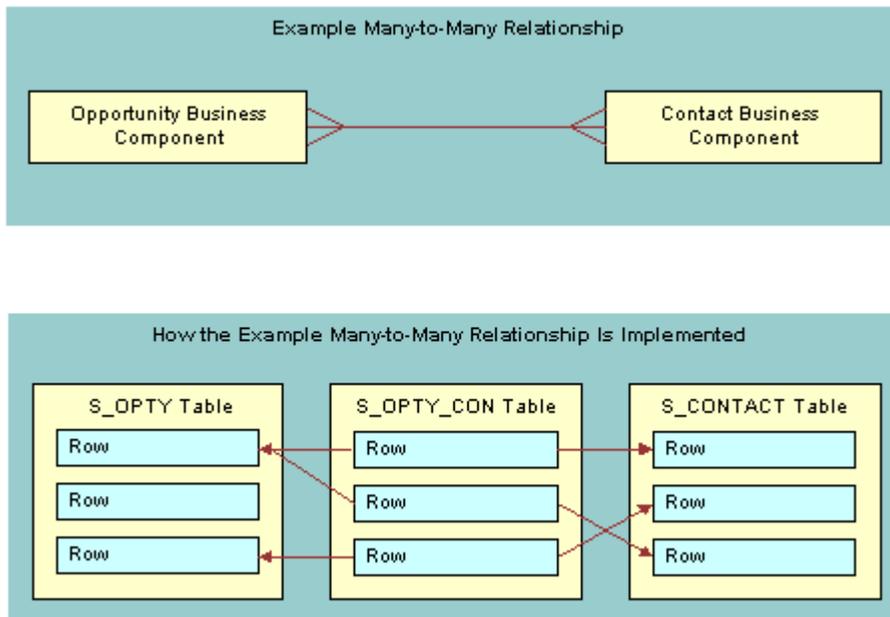


Figure 49. Example Many-to-many Relationship Between Two Business Components

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

Figure 50 describes the relationships and objects that Siebel CRM uses to create an association applet.

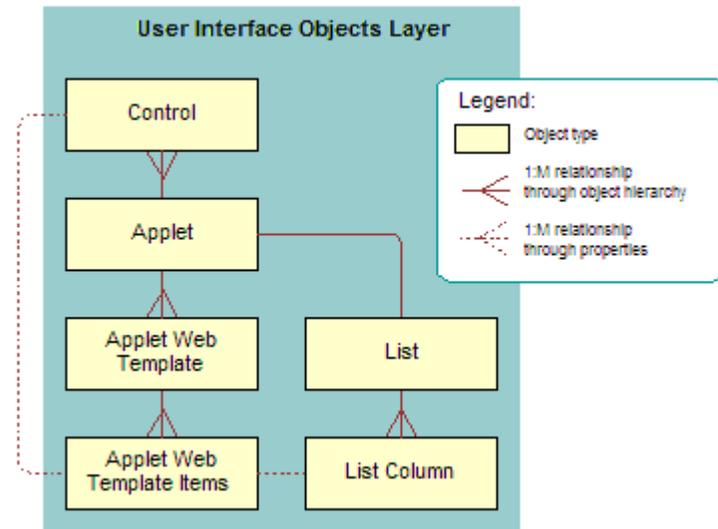


Figure 50. Relationships and Objects That Siebel CRM Uses to Create an Association Applet

Figure 51 describes a generic picture of how Siebel CRM creates an association applet.

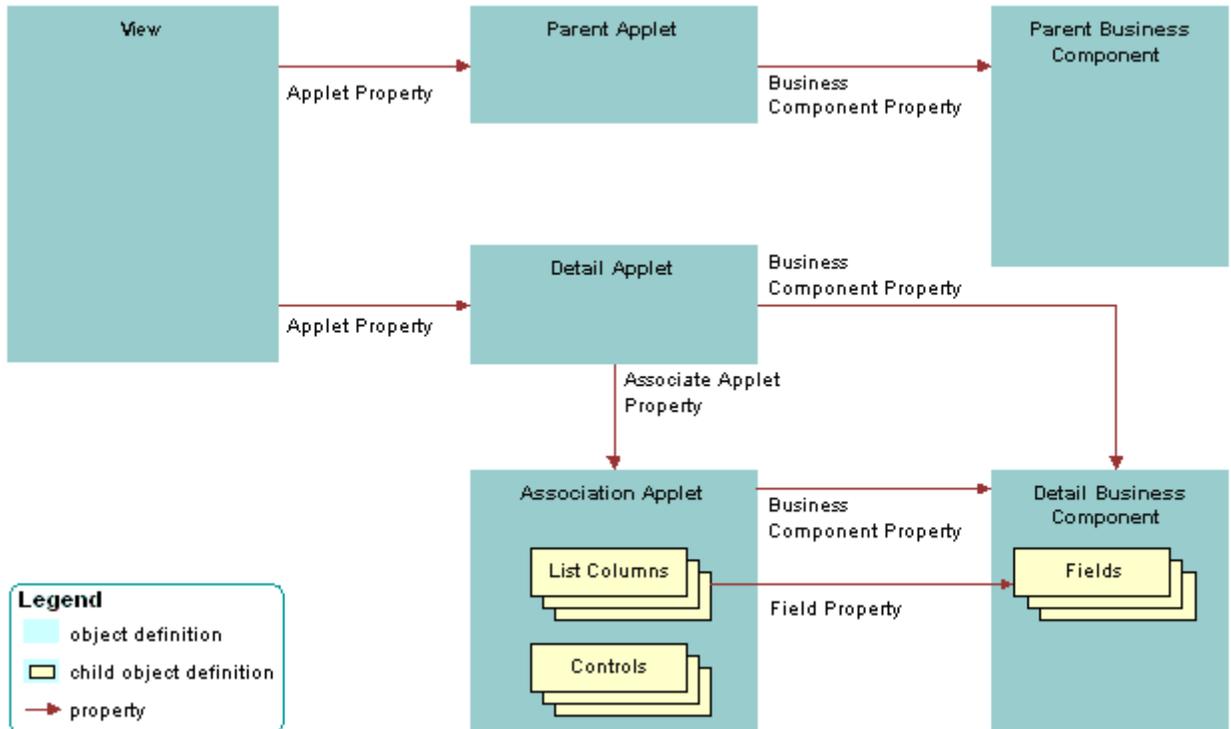


Figure 51. Generic Picture of How Siebel CRM Creates an Association Applet

Figure 52 includes an example of how Siebel CRM creates an association applet.

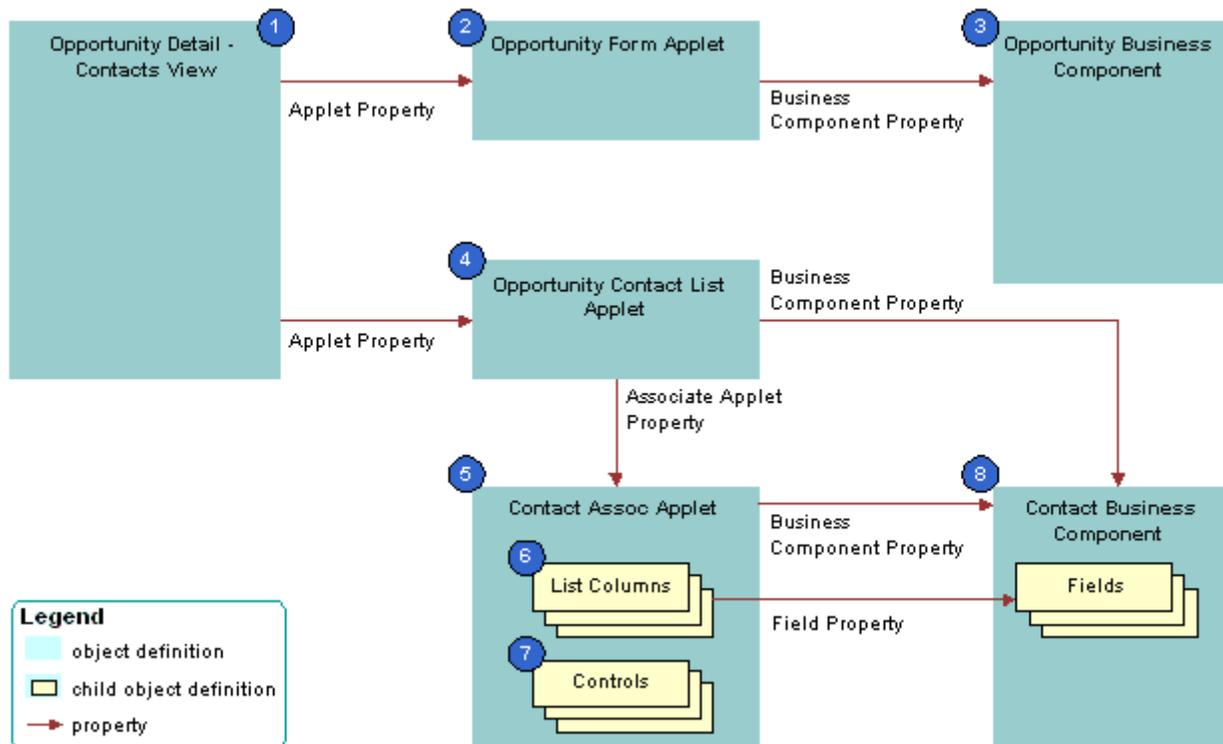


Figure 52. Example of How Siebel CRM Creates an Association Applet

Explanation of Callouts

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” on page 438.](#)
- 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

[Table 56](#) describes specialized controls Siebel CRM can display in an association applet. For more information, see [“Caution About Using Specialized Classes” on page 74.](#)

Table 56. Specialized Controls That Siebel CRM Can Display in an Association Applet

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 <code>PopupQueryAdd</code> and includes an <code>AddRecord</code> 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 <code>ButtonNew</code> and it includes a <code>NewRecord</code> 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

Figure 53 describes how Siebel CRM creates an association applet that it calls from a multi-value group applet.

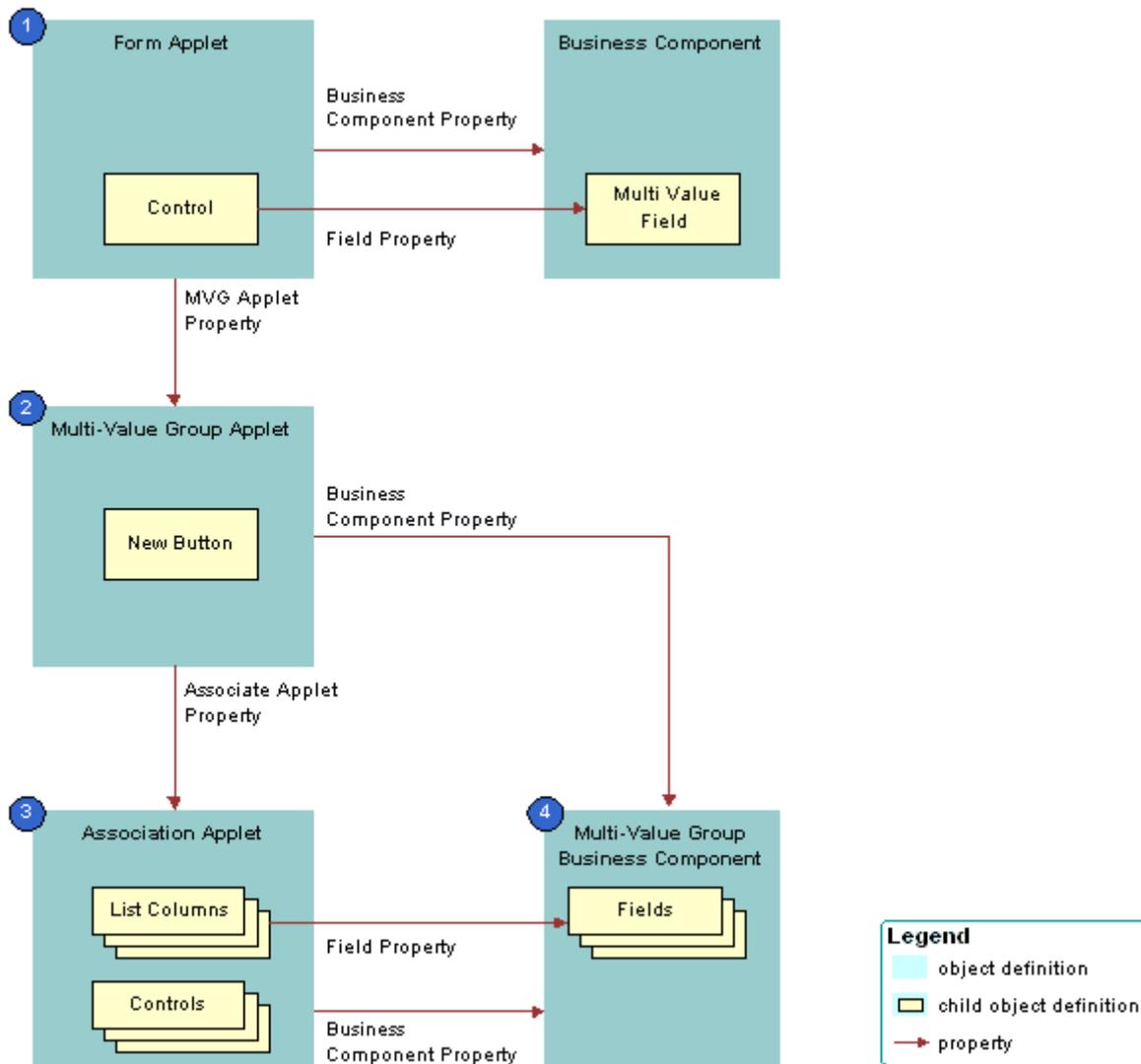


Figure 53. How Siebel CRM Creates an Association Applet It Calls from a Multi-Value Group Applet

Explanation of Callouts

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” on page 440](#).

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 on the left side of the shuttle applet.

A shuttle applet gets the following items from the multi-value group applet:

- Selected label.
- List body that Siebel CRM displays on the right side of 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:

- CCPopupListAssoc.swt
- CCPopupListMVG.swt

For more information, see [“Caution About Using Specialized Classes” on page 74](#).

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” on page 155](#).

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 on page 443](#)
- 2 [Creating the Multi-Value Group Applet on page 444](#)
- 3 [Creating the View on page 446](#)

Creating an Association Applet

This task is a step in [“Creating a Shuttle Applet” on page 443](#).

Siebel CRM displays the association applet on the left side of 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 using values from the following table.

Property	Description
Project	Choose the locked project where you must create the association applet.
Applet Name	Enter Create Contact Access List Assoc. For more information, see “Guidelines for Naming an Applet” on page 126 .
Display Title	Enter All Employees. For more information, see “Guidelines for Creating an Applet Title” on page 127 .
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” on page 116](#).

- 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
High Interactivity Enabled	Y

- 13 Save your modifications.

Creating the Multi-Value Group Applet

This task is a step in [“Creating a Shuttle Applet” on page 443](#).

In a shuttle applet, Siebel CRM displays the multi-value group applet on the right side of 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.

- 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" on page 126.
Display Title	Enter Team Members. For more information, see "Guidelines for Creating an Applet Title" on page 127.
Business Component	Choose Contact.
Upgrade Behavior	Choose Admin.

- 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"](#) on page 116.

- On the Web Layout - Fields page, choose the following fields, and then click Next:
 - SSA Primary Field
 - First Name
 - Last Name
- In the second Web Layout - Fields dialog box, remove Query Assistant from the list of chosen controls.
- Click Next, and then click Finish to create the applet.
- In the Object Explorer, click Applet.
- 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

- 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
High Interactivity Enabled	Y

12 Drag and drop 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 on the far left.
- b Drop the PositionOnRow control to the left of SSA Primary Field.

13 Save your modifications.

Creating the View

This task is a step in “Creating a Shuttle Applet” on page 443.

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 Vi ew.
View Title	Enter ABC Contact Team Vi ew.

Property	Description
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 then drag it to one of the empty side-by-side placeholders.

- 14 Compile and test your modifications.
 For more information, see *Using Siebel Tools*.

20 Configuring Menus, Toolbars, and Icons

This chapter describes how to configure menus, toolbars, and icons. It includes the following topics:

- [About Menus and Toolbars on page 449](#)
- [Configuring Menus and Toolbars on page 455](#)
- [Configuring Icons on page 461](#)

For more information, see [“Localizing an Application Menu” on page 541](#).

About Menus and Toolbars

This topic describes menus and toolbars. includes the following information:

- [Objects That Siebel CRM Uses in a Menu or Toolbar on page 450](#)
- [About the Method, Business Service, and Target Properties of the Command Object on page 451](#)

Menus and *toolbars* are user interface elements that allow the user to do something. For example:

- The *application menu* 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 top 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* 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 [Figure 54](#). Some icons on the application toolbar are redundant with menu items that the application menu contains.

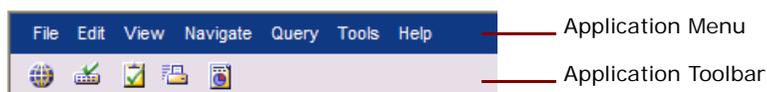


Figure 54. Menu and Toolbar in Siebel CRM

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 upper left corner of an 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” on page 479](#) 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” on page 455](#) and [“Properties of a Command” on page 608](#).

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” on page 610](#).

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” on page 611](#).

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” on page 611](#).

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” on page 612](#).

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” on page 612](#).

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

Table 57 summarizes the Target and Business Service properties.

Table 57. Summary of 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 on page 455](#)
- [Creating a New Toolbar on page 456](#)
- [Adding a New Toolbar Icon to a Predefined Toolbar on page 456](#)
- [Activating Menu Items and Toolbars on page 457](#)
- [Creating an Applet Menu on page 458](#)
- [Activating or Suppressing an Applet Menu Item on page 459](#)
- [Using JavaScript to Configure a Toolbar on page 460](#)

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” on page 451](#).

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” on page 192](#).
- 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” on page 149](#) 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” on page 192](#).
- 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” on page 462](#). 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" on page 608](#).
- 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" on page 611](#), and Article ID 517909.1 on My Oracle Support.

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" on page 451](#).

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.

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.

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 Web Server Extension:


```
ORACLE_HOME\scri pts
```
- 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” on page 192](#).
- 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, BarcodeTool bar.
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, barcodeTool bar. j s.

- 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, JSSBarcodeTool bar.
Project	Choose the project that you defined in Step 3 .

Property	Value
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” on page 456](#).

- 7 Add new toolbar items.

For more information, see [“Adding a New Toolbar Icon to a Predefined Toolbar” on page 456](#).

- 8 If you create a new toolbar, then add a swe:toolbar tag to the appropriate web template.

Make sure you set the name property that you define in the swe:toolbar tag to the name of the Toolbar object that you created in [Step 6](#). For more information, see [“Using Web Templates to Configure Toolbars” on page 479](#).

- 9 Add swe:toolbaritem tags to the swe toolbar tag.

For more information, see [“Using Web Templates to Configure Toolbars” on page 479](#).

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 on page 462](#)
- [Configuring a Bitmap Category and a Bitmap on page 463](#)
- [Displaying an Icon on a Button on page 464](#)
- [Displaying an Icon as a Link on page 465](#)
- [Using Icons to Represent Values in a Field on page 465](#)
- [Configuring Icons in a Tree Applet on page 467](#)

Overview of Configuring Icons That Siebel CRM Displays in Siebel Clients

Table 58 describes the object types that Siebel CRM uses to display images in the Siebel client.

Table 58. Object Types Siebel CRM Uses to Display Images

Object Type	Description
Bitmap	<p>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:</p> <ul style="list-style-type: none"> ■ 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. <p>A bitmap includes the following properties that Siebel CRM commonly uses:</p> <ul style="list-style-type: none"> ■ 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 <code>img</code> 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 <code>alt</code> attribute of the image tag. <p>You do not use the other properties of the bitmap with a Web image. Example properties include Data and Transparent Color.</p>
Bitmap Category	<p>Allows you to group image files together according to function. Includes the following bitmap categories:</p> <ul style="list-style-type: none"> ■ 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	<p>Allows you to configure Siebel CRM to display an image in a hierarchical applet, such as a tree applet.</p>
Icon Map	<p>Allows you to configure Siebel CRM to display an image for a field value. Includes the child Icon object type.</p>

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 public\images folder of your Siebel 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. It defines these objects in the configuration file that it uses for the Siebel application.

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” on page 192.](#)

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” on page 463.](#)
- 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: <ul style="list-style-type: none"> ■ To create a bitmap for a BMP file, leave the File Name property empty. ■ 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: <ul style="list-style-type: none"> ■ That resides in the eapps/publ i c/i mages folder, set the File Name property to i mage. gi f. ■ That resides in the eapps/publ i c/i mages/bt tns folder, set the File Name property to bt tns/i mage. gi f.
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 do a compile.

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 upper-right corner 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” on page 463](#).
- 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” on page 113](#).

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” on page 464](#), 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 SWF (Siebel Web Format) file, then the Siebel Web Engine uses the bitmaps:

```
swe: this property="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” on page 463](#).
- 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” on page 463](#).

- 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” on page 467](#).

- 11 Compile and test your modifications.

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” on page 465](#).

- 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.

You can configure other graphic elements in the tree applet. For more information, see [“Configuring the Graphic Elements of a Tree Applet” on page 372](#).

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” on page 463](#).
- 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” on page 469](#).
- 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” on page 469](#).

- 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” on page 363](#).
- 9 Compile and test your modifications.
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

Table 59 describes properties that Siebel CRM commonly uses with an HTML hierarchy bitmap.

Table 59. 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).
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 and Siebel Tags

This chapter describes how to configure Siebel web templates and Siebel tags. It includes the following topics:

- [Configuring Siebel Web Templates and Siebel Tags on page 471](#)
- [Configuring Web Templates to Display Menus, Toolbars, and Thread Bars on page 477](#)
- [Configuring an HTML Control Type on page 483](#)

Configuring Siebel Web Templates and Siebel Tags

This chapter describes how to configure Siebel web templates. It includes the following information:

- [Editing the Layout of a Web Page on page 471](#)
- [Adding Graphics to a Web Template on page 472](#)
- [Displaying Multiple Views on a Page on page 473](#)
- [Configuring How Siebel CRM Displays an Error That Occurs on the Siebel Server on page 475](#)

For other tasks that use web templates and tags, see the following topics:

- [Displaying Totals for a List Column in an Applet on page 338](#)
- [Using a Control to Allow the User to Click a Link to Activate a Record on page 343](#)
- [Configuring the Sort Order for Siebel CRM on page 298](#)
- [Using JavaScript to Configure a Toolbar on page 460](#)

For more information, see [Chapter 8, "About Siebel Web Templates and Siebel Tags."](#)

Editing the Layout of a Web Page

The Web Page Object is the top 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" on page 268.](#)

To edit the layout of a web page

- 1 Make sure the configuration context is set.
For more information, see [“Setting the Configuration Context” on page 304](#).
- 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 drag 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 swe:if or swe: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:

publ i c\i mages

The Siebel Web Server Extension installer creates the publ i c\i mages directory when you install the SWSE. The Siebel client includes three directories for a Siebel application. These directories contain all the files that Siebel CRM uses, including the graphics files.

- 2 Create a link to the graphic file from an HTML page.
- 3 Compile and test your modifications.

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” on page 267](#) and *Configuring Siebel Open UI*.

To display multiple views on a page

- 1 Replace the first line of the swe:frameset code of the CCPageContainer.swt file with the following example code:

```
<swe: frameset html Attr="rows=' 80, 50, 50, *' border=' 0' frameborder=' No' ">
```

For more information, see [“Guidelines for Naming a Siebel Web Template” on page 178](#).

- 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:

```
<swe: frame type="Content" html Attr="marginheight=' 0' marginwidth=' 0' noresize scrolling=' Yes' ">
```

```
<swe: include file="CCMainView.swt"/>
```

```
</swe: frame>
```

For more information, see [“Example Code of the CCMainView.swt File” on page 475](#).

- 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" on page 474](#).

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 and not the SWT file name. For example, to display the search view with the main view, do the following:

- Create a content container page, such as CCSMainAndSearchView.swt.
- Use the LoadContentContainer method to load this page.

To load the main view and search view into two frames, the CCSMainAndSearchView.swt file contains the following tags:

```
<swe: frameset htmlAttr="cols=' 100%' border=' 0' frameborder=' No' ">
  <swe: frame type="Vi ew" htmlAttr="noresize scrolling=' Yes' ">
    <swe: current-view/>
  </swe: frame>
  <swe: frame type="Al tVi ew" name="Search" htmlAttr="noresize scrolling=' Yes' ">
    <swe: vi ew name="Search Vi ew" id="Search" />
  </swe: frame>
</swe: frameset>
```

In this example, you still reference the main view in the swe:current-view tag. You reference alternate views in the swe:view 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.swt file.

SWE View Tag

The swe:view tag uses the following format:

```
<swe: vi ew name="xxx" id="yyy">
```

The swe:view 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 swe:frame 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.swt File

The CCMainView.swt file that you reference in the swe:frameset code of the CCPageContainer.swt page includes the following frameset. This frameset contains the main view:

```
<swe:frameset htmlAttr="cols=' 100%' border=' 0' frameborder=' No' ">
  <swe:frame type="View" htmlAttr=" noresize scrolling=' Yes' ">
    <swe:current-view/>
  </swe:frame>
</swe:frameset>
```

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 swe:error 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 swe:error 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 swe:error tag.
- The only attribute of the swe:error 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 swe:error tag in a Siebel web template file, 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 swe:form tag:

```
<swe: error>  
    <swe: this property="FormattedHtml " />  
</swe: error>
```

- 2 Make sure you use this tag in all swe:form tags.

Format of the Swe:error Tag

The basic format of the swe:error tag is as follows:

```
<swe: error>
```

The format of the swe:error tag with the FormattedHtml property is as follows:

```
<swe: error property="FormattedHtml " />
```

or

```
<swe: error>  
    <swe: this property="FormattedHtml " />  
</swe: error/>
```

Example of Using the Error Tag

The following code is an example of using the swe:error tag:

```
<swe: form>  
    <swe: error>  
        <b><font color="red"> <swe: this property="FormattedHtml " /> </font></b>  
    </swe: error>  
    ...  
</swe: form>
```

For another example, see [“Example Code of a Nongrid Form Applet Template” on page 157](#).

Usage of the SWEErrMsg Item

You must use the swe:error tag instead of the swe:pageitem tag that is mapped to the _SWEErrMsg item in the Error Web Page property of the Siebel application. Siebel CRM version 7.0 and higher does not support _SWEErrMsg.

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” on page 449](#). It includes the following information:

- [Using Web Templates to Display Menus and Buttons on page 477](#)
- [Using Web Templates to Configure Toolbars on page 479](#)
- [Using Web Templates to Configure the Thread Bar on page 481](#)

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 swe:menu tag to activate a list of menu items. The swe:menu tag displays menus in the following ways:

- Application container page for an application menu
- Applet for a applet menu

Siebel CRM displays an applet menu as an icon button, typically placed to the left of other buttons, such as Edit and Delete. In the Siebel client, it uses the configuration in the SRF file 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 swe:menu 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 swe:menu 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.swt file includes the swe:menu tag at the start of the definition for a banner:

```
<!--Start Banner-->
<swe:menu/>
<table class="banner" cellpadding="0" cellspacing="0" border="0" >
<tr>
  <td width="50%">
    
  </td>
  <td width="50%">
```

```
        
    </td>
</tr>
```

How the Menu Tag Displays a Menu

The swe:menu 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 swe:menu tag uses the following format:

```
<swe:menu type="XXX" bitmap="XXX" width="XXX" height="XXX" bgcolor="XXX"
fgcolor="XXX" />
```

The swe:menu 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” on page 541](#).
- **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.swt file 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>
        <swe:menu/>
```

```
</td>

<td valign="middle"></td>

<!-- EditRecord -->
<swe:control id="132">
  <td valign="middle" nowrap>
    <swe: this property="FormattedHtml" hintText="Edit" hintMapType="Control" />
  </td>
  <td>&nbsp;</td>
</swe:control >
(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.

SWE Tag That Specifies a Toolbar

The `swe:toolbar` 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 `swe:toolbar` tag uses the following format:

```
<swe: toolbar name="XXX" javaapplet="true/false" width="XXX" height="XXX" />
```

The `swe:toolbar` 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 swe:toolbaritem 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 swe:toolbaritem tag uses the following format:

```
<swe: toolbaritem>
```

The swe:toolbaritem 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 file:

```
<swe: toolbar name=xxx> // where xxx is the name of toolbar in the repository.  
    // any HTML stuff here...  
    <swe: toolbaritem>  
    // any HTML stuff here...  
</swe: toolbar>
```

- 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 file:

```
<swe: toolbar 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.

Table 60 describes how Siebel CRM responds to actions the user performs in the thread bar. For more information, see “Configuring the Thread Bar” on page 271.

Table 60. How Siebel CRM Responds to User Actions in 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 to the right of the thread button that the user clicked and proceeds to the step view that <code>SWEBMCount</code> 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=2SWECCount=3
```

where:

- `SWEBMCount=2` indicates that Siebel CRM uses bookmark number 2 to create the view.
- `SWECCount=3` is the bookmark ID for the current view.

For example, Siebel CRM uses the swe tags and thread link format to translate the thread button for the A.K. Parker account into the following HTML format:

```
<a href="www.siebel.com/start.swe?SWECmd=GotoBookmarkView&SWEBMCount=2&SWECCount=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 swe count increased by 1. The swe count is the count that Siebel CRM sends to the Siebel Server in the request.

Bookmark deletion policy is not modified with the above bookmark ID assignment policy. Siebel CRM keeps the most recently created 20 bookmarks and deletes all other bookmarks, by default. If the swe count in the user request is less than the swe count on the Siebel Server, then Siebel CRM deletes all the bookmarks that contain a swe count that is larger than the swe count in the user request.

Configuring the Thread Bar

You can use the following swe tags to configure an HTML thread bar:

- **swe:threadbar**. Defines the start and finish of the thread bar section.
- **swe:threadlink**. 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.
- **swe:stepseparator**. Specifies the symbol that Siebel CRM displays to separate thread buttons.

Use the swe:threadlink and swe:stepseparator tags only in the swe:threadbar tag.

The usage of these swe 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 file. Use the swe:threadbar, swe:threadlink, and swe:stepseparator tags. For usage with frames, do the following:
 - **Application does not use frames**. Insert the definition in a container page. For example, CCPageContainer.swt.
 - **Application uses frames**. Insert the definition in the Siebel web template file 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 file:

```
<!-- Begin Threadbar section -->  
<table class="Threadbar" width=100% border="0" cell spacing="0" cell padding="0">  
<tr valign="left">
```

```

<td nowrap bgcolor="#6666CC" width=110>
    
</td>
<td width=99%>
    <swe: threadbar>
        
        <swe: threadlink property="FormattedHtml">
            <font color="#000000"> <span >&nbsp; <nobr><swe: this property="Title"/></
            nobr>&nbsp; </span> </font>
        </swe: threadlink>
        <swe: stepseparator>&gt; </swe: stepseparator>
    </swe: threadbar>
    
</td>
</tr>
</table>
<!-- End Threadbar section -->

```

Configuring an HTML Control Type

This topic describes how to configure an HTML control type. includes the following information:

- [Creating a New HTML Type on page 484](#)
- [How the Siebel Web Engine Uses a Custom HTML Type on page 485](#)
- [Configuring an HTML Type on page 485](#)

Siebel CRM supports different control types. For example, Check Box, Button, Mail To, Text Area, and so on. You can add a definition to the SWF (Siebel Web Format) file to define your own HTML type.

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 `chtmltype.swf` file 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 file. You can define types in the `chtmltype.swf` file that 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 one of the SWF files that Siebel CRM uses:

- a Add the format information for the new type.

Siebel CRM uses two SWF files. The file you modify must use the SWF extension. It must reside in the same directory as the template files. One file contains the special types that Siebel CRM defines. The other file contains custom definitions that you define to add more types or to override Siebel types.

Use the `swe:htmltype` and `swe:this` tags to define how to display the custom type using the following format:

```
<swe:html type name="XXX" mode="AAA" state="BBB">
    . . . . . HTML . . . . .
    <swe: this property="YYY" />
    . . . . More HTML . . . .
</swe:html type>
```

- b Set the UserSWFName parameter to the name of the SWF file 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 swe:control tag or the swe:pageitem 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 SWF file, the swe:this tag can reference these properties, except the FormattedHtml property.

Note the following examples:

- `<swe: control id="1" property="FormattedHtml" />`
- `<swe: control id="1"> ... <swe: this property="FormattedHtml" /> ... </swe: control >`
- `<swe: pageitem id="1" property="FormattedHtml" />`
- `<swe: pageitem id="1"> ... <swe: this property="FormattedHtml" /> ... </swe: pageitem>`

Format Requirements of an SWF File

You must make sure each format specification in an SWF file 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 swe:this 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: html type name="Label Red">
  <font color="red"> <swe: this property="Di spl ayName" /> </font>
```

```
</swe:html type>
```

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:html type name="Siebel Text">
  <swe: this property="Data" type="Text"/>
</swe:html type>

<swe:html type name="Siebel Text" mode="Edit">
  <swe: this property="DisplayName"/> &nbsp;  <swe: this property="Data" type="Text"/>
</swe:html type>
```

For more information, see [“Options to Control How the User Creates, Edits, Queries, and Deletes CRM Data” on page 116](#).

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:html type name="Mini Button" state="Disabled">
  
  <swe: this property="Data" type="Link" />
  
</swe:html type>
```

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 SWF file 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:

```
swe: this 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:html type name="Mini Button" state="Disabled"></swe:html type>
```

This code hides only the swe:control tag or the swe:this 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:html type name="Siebel Text" mode="Edit" state="Required">
    *&nbsp;
    <swe: this property="DisplayName" />
    : &nbsp;
    <swe: this property="Data" type="Text" />
</swe:html type>
```

The Siebel Web Engine uses the following order of precedence when it looks up HTML Type definitions in the SWF file:

- 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 SWE *This* Tag

The Data property of the swe:this 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 swe:this 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 Web Engine intrinsic type named Link:

```
<swe:html type name="Mini Button">
```

```
<img SRC="images/btn_left.gif" border=0 height=15 width=2>  
<swe: this property="Data" type="Link" />  
  
</swe: html type>
```

The following code outputs the same HTML as if the template included a separate swe: this tag, where the property is FormattedHtml and the HTML type of the control is the predefined Link type:

```
swe: this 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 Business Applications

This chapter describes how to tune and improve the performance of a Siebel application. It includes the following topics:

- [Using the Case Insensitivity Wizard to Improve Query Performance on page 489](#)
- [Improving the Performance of a Siebel Application on page 503](#)

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 on page 490](#)
- [Overview of the Case Insensitivity Wizard on page 491](#)
- [Variables You Can Use with the Case Insensitivity Wizard on page 493](#)
- [Using the Case Insensitivity Wizard on a Table on page 496](#)
- [Using the Case Insensitivity Wizard on a Table Column on page 498](#)
- [Using the Case Insensitivity Wizard on Columns That Do Not Contain an Index on page 498](#)
- [Using the Case Insensitivity Wizard to Do Various CIAI Configuration Tasks on page 500](#)
- [Using the Case Insensitivity Wizard to Deactivate CIAI Configuration on page 501](#)
- [Choosing the Correct Repository when Running the Case Insensitivity Wizard on page 501](#)
- [Limiting the Length of Schema Object Names Manually on page 501](#)
- [Other Configurations to Set Case Sensitivity on page 502](#)

For more information, see “How a CIAI Index Can Improve a Query” on page 490.

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” on page 624](#).

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” on page 497](#).
- Validates that all tables and columns are eligible for CIAI configuration. For more information, see [“How the Case Insensitivity Wizard Verifies Eligibility” on page 491](#).
- 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” on page 494](#).
- 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 does not create columns or indexes in the Siebel database until you compile the Siebel repository.

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” on page 498](#).

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” on page 518](#) and [“Objects You Use with Enterprise Integration Manager” on page 617](#).

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 the leftmost underscore.
- Deletes vowels one at a time, beginning with the rightmost vowel.
- Deletes characters one at a time, beginning with the rightmost 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” on page 501](#).

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 rightmost 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” on page 494](#).

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.

[Table 61](#) 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.

Table 61. Index Strategies You Can Use With the Database Method

Index Strategy	Work That the Case Insensitivity Wizard Performs
None	Defines no new CIAI columns or indexes.

Table 61. Index Strategies You Can Use With the Database Method

Index Strategy	Work That the Case Insensitivity Wizard Performs
Single	<p>Defines a new CIAI column and defines a single CIAI index on it.</p> <p>For every index that includes the base column, the wizard does not create another index that references the CIAI column.</p>
Copy All	<p>Defines a new CIAI column and a CIAI index for the column.</p> <p>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.</p>

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” on page 497](#).

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” on page 501](#).

- 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 tool s\obj ects 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” on page 497](#).

- 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.

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” on page 501](#).

- 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 on page 496](#).

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(Smi th)
```

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” on page 501](#).

- 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 [Table 62](#) to determine how to run the wizard.

Table 62. Running the Case Insensitivity Wizard to do Various CIAI Configuration Tasks

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.
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: <ul style="list-style-type: none"> ■ 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. ■ 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 on page 503](#)
- [Defining the Primary ID Field of a Multi-Value Link on page 504](#)
- [Modifying Custom Search Specifications on page 507](#)
- [Using Declarative Configuration to Enable a Button on page 508](#)
- [Improving Performance When Using Applet Toggles on page 508](#)
- [Deactivating Unused Screens on page 508](#)
- [Considering Factors That Affect Chart Performance on page 509](#)
- [Considering Factors That Affect MLOV Performance on page 509](#)

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” on page 506](#), and [“How Siebel CRM Creates a Multi-Value Group” on page 426](#).

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” on page 504](#).

- 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” on page 506](#).

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” on page 98](#).
 - 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 on page 95](#)
- [How Siebel CRM Creates a Multi-Value Group on page 426](#)
- [Creating Multi-Value Groups and Multi-Value Group Applets on page 423](#)
- [System Fields of a Business Component on page 89](#)

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” on page 506](#).

- 5** Set the Auto Primary property.

For more information, see [“Configuring the Auto Primary Property of a Multi-Value Link” on page 506](#).

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 [“About the Selected Option of the Auto Primary Property” on page 506.](#)
- **NONE.** The user must use the pick map to include the Primary Owner Id manually. For more information, see [“Configuring Siebel CRM to Not Override the Pick Map Value” on page 507.](#)

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 error 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” on page 89.](#)

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.

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 recompile the SRF file. 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 to deactivate the screen. This configuration requires you to compile the SRF file. When you compile, Siebel CRM does not include the inactive screen in the SRF file.

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 bottom 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

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 on page 511](#)
- [Mapping a Custom Table to an Interface Table on page 518](#)

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 on page 511](#)
- [Object Types That Enterprise Integration Manager Uses on page 512](#)

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 on page 58](#)
- [Guidelines for Configuring a Foreign Key That Affects Enterprise Integration Manager on page 66](#)
- [Adding an Extension Column to a Base Table on page 236](#)
- [Guidelines for Using Enterprise Integration Manager with an MLOV on page 554](#)

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 begins 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” on page 518](#).

Object Types That Enterprise Integration Manager Uses

Figure 55 describes the objects and relationships that EIM uses.

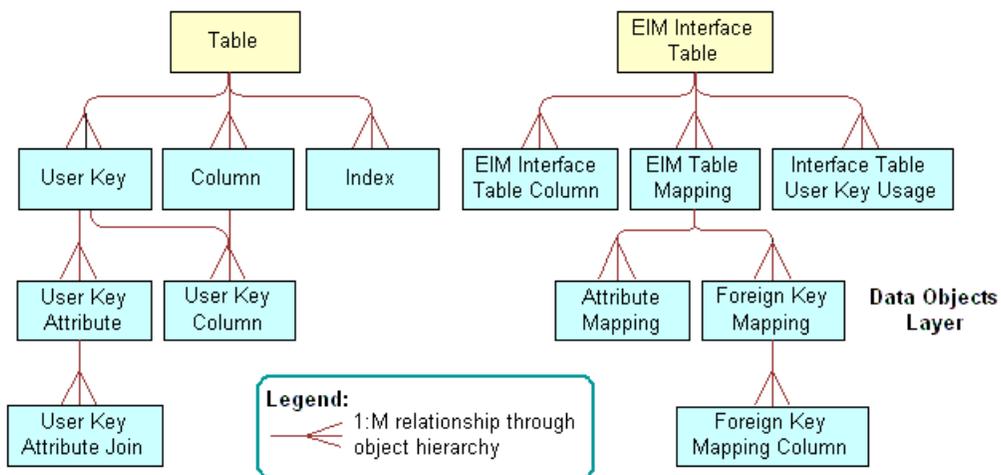


Figure 55. Objects and Relationships That EIM Uses

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.

Figure 56 describes 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.

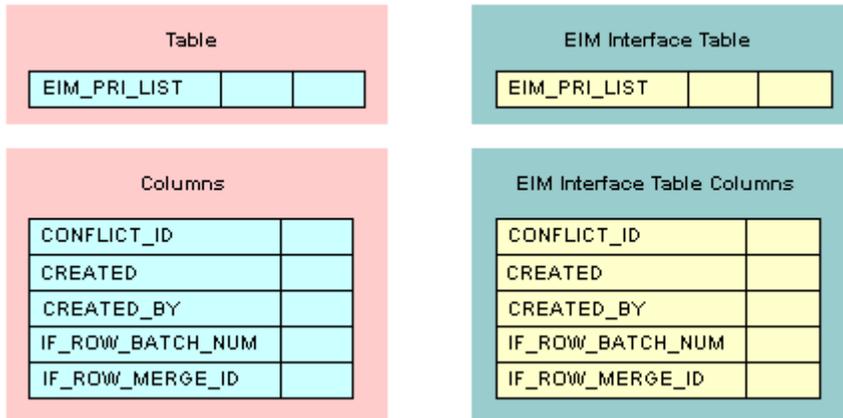


Figure 56. Child Columns of an Interface Table Are the Same as Child Columns of a Table

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.

Figure 57 describes how the Destination Table property of each EIM table mapping object identifies the name of the data table to update.

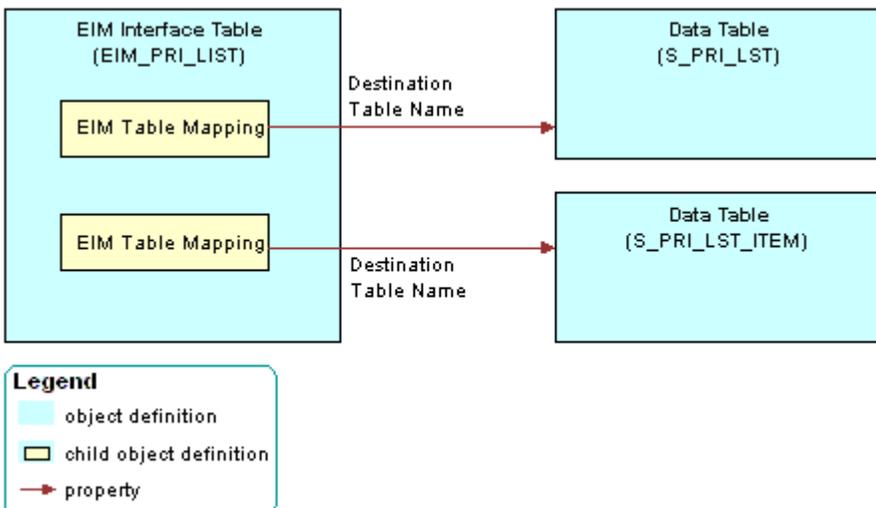


Figure 57. Example of How an EIM Interface Table References a Data Table

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.

Figure 58 includes an example of how an EIM table mapping references a data table.

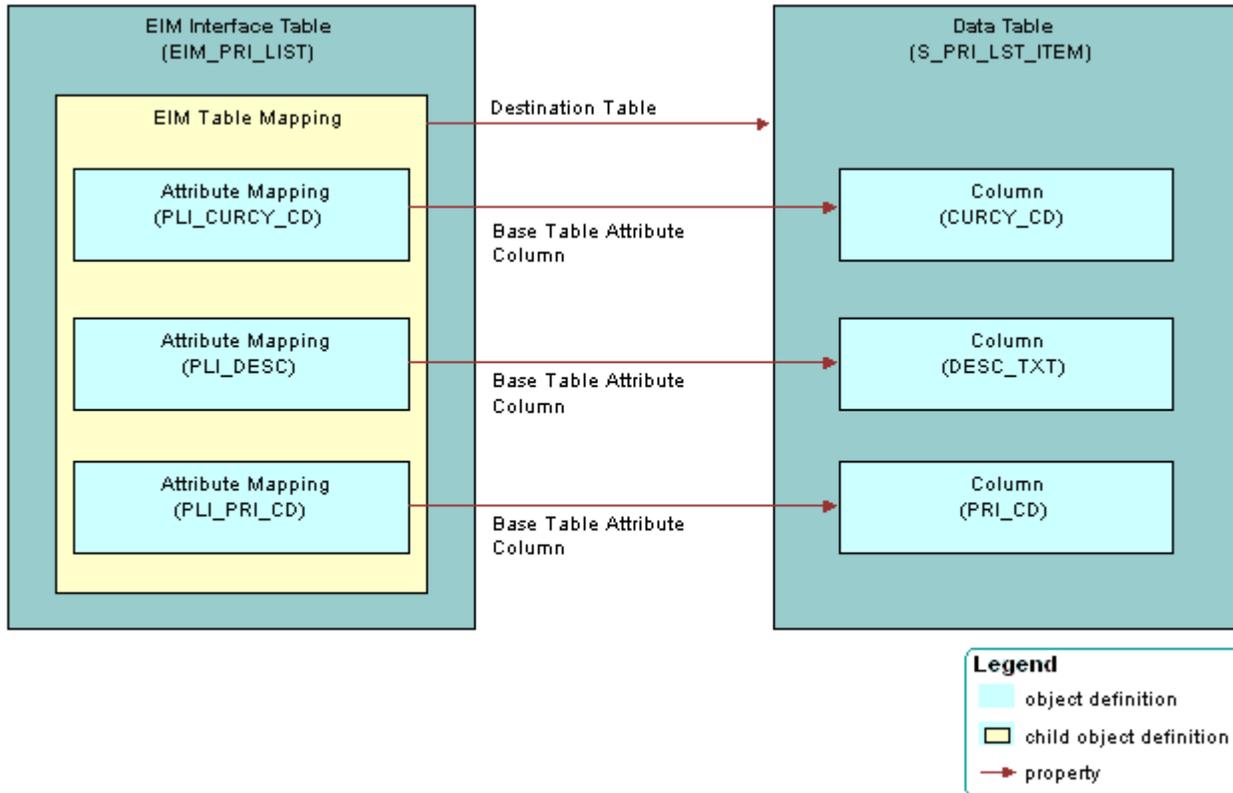


Figure 58. Example of How an EIM Table Mapping References a Data Table

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.

Figure 59 includes 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.

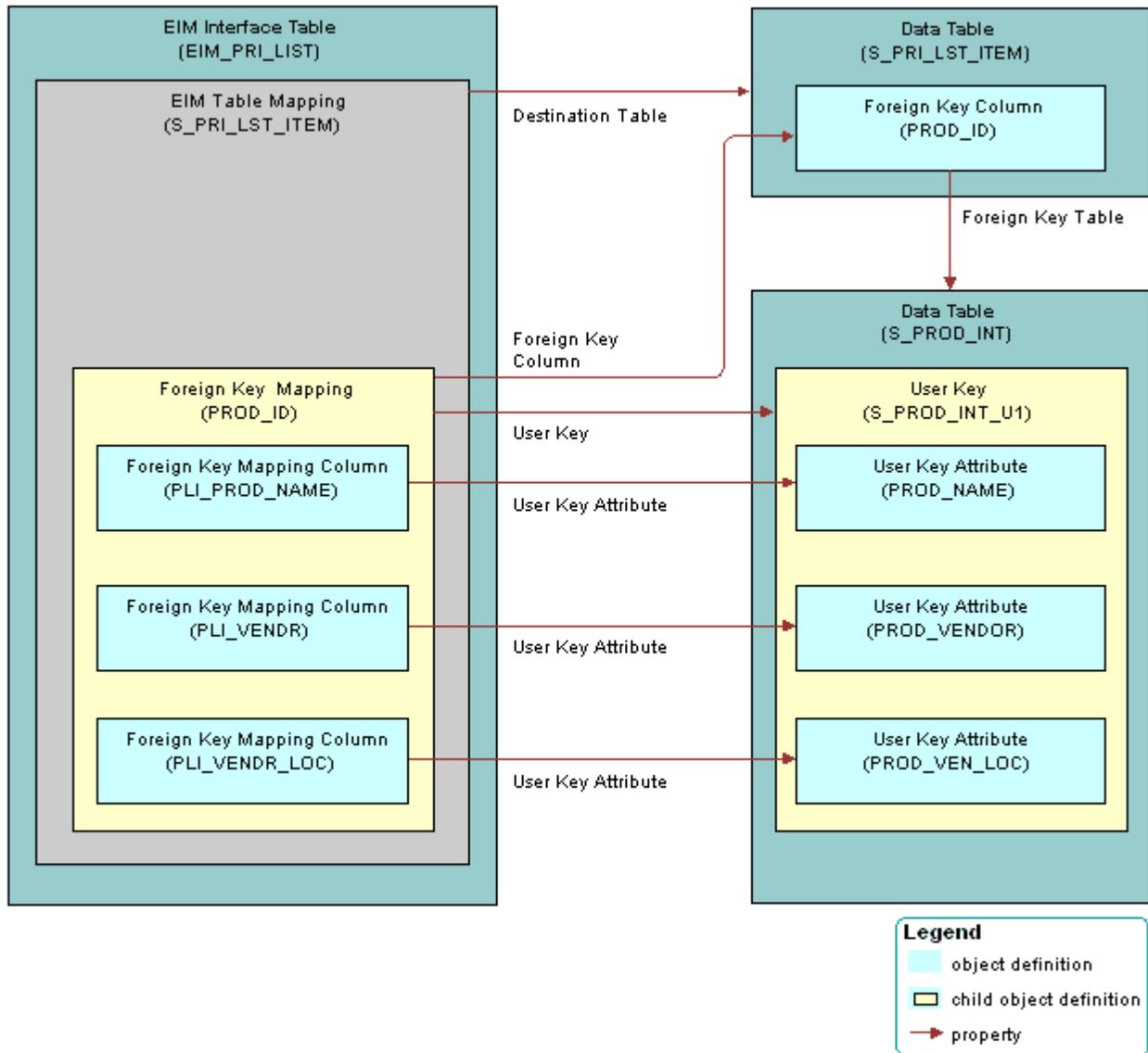


Figure 59. Example of How a Foreign Key Map References a Data Table

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” on page 57](#).

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 on page 518](#)
- [Relations That the EIM Table Mapping Wizard Creates on page 520](#)
- [Guidelines for Using the EIM Table Mapping Wizard on page 520](#)
- [Starting the EIM Table Mapping Wizard for a Table That Does Not Use the Foreign Key on page 521](#)
- [Deactivating Instead of Deleting an EIM Attribute Mapping on page 522](#)
- [Modifying Data from NULL to No Match Row Id on page 523](#)

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” on page 520](#).
- 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” on page 192](#).
- 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. 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" on page 520](#).

- 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\di ccache.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

Figure 60 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. For more information about EIM objects that the wizard creates, see “Objects You Use with Enterprise Integration Manager” on page 617.

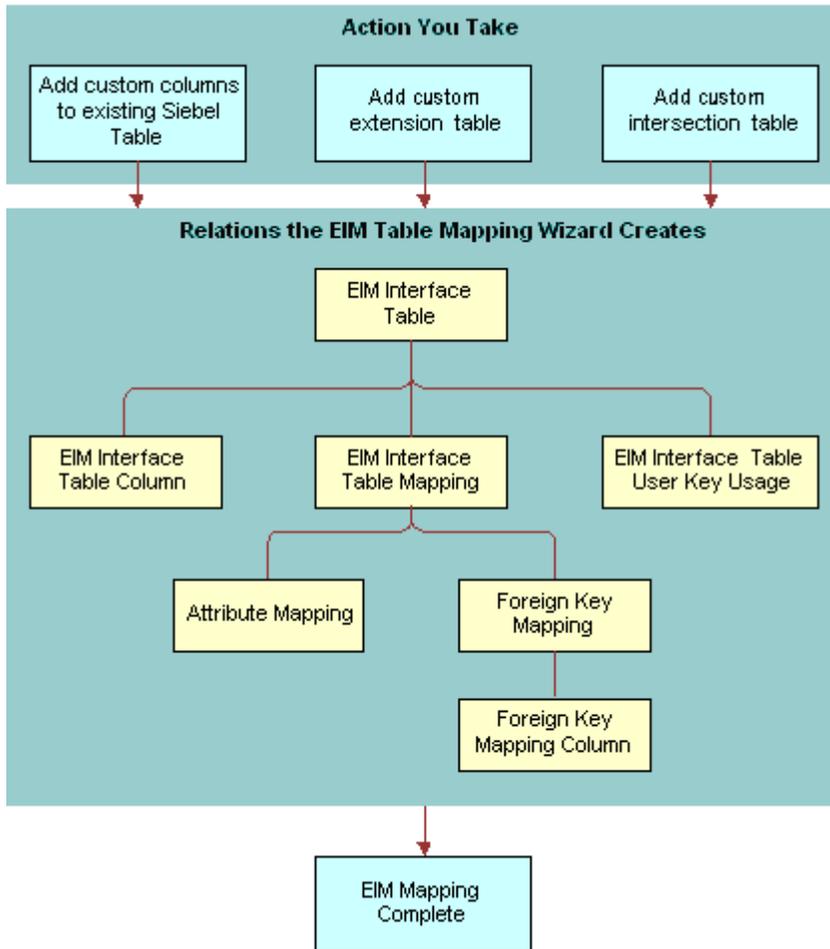


Figure 60. Relations That the EIM Table Mapping Wizard Creates

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

Table 63 describes restrictions on adding or modifying EIM mappings.

Table 63. Restrictions on Adding and 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. For more information, see Article ID 507151.1 on My Oracle Support.

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.

Property	Value
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" on page 518](#).

- 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" on page 192](#).
- 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.

24 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 on page 525](#)
- [Configuring Dock Objects on page 530](#)

About Dock Objects

This topic describes dock objects. includes the following information:

- [Dock Object Table on page 526](#)
- [Dock Object Visibility Rule on page 526](#)

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” on page 526](#).

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

Table 64 describes the values you can enter in the Type property of the dock object visibility rule when you use a limited dock object.

Table 64. Type Property of the Dock Object Visibility Rule

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 “Check Dock Object Visibility Rule” on page 527 .
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” on page 192.](#)
- 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 [Step 1](#) 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 Repeat [Step 2](#) 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 on page 531](#)
- [Creating a New Dock Object on page 532](#)
- [Adding a Dock Object Table to an Existing Dock Object on page 535](#)
- [Verifying That Siebel Tools Created Dock Objects on page 536](#)
- [Rebuilding the Databases After You Run the Docking Wizard on page 537](#)
- [Cleansing Dock Objects on page 537](#)
- [Creating a Table for a Dock Object on page 538](#)

For more information, see the following topics:

- [Setting Up a Developer with a Local Database on page 193](#)
- [Guidelines for Creating a Custom Docking Rule on page 67](#)
- [Determining Technical Fit for Reusing a Predefined Object on page 214](#)
- [Downloading a Data Layer Customization to Remote Users on page 243](#)

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.

Table 65 lists some of the business components and their associated dock objects.

Table 65. Some of the Business Components and Their 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
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” on page 192](#).
- 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 in [Step 4](#).
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 [Step a](#).
 - 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” on page 535](#).

You can use the Docking Wizard to create public, private, and limited dock objects.

For more information, see [“Guidelines for Using the Docking Wizard” on page 534](#) and [“How the Docking Wizard Behaves Depending on Where You Start It” on page 534](#).

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” on page 531](#).
- 2 Make sure related projects are locked.
For more information, see [“Locking Related Projects” on page 535](#).
- 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” on page 535](#).
- 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” on page 536](#).
- 9 Rebuild the databases.
For more information, see [“Rebuilding the Databases After You Run the Docking Wizard” on page 537](#).

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” on page 23](#).
- 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” on page 23](#).

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” on page 526](#).

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 on page 532](#) through [Step 4 on page 533](#).
- 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” on page 536](#).
- 8 Rebuild the databases.
For more information, see [“Rebuilding the Databases After You Run the Docking Wizard” on page 537](#).

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” on page 192](#).
- 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” on page 526](#) 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\di\ccache.dat`
 - `ORACLE_HOME\bin\di\cdata.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 IdDbRecreate 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 [Step b](#).

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” on page 535](#).
- 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 Business Applications

This chapter describes how to configure your Siebel application so that you can deploy it in a localized environment. It includes the following topics:

- [Overview of Localizing a Siebel Application on page 539](#)
- [Localizing a Multilingual List of Values on page 542](#)
- [Converting Your Current Data for an MLOV on page 555](#)
- [Configuring Certain Siebel Modules to Use MLOV Fields on page 561](#)

For more information about localizing Siebel Business Applications, 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 on page 540](#)
- [Deleting a Control or List Column While in Language Override Mode on page 541](#)
- [Localizing an Application Menu on page 541](#)
- [Localizing Help on page 541](#)

For more information, see the following topics:

- [Creating and Deploying a Browser Script on page 200](#)
- [Troubleshooting a View That Siebel CRM Does Not Display in the Siebel Client on page 279](#)
- [Setting the Language Mode of the Applet Layout Editor on page 303](#)
- [Setting the Input Method Editor Mode on a Control or List Column on page 312](#)
- [Using the Conversion Wizard to Convert a Form Applet to Grid Layout on page 316](#)
- [Guidelines for Modifying a Predefined Query on page 180](#)
- [Properties of a Command on page 608](#)

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 compile a new SRF file. You only need to compile 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” on page 555](#).

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 Siebel 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 Layout Editor without saving your modifications. For more information, see [“Deleting a Control or List Column” on page 307](#).

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 swe tag.

Localizing an application menu

- Increase the width parameter in the swe:menu tag in the CCFramebanner.swt 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.swt web template, see [“Configuring Web Templates to Display Menus, Toolbars, and Thread Bars” on page 477](#).

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” on page 542](#).
- 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 your 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 Web Server Extension:

```
ORACLE_HOME\publi c\install \ language\help
```

where:

- *ORACLE_HOME* is the directory where you installed the Siebel Web Server Extension.
- *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 on page 543](#)
- [Configuring a Multilingual List of Values on page 544](#)
- [Defining Properties of an MLOV on page 548](#)
- [Adding Records for All Supported Languages on page 549](#)
- [Searching a Multilingual List of Values on page 550](#)
- [Searching Fields That an MLOV Controls on page 551](#)
- [Deactivating an MLOV Record Instead of Deleting It on page 552](#)
- [Guidelines for Localizing a Multilingual List of Values on page 552](#)

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 on page 389](#)
- [Creating a List of Values on page 416](#)
- [Guidelines for Modifying a Predefined Query on page 180](#)

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

[Table 66](#) 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.

Table 66. Example of How Language-Independent Code Works

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” on page 561](#).

To configure a multilingual list of values

- 1** Consider potential performance issues.
For more information, see [“Considering Factors That Affect MLOV Performance” on page 509](#).
- 2** In Siebel Tools, display the following object types:
 - Dock Object
 - Dock Object Visibility RuleFor more information, see [“Displaying Object Types You Use to Configure Siebel CRM” on page 192](#).
- 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” on page 417](#).
 - b** Make sure the Translate property contains a check mark.
For more information, see [“Modifying the Translate Property” on page 549](#).
 - 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” on page 546](#).
 - 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”](#) on page 547.
- 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 Compile your modifications.
- 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 Locate the list of values you must modify.
 - c Create a new record for each display value for the type of list of values that you use for a language.
For more information, see [“Adding Records for All Supported Languages”](#) on page 549.
 - d Repeat [Step c](#) 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”](#) on page 555.
- 12 Test your modifications.

Example of Determining Whether the List Is Bounded

Table 67 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” on page 560.

Table 67. Example of Inconsistently Bounded Columns

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 right 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” on page 417](#).
- 2 Add translated display values for the languages that Siebel CRM must display.
For more information, see [“Adding Records for All Supported Languages” on page 549](#).
- 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

Table 68 lists columns that you cannot use with an MLOV.

Table 68. 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
S_SRC	SRC_CD	SOURCE_TYPE	Yes

Table 68. Columns That You Cannot Use with an MLOV

Table	Column	LOV Type	Bounded?
S_SRC	STATUS_CD	CAMPAIGN_STATE	No
S_SRC_EVT	FORMAT_CD	EVENT_FORMAT	Yes
S_SRCH_PROP	NAME	No LOV type	No

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” on page 417](#).
- 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” on page 555 . 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 for a Predefined List of Values” on page 549 .

Property	Description
Language-Independent Code	<p>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-independent code must be 30 characters or less. It is typically the English-American version. You cannot modify the language-independent code.</p> <p>If you click the List of Values Explorer link in the Siebel client, then the Code field displays the language-independent code.</p>
Display Value	<p>Contains the text that Siebel CRM displays in a list. It stores it in the Siebel database for an MLOV that is not enabled.</p> <p>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.</p>

Modifying the Translate Property

A *translatable list of values* is 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” on page 565](#).

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” on page 548](#).

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” on page 553](#). For more information, see [“Guidelines for Modifying a Predefined Query” on page 180](#).

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” on page 118](#).

- 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” on page 554](#).
- 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” on page 23](#).

- 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” on page 555](#).

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” on page 118](#).
- 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” on page 118](#).

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 on page 556](#)
- [Parameters You Use to Run the MLOV Converter Utility on page 557](#)
- [Resuming the MLOV Converter Utility If an Error Occurs on page 558](#)
- [Using the MLOV Converter Utility to Convert Multiple Languages on page 559](#)
- [Troubleshooting Problems with an MLOV Conversion on page 560](#)

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 for Microsoft Windows*. If you use UNIX, then see *Siebel Installation Guide for UNIX*.
- 3 Define the required parameters.
For more information, see [“Parameters You Use to Run the MLOV Converter Utility” on page 557](#).

- 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 `ml ovupgd_veri fy. l og`. The default location of the file is the `si ebsrvr\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” on page 558](#).
- 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 Compile and test your modifications.

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

Table 69 describes the parameters you use to run the MLOV Converter Utility.

Table 69. 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 Name Server 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: ODBC Data Source Name	ODBC 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

Table 69. Parameters You Use to Run the MLOV Converter Utility

Name of Dialog Box	Values You Must Choose
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:

`si ebug /m master_ml ov_veri fy. ucf`

- To resume running in translation mode, type the following command:

`si ebug /m master_ml ov_transl ate. 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:

`srvrupgwi z /m master_ml ov_veri fy. ucf`

- To resume running in translate mode, type the following command:

`srvrupgwi z /m master_ml ov_transl ate. 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” on page 555](#).

- 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:

```
si ebsrvr\log\mlov_verify_validation\output or si ebsrvr\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 on page 544](#).
- 3 Make sure the Translation Table property is set properly.
- 4 For more information, see [Step 5 on page 544](#).
- 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” on page 552](#).
- 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” on page 549](#).

- b** Compile your modifications.

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 on page 561](#)
- [Configuring Siebel Assignment Manager to Use MLOV Fields on page 565](#)
- [Configuring Siebel Anywhere to Use MLOV Fields on page 566](#)

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” on page 192](#).

- 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” on page 562](#).
 - b [“Creating a List That Uses Language-Independent Code” on page 563](#).

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”](#) on page 563.

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”](#) on page 564.

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 on page 563 .
PickList	Choose the picklist you created in Step 3 on page 563 .
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 Compile your modifications.
- 4 Administer the values:
 - a Open the Siebel client, navigate to the Administration - Business Process screen, and then click the Workflow Policies link.
Make sure the Siebel client is connected to the Siebel repository file you just compiled.
 - 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” on page 561](#).

For more information, see [“Siebel Assignment Manager” on page 38](#) 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” on page 562](#) with the following modifications:
 - a In [Step 1 on page 562](#), display the Workflow Policy Column and Assignment Attribute object types.
 - b In [Step 6 on page 562](#), 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 on page 565](#):
 - 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, "[Preparing Policy Conditions and Action Arguments for an MLOV](#)" on page 562.

- 4 Compile and test your modifications.
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 Compile and test your modifications.
For more information, see *Using Siebel Tools*.

26 Configuring the Customer Dashboard

This chapter describes how to configure the Customer Dashboard. It includes the following topics:

- [Overview of the Customer Dashboard on page 567](#)
- [Enabling the Customer Dashboard on page 568](#)
- [Process of Configuring the Customer Dashboard on page 569](#)
- [Modifying the Appearance and Layout of the Customer Dashboard on page 574](#)
- [Options to Update the Customer Dashboard on page 582](#)

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” on page 582](#).
- 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” on page 583](#).
- 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 file, CCApplletDashboard.swt. See also [“Mapping a Business Component Field to a Customer Dashboard Field” on page 571](#).

Object Types That the Customer Dashboard Uses

Table 70 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.

Table 70. Object Types That the Customer Dashboard Uses

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” on page 192.](#)
- 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” on page 569.](#)
- 2 [“Mapping a Business Component Field to a Customer Dashboard Field” on page 571.](#)
- 3 Also perform any necessary tasks described in [“Modifying the Appearance and Layout of the Customer Dashboard” on page 574](#), such as [“Creating a Label for a Customer Dashboard Field” on page 574.](#)

Adding a Business Component to the Customer Dashboard

This task is a step in [“Process of Configuring the Customer Dashboard” on page 569.](#)

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” on page 192.](#)

- 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" on page 571](#).

Property	Description
Name	<p>Enter a name for this business service user property that represents the business component list. Use the following format:</p> <p style="text-align: center;"><i>Li st integer</i></p> <p>where <i>integer</i> is a number. For example:</p> <p style="text-align: center;"><i>Li st 1</i></p>
Value	<p>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:</p> <p style="text-align: center;"><i>business component name; business component field name; business component field name</i></p> <p>where:</p> <ul style="list-style-type: none"> ■ <i>business component name</i> 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. ■ <i>business component field name</i> 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: <p style="text-align: center;">Contact; Last Name; First Name; Full Name</p>

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.

Table 71 describes the predefined List 1 business component list for contacts and List 2 for opportunities.

Table 71. Example Business Component Lists in the Business Service User Property of the Persistent Customer Dashboard Business Service

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" on page 569](#).

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	<p>Enter a name for this business service user property that represents the customer dashboard field. Use the following format:</p> <p style="text-align: center;"><i>Field integer</i></p> <p>where:</p> <ul style="list-style-type: none"> ■ <i>integer</i> is a number. <p>For example:</p> <p style="text-align: center;"><i>Field 1</i></p>
Value	<p>You must use the following format:</p> <p style="text-align: center;"><i>name of the business component list.position</i></p> <p>where:</p> <ul style="list-style-type: none"> ■ <i>position</i> is the position of the business component field in the business component list. <p>Consider the following examples from Table 71 on page 571:</p> <ul style="list-style-type: none"> ■ 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: <p style="text-align: center;"><i>List 1.1</i></p> ■ 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: <p style="text-align: center;"><i>List 1.2</i></p>

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	<p>Enter a name for this business service user property that represents the customer dashboard field. For example:</p> <p style="text-align: center;">Field 1</p>
Value	<p>Reference each field from each business component list. You must use the following format:</p> <p style="text-align: center;"><i>name of the business component list.position; name of the business component list.position</i></p> <p>where:</p> <ul style="list-style-type: none"> ■ <i>position</i> is the position of the business component field in the business component list. <p>For example, using the business component lists described in Table 71 on page 571, assume you must display the following information:</p> <ul style="list-style-type: none"> ■ If the Customer Dashboard is in the Contacts context, then the Field 1 customer dashboard field must display the Last Name of the contact. ■ If the Customer Dashboard is in the Opportunities context, then the Field 1 customer dashboard field must display the Opportunity Name. <p>You use the following code to implement this example:</p> <p style="text-align: center;">List 1.1; List 2.1</p> <p>List 1.1 represents the first field of List 1. List 2.1 represents the first field of List 2.</p>

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” on page 574](#)
- [“Formatting a Customer Dashboard Phone Number Field” on page 575](#)
- [“Modifying the Go To List in the Customer Dashboard” on page 577](#)
- [“Modifying the Background Color and Border of the Customer Dashboard” on page 580](#)
- [“Adding a Custom Control to the Customer Dashboard” on page 580](#)
- [“Modifying a Custom Control in the Customer Dashboard” on page 581](#)

Creating a Label for a Customer Dashboard Field

This topic is part of [“Modifying the Appearance and Layout of the Customer Dashboard” on page 574](#).

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 Repository File 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	<p>Enter a name. Use the following format:</p> <p style="text-align: center;">Label <i>business component name.business component field name</i></p> <p>For example, to reference the SR Number field of the Service Request business component, you enter the following:</p> <p style="text-align: center;">Label ServiceRequest.SR Number</p>
Caption	<p>Define the label text that Siebel CRM must display in the Customer Dashboard.</p> <p>For example, to display the text SR Number to the left of the customer dashboard field, you enter the following:</p> <p style="text-align: center;">SR Number</p>

- 5 Repeat [Step 4](#) for each label that you must display in the Customer Dashboard.
- 6 Compile, and then test your modifications.

Formatting a Customer Dashboard Phone Number Field

This topic is part of [“Modifying the Appearance and Layout of the Customer Dashboard”](#) on page 574.

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” on page 192.](#)
- 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	<p>Enter a name for this business service user property that represents the phone number prefix. You must use the following format:</p> <p style="text-align: center;">Phone Number Prefi x <i>integer or letter</i></p> <p>For example:</p> <p style="text-align: center;">Phone Number Prefi x 1</p>
Value	<p>Define the parameters for the phone number prefix. You must use the following format:</p> <p style="text-align: center;"><i>extension di gi ts; remove di gi ts; pref i x</i></p> <p>where:</p> <ul style="list-style-type: none">■ <i>extension digits</i> is number of digits in the extension.■ <i>remove digits</i> is the number of digits to remove from the front of the extension.■ <i>prefix</i> is the prefix to append to the beginning of the number. <p>For example:</p> <p style="text-align: center;">5; 1; 650555</p> <p>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:</p> <ul style="list-style-type: none">■ Specifies that the extension is 5 digits in length.■ 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. <p>The resulting phone number is 650-555-4565.</p>

- 6 Compile, and then test your modifications.

Modifying the Go To List in the Customer Dashboard

This topic is part of [“Modifying the Appearance and Layout of the Customer Dashboard”](#) on page 574.

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.

[Table 72](#) 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.

Table 72. Example Predefined Business Service User Properties That Represent Views in the Go To List

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”](#) on page 192.
- 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.

- 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	<p>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:</p> <p><i>View integer</i></p> <p>For example:</p> <p><i>View 1</i></p>
Value	<p>Create a view that Siebel CRM displays in the Go To list. You must use the following format:</p> <p><i>business component name; view name; name of the primary applet on the view; name of the foreign key field</i></p> <p>For example:</p> <p>Contact; All Activity List View; Activity List Applet With Navigation; Contact Id</p> <p>The name for each of these items must match exactly the name that is defined in the Siebel Repository File. The foreign key field is conditional. For more information, see "Referencing a Foreign Key Field from the Go To List" on page 578.</p>

- Optional. Configure a label for the view.
For more information, see ["Configuring the Label for the View in the Go To List" on page 579](#).
- To modify a view or remove a view from the Go To list, do the following:
 - Locate the view in the Business Service User Props list.
 - Modify or delete the record, as required.
- Compile and test your modifications.
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	<p>Enter a name. Use the following format:</p> <p style="text-align: center;">Label <i>name of the user property</i></p> <p>where:</p> <ul style="list-style-type: none"> ■ <i>name of the user property</i> is the name of the business service user property that defines the view that Siebel CRM displays in the Go To list. <p>For example, to reference the business service user property that references the All Activity List View in Step 5 on page 578, you enter the following:</p> <p style="text-align: center;">Label Vi ew 1</p>
Caption	<p>Define the label text that Siebel CRM must display as the list item in the Go To list.</p> <p>For example, to display the text Activities for This Contact, enter the following:</p> <p style="text-align: center;">Acti vi ti es for Thi s Contact</p>

- 5 Repeat [Step 4 on page 579](#) for each label that you must display in the Go To list.
- 6 Compile, and then test your modifications.

Modifying the Background Color and Border of the Customer Dashboard

This topic is part of [“Modifying the Appearance and Layout of the Customer Dashboard”](#) on page 574.

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 Web Server Extension installation:

```
PUBLIC\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 Defi ni ti ons*/  
/*-----*/  
. siebui -dashboard-frame {background: #999999; }  
. siebui -dashboard-frame {border: 2px sol id #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”](#) on page 574.

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 Save your changes, and then compile your changes into an SRF file.
 - 7 On the Siebel Server computer, copy the file `SI EBSRVR_ROOT\webtempl \CCAppletDashboard. swt` to `SI EBSRVR_ROOT\webtempl \custom`.
 - 8 Edit the `CCAppletDashboard. swt` file to add the new SWE control, as follows:

For example, to add a new Button control, add a control tag similar to the following:

```
<swe: control id="2902">  
<div><swe: this property="FormattedHtml " hintText="Button" /></div>  
</swe: control >
```

For example, to add a new Field control, add a control tag similar to the following:

```
<swe: control id="1300">  
<swe: this property="FormattedHtml " hintText="Field"  
hintMapLabelId="swe: currentId+100" />  
</swe: control >
```

For example, to add a new Label control, add a control tag similar to the following:

```
<swe: control id="1702">  
<div><swe: this property="FormattedHtml " hintText="Label " /></div>  
</swe: control >
```

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.

- 9 Save the `CCAppletDashboard.swt` file.
- 10 Restart the Siebel Server to use the updated SRF file from [Step 6 on page 581](#).

Modifying a Custom Control in the Customer Dashboard

This topic is part of ["Modifying the Appearance and Layout of the Customer Dashboard" on page 574](#).

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" on page 580](#).

To modify a custom control in the Customer Dashboard

- 1 On the Siebel Server computer, open the file `SI EBSRVR_ROOT\webtempl \custom\CCAppletDashboard.swt` for editing.
You previously copied this file to the custom directory, in [“Overview of the Customer Dashboard” on page 567](#).
- 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” on page 567](#).
- 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 on page 582](#)).
- 7 Modify the Web template item as desired.
- 8 Save your changes, and then compile your changes into an SRF file.
- 9 Restart the Siebel Server to use the updated SRF file.

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” on page 583](#)
- [“Configuring Communications Events to Update the Customer Dashboard” on page 584](#)
- [“Configuring SmartScript to Update the Customer Dashboard” on page 586](#)
- [“Using Siebel Visual Basic or Siebel eScript to Update Information in the Customer Dashboard” on page 588](#)
- [“Using Personalization to Update the Customer Dashboard” on page 591](#)

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” on page 584.](#)
- **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” on page 586.](#)
- **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” on page 588.](#)
- **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” on page 591.](#)
- **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” on page 582.](#)

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” on page 588.](#)

- 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 Compile and test your modifications.

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” on page 582](#).

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*.

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 [Table 73 on page 585](#).

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 Table 73 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.

Table 73. Example Parameters for Communication Events to Update Customer Dashboard

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

Name	Value
ServiceParam.Value	{Contact.Id}
WorkTrackingObj.ContactId	{Contact.Id}

Configuring SmartScript to Update the Customer Dashboard

This topic is part of [“Options to Update the Customer Dashboard” on page 582](#).

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 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” on page 192](#).
- 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
	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" on page 571](#).

- 4 Compile your modifications, and then open the Siebel client.
- 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” on page 582](#).

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” on page 582](#).

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*.

A

Reference Materials for Configuring Siebel Business Applications

This appendix provides reference information for Siebel CRM. It includes the following topics:

- [Properties of Object Types on page 593](#)
- [Types of Applet Controls and List Columns on page 613](#)
- [Objects You Use with Enterprise Integration Manager on page 617](#)
- [Types of Tables and Columns That CIAI Query Supports on page 624](#)
- [Extensive Code Examples That This Book Uses on page 626](#)

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 on page 594](#)
- [Properties of a Table Column on page 594](#)
- [Properties of an Index of a Siebel Table on page 598](#)
- [Properties of an Index Column on page 599](#)
- [Properties of a Business Component on page 599](#)
- [Type Property of a Business Component Field on page 600](#)
- [Display Format Property of a Control or List Column on page 604](#)
- [Properties of a Screen View on page 606](#)
- [Properties of an Application on page 607](#)
- [Properties of Objects You Use with a Menu or Toolbar on page 608](#)

For a complete list of properties of objects, see *Siebel Object Types Reference*.

Properties of a Siebel Table

Table 74 describes properties that Siebel CRM commonly uses with a Siebel table.

Table 74. Properties That Siebel CRM Commonly Uses with a Siebel Table

Property	Description
Name	The name of the table in the RDBMS.
Type	The table type. For more information, see Table 77 on page 597 .
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

Table 75 describes properties that Siebel CRM commonly uses with a table column.

Table 75. 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" on page 518 .
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.

Table 75. Properties That Siebel CRM Commonly Uses with a Table Column

Property	Description
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” on page 595 .
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.
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” on page 597 .
User Key Sequence	Stores the sequence in the user key where this column fits.

Physical Type Property of a Table Column

[Table 76](#) describes the physical types that Siebel CRM supports. The Physical Type property identifies the physical type of the column in the Siebel database.

Table 76. Physical Types That Siebel CRM Supports in the Physical Type Property of a Table Column

Physical Type	Description
Character	<p>Stores text that is fixed in length. Also used for a <i>Boolean column</i>, which is a character column that contains a length of 1. You cannot use Char greater than 1, by default.</p> <p>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:</p> <p style="padding-left: 40px;">Allow to create column of type ‘Character’ being greater than 1</p> <p>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).</p>
Character Large Object (CLOB)	For more information, see “Character Large Object (CLOB) Physical Type” on page 596 .
Date	Stores the date only, without time.
Date Time	Stores combined date and time in the same column.

Table 76. Physical Types That Siebel CRM Supports in the Physical Type Property of a Table Column

Physical Type	Description
Long	Stores long text. You can store approximately 16 KB of data in a Long column.
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 “Maximum Number of Digits for a Numeric Physical Type” on page 597 .
Time	Stores time only, without the date.
UTC Date Time	Stores the date and time. Siebel CRM saves time in Greenwich Mean Time (GMT).
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. Siebel CRM version 8.0 and higher supports this text. CLOB is similar to Long, 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:

- A column in a Siebel table is limited to 128 KB of data. You cannot define a column of type CLOB that is greater than 128 KB.
- 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 define a CLOB type. It treats a CLOB as a varchar(max) or nvarchar(max) object.
- 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 to the left of the decimal point, the number of usable digits to the right of 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 to the left of the decimal point.
- You cannot use more than 7 digits to the right of 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” on page 55](#).
- **Extension.** For more information, see [“Extension Columns of a Siebel Table” on page 55](#).
- **System.** For more information, see [“System Columns of a Siebel Table” on page 56](#).
- **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.

[Table 77](#) describes possible values for the Type property of a Siebel table.

Table 77. 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.

Table 77. Values for the Type Property of a Siebel Table

Type	Description
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

Table 78 describes some of the properties that Siebel CRM commonly uses with an index.

Table 78. Properties That Siebel CRM Commonly Uses with an Index

Property	Description
Name	Stores the name of the database index.
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” on page 23 .
Type	Indicates the type of index. Siebel CRM commonly uses the following types: <ul style="list-style-type: none"> ■ 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

Table 79 describes some of the properties that Siebel CRM commonly uses with an index column.

Table 79. 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: <ul style="list-style-type: none"> ■ Asc (ascending) ■ Desc (descending)

Properties of a Business Component

Table 80 describes some of the properties that Siebel CRM commonly uses with a business component.

Table 80. 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” on page 74 .
Name	A name that must be unique among all business components that reside in the SRF. 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” on page 118 .
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” on page 246 .
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

Table 81 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_.

Table 81. Type Property of a Business Component Field

Field Data Type	Physical Type	Maximum Length	Description
DTYPE_BOOL	Character	1	<p>Describes data as Y or N. Siebel Tools often displays this type in the following ways:</p> <ul style="list-style-type: none"> ■ TRUE or FALSE ■ checked or unchecked
DTYPE_CURRENCY	Number	22	<p>Describes data as currency.</p> <p>You can use the Windows Control Panel to control the appearance of currency values that Siebel CRM displays in a screen.</p> <p>To define an explicit format mask in the Display Format property, you can use the following symbols:</p> <ul style="list-style-type: none"> ■ 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.
DTYPE_DATE	Date	7	<p>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” on page 602.</p>

Table 81. Type Property of a Business Component Field

Field Data Type	Physical Type	Maximum Length	Description
DTYPE_DATETIME	Date Time	7	<p>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.</p> <p>Alternatively, you can use one of the following properties. Each of these properties use the format configured in the Windows Control Panel:</p> <ul style="list-style-type: none"> ■ 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	<p>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.</p> <p>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.</p>
DTYPE_ID	Varchar	15	<p>Describes data as the primary key that Siebel CRM creates.</p> <p>A field mapped to an extension column with a physical type of Varchar(15) defaults to a DTYPE_ID data type.</p>
DTYPE_INTEGER	Number	22*	<p>Describes data as a whole number ranging in value from negative 2147483648 to 2147483647.</p>
DTYPE_NOTE	Long	16 KB	<p>Describes data as a long string that is less than or equal to 16 KB, or 16383 bytes. If the length is not explicitly defined, then the default is 16 KB. If used with the Pop-up Edit property in a control or list column, then the DTYPE_NOTE data type indicates to the Siebel client to use a multiline edit box.</p> <p>The user cannot query on a DTYPE_NOTE field.</p>

Table 81. Type Property of a Business Component Field

Field Data Type	Physical Type	Maximum Length	Description
DTYPE_NUMBER	Number	22	Describes data as a number. For more information, see “Formats for a DTYPE_NUMBER Business Component Field” on page 603.
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	<p>Describes data as a time.</p> <p>When Siebel CRM gets the time, it ignores other information. You can set the appearance of time values through the Windows Control Panel, or you can use the following symbols to specify an explicit time:</p> <ul style="list-style-type: none"> ■ 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”](#) on page 605.

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 [“Maximum Number of Digits for a Numeric Physical Type”](#) on page 597.

Display Format Property of a Control or List Column

Table 80 describes values for the Display Format property of a control or list column.

Table 82. Values for the Display Format Property of a Control or List Column

Property	Description
DTYPE_NUMBER	<p>Can include the following values:</p> <ul style="list-style-type: none"> ■ 0 (zero) ■ # (pound sign) ■ + (plus sign) ■ - (minus sign) ■ , (comma) ■ . (period) <p>If empty, then to determine the appearance of numeric values, Siebel CRM uses the Regional Settings section of the Windows Control Panel.</p>
DTYPE_CURRENCY	<p>Uses the same symbols as with the DTYPE_NUMBER property, in addition to the dollar sign.</p> <p>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.</p>
DTYPE_DATETIME	<p>Stores one of the following values in the Display Format property:</p> <ul style="list-style-type: none"> ■ Date ■ DateTime ■ DateTimeNoSec ■ TimeNoSec
DTYPE_DATE	<p>Uses a combination of the following values:</p> <ul style="list-style-type: none"> ■ MM/DD/YY <p>where:</p> <ul style="list-style-type: none"> ■ <i>MM</i> is the month. ■ <i>DD</i> is the day. ■ <i>YY</i> is the year. <p>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" on page 605.</p>

Table 82. Values for the Display Format Property of a Control or List Column

Property	Description
DTYPE_TIME	<p>You can use one of the following formats:</p> <ul style="list-style-type: none"> ■ Specify TimeNoSec. ■ Specify a format mask using a combination of the following values: <ul style="list-style-type: none"> ■ HH:hh:mm:ss where: <ul style="list-style-type: none"> ■ <i>HH</i> is the hours. ■ <i>hh</i> is the hours. ■ <i>mm</i> is the minutes. ■ <i>ss</i> is the seconds. <p>If empty, then to determine the appearance of time values, Siebel CRM uses the Regional Settings section of the Windows Control Panel.</p>
DTYPE_PHONE	<p>If empty, then to determine the appearance of phone values, Siebel CRM uses the Regional Settings section of the Windows Control Panel.</p>

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.

Properties of a Screen View

Table 83 describes some of the properties that Siebel CRM commonly uses with a screen view.

Table 83. Properties That Siebel CRM Commonly Uses with a Screen View

Property	Description
Category Default View	<p>Defines the default view for a screen view as an Aggregate Category or Detail Category.</p> <p>It is recommended that you define the Category Default View property. If this property is empty, then Siebel CRM lists views alphabetically.</p> <p>If the category for the screen view is Aggregate View or Detail View, then the Category Default View property is read-only.</p>
Category Name	<p>Name of the Aggregate Category or Detail Category. This property must be unique in each screen.</p> <p>If the category for the screen view is Aggregate View or Detail View, then the Category Name property is read-only.</p>
Display in Page	<p>If Siebel CRM must display the screen view in the page, then you must set the Display in Page property to TRUE.</p> <p>If the category for the screen view is Aggregate Category or Detail Category, then you must set the Display in Page property to TRUE.</p> <p>If no views in the category are included, then Siebel CRM does not display the screen view.</p>
Display in Site Map	<p>If Siebel CRM must display the screen view on the Site Map, then the Display in Site Map property must be TRUE.</p>
Menu Text	<p>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” on page 338.</p>
Name	<p>Calculated field.</p>
Parent Category	<p>If the Type property is Aggregate Category, then the Parent Category property is empty because no parent exists.</p> <p>If the Type property is Aggregate View, then the Parent Category property might or might not be empty.</p> <p>If the Type property is Detail View or Detail Category, then the Parent Category property must contain a value.</p>

Table 83. Properties That Siebel CRM Commonly Uses with a Screen View

Property	Description
Type	<p>Can be one of the following values:</p> <ul style="list-style-type: none"> ■ Aggregate Category ■ Aggregate View ■ Detail Category ■ Detail View <p>For more information, see “About Screen Views” on page 133.</p>
View	<p>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.</p>

Properties of an Application

Table 84 describes properties of the Application object type.

Table 84. Properties of the Application Object Type

Property	Description
Acknowledgement Web Page	<p>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.</p>
Acknowledgement Web View	<p>The view that Siebel CRM displays after the user successfully logs in, except in the following situations:</p> <ul style="list-style-type: none"> ■ 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: <ul style="list-style-type: none"> ■ An anonymous user logs in to a customer application. ■ The Explicit Login property is set to TRUE for the view that the user attempts to access. ■ The user successfully enters the login credentials. <p>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.</p>

Table 84. Properties of the Application Object Type

Property	Description
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” on page 149.
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

[Table 85](#) describes some of the properties that Siebel CRM commonly uses with a command.

Table 85. 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: <ul style="list-style-type: none"> ■ If the Business Service property is empty, then Siebel CRM targets the browser or server infrastructure rather than a specific service. ■ 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.

Table 85. Properties That Siebel CRM Commonly Uses with a Command

Property	Description
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” on page 451.
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: <ul style="list-style-type: none"> ■ 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: <ul style="list-style-type: none"> ■ 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. <p>The object manager infrastructure is the Siebel Web Engine UDF loader or the model.</p> <ul style="list-style-type: none"> ■ Browser Applet. <p>For more information, see “About the Method, Business Service, and Target Properties of the Command Object” on page 451.</p>
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 Chapter 25, “Localizing Siebel Business Applications.”

Properties of a Toolbar

Table 86 describes some of the properties that Siebel CRM commonly uses with a toolbar.

Table 86. Properties That Siebel CRM Commonly Uses with a Toolbar

Property	Description
Class	Note the following: <ul style="list-style-type: none"> ■ 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 swe:toolbar tag in the name clause.

Properties of a Menu Item

Table 87 describes some of the properties that Siebel CRM commonly uses with a menu item.

Table 87. 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: <ul style="list-style-type: none"> ■ 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

Table 88 describes some of the properties that Siebel CRM commonly uses with a toolbar item.

Table 88. 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: <ul style="list-style-type: none"> ■ ComboBox ■ Button ■ Edit ■ Label ■ 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: <ul style="list-style-type: none"> ■ 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

Table 89 describes some of the properties that Siebel CRM commonly uses with an applet method menu item.

Table 89. 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

Table 90 describes some of the properties that Siebel CRM commonly uses with a class method menu item.

Table 90. Properties That Siebel CRM Commonly Uses with a Class Method Menu Item

Property	Description
Business Service	Note the following: <ul style="list-style-type: none"> ■ 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.
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.

Table 90. Properties That Siebel CRM Commonly Uses with a Class Method Menu Item

Property	Description
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	<p>Specifies the entity that handles the method that is defined in the Method property. The following options are available:</p> <ul style="list-style-type: none"> ■ 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*.

Table 91 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.

Table 91. Applet Controls and List Columns That Siebel CRM Commonly Uses

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” on page 114 and “Calling a Method from a Button in an Applet” on page 336 .
CheckBox	Represents a TRUE or FALSE situation: <ul style="list-style-type: none"> ■ 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. <p>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.</p>
ComboBox	Defines one of the following special-purpose lists: <ul style="list-style-type: none"> ■ In a chart applet, it defines the Show and By combo boxes. ■ In a calendar applet, it defines the user name combo box. ■ In a pick applet, it defines the Find combo box. <p>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.</p> <p>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” on page 389.</p>
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” on page 115 .
FieldLabel	Displays a field label for a list applet.
File	Allows the user to attach a file.

Table 91. Applet Controls and List Columns That Siebel CRM Commonly Uses

HTML Type Property	Description
Hidden	Not visible in the Web page but you can access it through a script.
FormSection	<p>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.</p> <p>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.</p> <p>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.</p>
ImageButton	A minibutton that references an image. For more information, see “MiniButton Control and MiniButton List Column” on page 114.
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” on page 74.
Label	<p>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.</p> <p>If a caption includes any HTML reserved character, then you must encode the HTML with &amp; ; , &lt; ; , &gt; ; , &quot; ; , and so on. Example HTML reserved characters include the ampersand (&), less than symbol (<), greater than symbol (>), period (.), and so on.</p>
Link	<p>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.</p> <p>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.</p>
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.

Table 91. Applet Controls and List Columns That Siebel CRM Commonly Uses

HTML Type Property	Description
RTC	<p>The following RTC (Rich Text Component) controls define the dimensions and font qualities of a container that includes a rich text component:</p> <ul style="list-style-type: none"> ■ RTCEmbedded. An embedded text editor. Siebel CRM supports the following HTML tags: <ul style="list-style-type: none"> ■ Bold ■ italic ■ Underline <U> ■ Ordered list ■ Unordered list ■ List items ■ <P> ■ ■ <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. <p>You cannot use an RTC control in a list applet, list, or a multi-value group applet.</p>
RadioButton	Displays a radio button.
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	<p>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.</p> <p>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.</p> <p>For more information, see “Text Control and Text List Column” on page 115.</p>
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

Table 92 describes properties of the EIM interface table.

Table 92. 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

Table 93 lists system columns of the EIM interface table.

Table 93. System Columns of the EIM InterfaceTable

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

Table 94 describes the generic EIM interface table columns to facilitate EIM processing for each EIM table interface. You cannot modify the values of these columns.

Table 94. EIM Interface Table Columns to Facilitate EIM Processing

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
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

Table 95 describes EIM interface table columns for processing a mapping to a defined table.

Table 95. 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: <i>T_PART ONE_PROCESS SUFFIX</i> where: ■ <i>PART ONE</i> 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

Table 96 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.

Table 96. Example of EIM Interface Table Columns for Processing a Mapping to a Defined Table

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
T_SEC_LEV_UNQ	Character	1	IFMGR: Unique	CX_SEC_LEV	TRUE

EIM Interface Table Columns for Processing a Foreign Key

Table 97 describes the columns that EIM creates for each foreign key on the target EIM table mapping.

Table 97. EIM Interface Table Columns for Processing a Foreign Key

Column	Value
Name	Derived from the target table name and the corresponding foreign key column on the target table using the following format: <i>PART ONE FOREIGN KEY COLUMN OF THE TARGET TABLE</i> where: ■ <i>PART ONE</i> 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: <i>TARGET TABLE NAME or EIM TABLE MAPPING NAME.FOREIGN KEY COLUMN NAME</i>

Table 98 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

Table 98. Example of EIM Interface Table Columns for Processing a Foreign Key

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

Table 99 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.

Table 99. EIM Interface Table Columns for Foreign Keys

Column	Value
Name	Derived using the following format: <i>PART ONE_NAME OF THE FOREIGN KEY COLUMN IN THE TARGET TABLE</i> where: <ul style="list-style-type: none"> ■ <i>PART ONE</i> 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)

Table 100 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.

Table 100. Example of Interface Table Columns for Foreign Keys

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

Table 101 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

Table 101. EIM Interface Table Column for Attributes on the Target Table

Column	Value
Name	Derived using the following format: <i>PREFIX_NAME OF THE CORRESPONDING COLUMN IN THE TARGET TABLE</i> 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.

Table 102 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)

Table 102. Example of EIM Interface Table Columns Attributes on the Target Table

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

Table 103 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.

Table 103. Example of EIM Table Mappings That Reference the Target Table

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

Table 104 describes properties of the attribute mappings for each EIM interface column that EIM creates.

Table 104. Attribute Mapping Properties of EIM Interface Columns 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 Table 102 on page 621 .
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

Table 105 describes the properties of each foreign key mapping that EIM creates for each foreign key mapping column on the target table.

Table 105. Foreign Key Mapping Properties of Foreign Key Columns on the Target Table

Property	Value
Name	Name of the user key column.
Foreign Key Column	Name of the user key column.

Table 105. Foreign Key Mapping Properties of Foreign Key Columns on the Target Table

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: <code>T_PART ONE_NAME OF THE USER KEY COLUMN</code> where: ■ <i>PART ONE</i> is the name of the target table without the CX_ prefix.

Table 106 describes the foreign key mapping, using the CX_SEC_LEV table as an example.

Table 106. Example of Foreign Key Mapping Properties of Foreign Key Columns on the Target Table

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

Table 107 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 Table 105 on page 622.

Table 107. Foreign Key Mapping Columns for Foreign Key Mappings

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 Table 99 on page 620.
User Key Attribute	Name of the user key column that is part of the user key defined in Table 105 on page 622.

Table 108 describes the foreign key mapping, using the CX_SEC_LEV table as an example.

Table 108. Example of Foreign Key Mapping Columns for Foreign Key Mappings

Name	Interface Data Column	User Key Attribute
OPTY_BU_ID	OPTY_BU_ID	BU_ID
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

Table 109 lists the types of tables that CIAI query supports.

Table 109. 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

Table 110 lists the types of columns that CIAI query supports.

Table 110. 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

Table 111 lists the physical types of columns that CIAI query supports.

Table 111. Physical Types of Columns That CIAI Query Supports

Physical Type		Supported for a CIAI Query
DB Value	Type	
C	Char	Yes
D	Date	No
L	Long	Yes
L	CLOB	Yes
N	Number	No
S	Time Stamp	No
T	Date Time	No
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” on page 377](#). 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", "Ni eren");
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( Cancel Operation );
}
```

Index

A

Accelerator object

- about keyboard shortcuts 300
- adding keyboard shortcuts 301
- modifying keyboard shortcuts 302

action arguments, enabling 562, 565

Administration views LOV 548

advanced database customization 60

answers, configuring SmartScripts 587

Anywhere, configuring to use MLOV 566

applet controls

- about and list columns 113, 125, 127, 130, 133, 226, 382, 418, 419, 449, 483, 525, 555

button controls 614

check box controls 614

combo box controls 614

DrillDown title control 614

field control 614

FieldLabel control 614

file control 614

FormSection control 615

hidden controls 615

ImageButton control 615

JavaApplet control 615

label controls 615

Mailto control 615

MiniButtons control 114

password control 615

PositionOnRow control 615

RadioButton control 616

RecNavNxt control 616

RecNavPrv control 616

RTCEmbedded text editor 616

RTCEmbeddedLinkField 616

Show More/Less button 310

SSNxt control 616

SSPrv 616

text control 115

TextArea control 616

URL control 616

Applet Layout Editor

- about grid layout 315
- adding new controls and list columns 305
- adding predefined controls and list columns 305
- applets that cannot be converted 319

checking mappings 314

configuring controls for Show More
button 311

converting by changing Web template 318

converting form applet to grid layout 316

copying and pasting items 313

creating tree applet 370

deleting controls and list columns 307

editing layouts 303

editing list columns display names and control
captions 309

exporting applet preview 308

positioning controls and list columns 306

previewing applet layout 307

setting field tab order 312

settings 303

troubleshooting conversions to grid
layout 320

working with grid layout 321

applet layouts, editing

about grid layout 315

adding new controls and list columns 305

adding predefined controls and list
columns 305

Applet Layout Editor settings 303

applets that cannot be converted 319

checking mappings 314

configuring controls for Show More
button 311

converting by changing Web template 318

converting using Conversion Wizard 316

copying and pasting items 313

deleting controls and list columns 307

displaying column value of parent record as
title of detail applet 309

editing layouts 303

editing list columns display names and control
captions 309

exporting applet preview 308

positioning controls and list columns 306

previewing applet layout 307

setting field tab order 312

setting input method editor (IME) mode 312

troubleshooting conversions to grid
layout 320

working with grid layout 321

Applet Menu Method Wizard, using 458

applet method menu item, about 451
applet mode roles 116
applet search specifications, setting 118
applet templates, about 154
Applet Web Template Conversion Wizard
 applets that cannot be converted 319
 troubleshooting conversions to grid layout 320
 using to convert form applet to grid layout 316

applets

about detail business component 429
 about originating multi-value group business component 428
 about origination MVG applet 428
 about pick applets 393
 about the Link Object definition 104
 about toggles 141
 about toggles example 273
 association applets 433
 association applets called from parent-child views 439
 creating applet menus 458
 defining MVG applets 426
 MVG applets 423
 pick applet details 394
 reusing 209
 shuttle applets 441

applets, configuring

about 111
 about applet controls and list columns 113, 125, 127, 130, 133, 226, 382, 418, 419, 449, 483, 525, 555
 about form applets 111, 132
 about list applets 112
 about pop-up controls 328
 adding Web templates applets 345
 applet child objects 111
 applet naming format 126
 applet naming formats for titles 127
 changing styles of label text 338
 configuration guidelines 125
 control and list column configuration guidelines 127
 creating form applets 325
 creating list applets 323
 displaying a limited subset of data 310
 exposing system fields 351
 link control 615
 MVG applets 432
 pop-up applets 328
 role of applet modes 116
 setting applet search specifications 118
 setting default method 345

Show More/Less button 310
 shuttle applets 443

applications, configuring

about applications 142
 associating screens with screen menu items 282
 configuration guidelines 143
 creating applications 297
 defining page tabs 282
 defining screen menu items 282
 personalizing your Siebel application 37

architecture

about Siebel Object architecture 23
 Business Object layer 28
 data object layer 30
 logical user interface object layer 26
 physical user interface layer 25
 Siebel operating architecture 32
 summary of object types 31

Assignment Criteria and Skills, configuring for MLOVs 565

Assignment Manager

about using 38
 configuring assignment criteria and skills for MLOVs 565
 configuring to use MLOV enabled fields 565
 configuring Workload rules 566

association applets

about 433
 called from parent-child views 439

attachment applets, configuring 382

attachment business component

configuring 384
 configuring attachment tables 386

attachment list applet

configuring attachment business component 384
 configuring attachment tables 386

attachment tables, configuring 386

Attribute Mapping object type 514

B

background color, changing in customer dashboard 580, 581

base tables, assigned to business component 70

BC Read Only Field user property 86

bitmap categories and bitmap objects, creating 462

border, changing background in customer dashboard 580, 581

bounded lists, determining 546

browser group specific templates

- rendering hierarchical list applets 375
- browser scripting** 198
- browser scripts, generating** 200
- business component fields, reusing** 252
- business component properties,**
 defining 599, 608
- business components**
 - about detail business component 429
 - associating attributes to business component fields 225
 - binding entities 224
 - configuring attachment business component 384
 - copying 209
 - determining the dock object 531
 - finding dock objects 531
 - reusing 219
- business components, configuring**
 - about 69
 - about base table assigned to 70
 - about calculated fields 81
 - about creating sequence fields 82
 - about field data types 81
 - about fields 78
 - about implicit joins 44
 - about including joined table data 71
 - about joins 91
 - about reusing business components 72
 - about system fields 89
 - adding sequence fields 246
 - configuring client import 257
 - configuring dual currency 254
 - configuring read-only behavior 85
 - creating business components 245
 - defining business component properties 599, 608
 - defining Search Specification property 599
 - defining sort specifications 246
 - guidelines 76
 - intersection business components 72
 - join configuration guidelines 94
 - join construction, diagram 92
 - managing unused business components 78
 - predefault value for join field 260
 - roles of objects in joins 92
 - virtual business components 72
- business objects, configuring**
 - about business objects 105
 - configuration guidelines 109
 - how business objects are constructed 108
- business objects, copying** 209
- buttons**
 - configuring to display images 464
 - control properties 614

C

- calculated fields, about** 81
- cascade copy, construction with multi-value link** 250
- Cascade Delete property** 261
- cascading style sheets**
 - about 477
- Case Insensitivity (CIAI) Wizard**
 - about 490
 - column and index naming formats 491
 - defaults 495
 - effect of CIAI columns on sorting 490
 - eligibility criteria 491
 - error reporting 492
 - implementation of CIAI columns and indexes 490
 - input file format 497
 - method and index strategy 493
 - Operation field, controlling activation of columns and indexes 501
 - running multiple times 500
 - supported column functional types 624
 - supported column physical types 624
 - supported table types 624
- catalog-style list applets**
 - about 165
 - example 167
- chart applets**
 - performance considerations 509
- check box controls** 614
- check in/out, about** 540
- Class Method Menu Item, about** 451
- cleansing dock objects** 537
- client import, configuring for business component** 257
- color, changing background in customer dashboard** 580, 581
- columns**
 - about 55
 - about data columns 55
 - about extension columns 55, 56, 57, 598
 - about index column object type 57
 - about index columns 57, 598
 - about system columns 56
 - adding to predefined tables 63, 236
 - column properties 55
 - modifying extension columns 238
- combo box controls** 614
- Command object definition**
 - about and toolbars and menus 450
 - creating 455
- Command Object Wizard, using to choose object properties** 455

- commands, about customer dashboard**
 - commands** 588
- Comment property** 526
- communications events, configuring to enter data for customer dashboard** 584
- Competitor field, restricting** 248
- configuration goals and objectives** 185
- Container Page**
 - about 149
 - about HTML frames 150
 - areas 149
 - HTML frames in Container Page
 - templates 151
- controls**
 - configuration guidelines 127
 - configuring Show More/Less button 310
 - setting input method editor (IME) mode 312
- Conversion Wizard**
 - applets that cannot be converted 319
 - troubleshooting conversions to grid layout 320
 - using to convert form applet to grid layout 316
- copying entity relationship diagrams** 228
- currency, configuring dual currency** 254
- custom HTML control type**
 - about formats 485
 - creating 477
 - when SWE uses a custom HTML type 485
- customer applications**
 - about global deployment 541
 - language folders 542
 - localizing help 541
- customer dashboard**
 - about 567
 - about customer dashboard commands 588
 - about populating with data 567
 - activating SmartScripts Player applet 586
 - adding business components 569
 - adding user properties 570
 - architecture 568
 - changing background color and border 580, 581
 - configuring GoTo views 579
 - configuring SmartScripts 587
 - creating field labels 574
 - defining GoTo list 577
 - enabling customer dashboard 568
 - entering values using Siebel Visual Basic and Siebel eScript 583
 - formatting phone number fields 575
 - GetCurrentContactId command 588
 - GetDashboardFieldValue command 589
 - mapping business component fields 571

- mapping SmartScripts variables 586
- predefined behavior 569
- process for configuring 569
- to enter data using SmartScripts 586
- Update Dashboard command 589
- using commands with Siebel eScript 589
- using commands with Siebel Visual Basic 590
- using communication to enter data events 584

D

dashboard

- about customer dashboard 567
- about customer dashboard commands 588
- about populating with data 567
- activating SmartScripts Player applet 586
- adding business components 569
- adding user properties to the customer dashboard 570
- changing background color and border 580, 581
- configuring GoTo views 579
- configuring SmartScripts to save answers 587
- creating field labels 574
- customer dashboard architecture 568
- customer dashboard predefined behavior 569
- defining GoTo list 577
- enabling customer dashboard 568
- entering data using to SmartScripts 586
- entering values using Siebel Visual Basic and Siebel eScript 583
- formatting phone number fields 575
- GetCurrentContactId command 588
- GetDashboardFieldValue command 589
- mapping business component fields to 571
- mapping SmartScripts variables 586
- process for configuring 569
- Update Dashboard command 589
- using commands with Siebel eScript 589
- using commands with Siebel Visual Basic 590
- using communications events to enter data 584

data

- about data columns 55
- upgrading using MLOV Converter Utility 555

data model

- applying changes to server database 242

data objects layer

- customizing guidelines 62

- process for customizing 233
- database customization**
 - advanced database customization 60
 - applying to the local database 240
- database extension**
 - options 59
 - propagating changes to other local databases 243
- deleting**
 - controls and list columns 307
 - dock objects, and cleansing 537
 - extension tables or columns 239
- developers, setting up as remote users** 193
- development process overview** 36
- development work**
 - about structuring 185
 - configuration strategy 185
- dock objects, configuring**
 - about 525
 - adding new dock object table 535
 - creating new dock objects 538
 - deleting and cleansing dock objects 537
 - determining dock object belonging to a business component 531
 - Dock Object Table 526
 - Dock Object Visibility Rule 526
 - Docking Wizard, about 532
 - finding for business component 531
 - rebuilding visibility databases after running Docking Wizard 537
 - types 525
 - verifying 536
 - visibility rules 525
- Docking Wizard**
 - about 532
 - rebuilding visibility databases after running 537
- DrillDown title control** 614
- drilldowns**
 - about 138
 - dynamic drilldowns 140
 - static drilldowns 139
- dual currency, configuring** 254
- dynamic data capture, with Siebel SmartScript** 38
- dynamic list**
 - about 395
- dynamic lists**
 - about pick applets 393
 - about the originating applet 400
 - constraining dynamic lists 403
 - creating dynamic list objects 412
 - data flow 394
 - object types 397

- pick applet details 394

E

- EIM Interface Table Column object type** 512
- EIM Interface Table object type** 512
- EIM interface tables, configuring**
 - about EIM Table Mapping Wizard 518
 - about interface tables 511
 - Attribute Mapping object type 514
 - attribute mapping objects 622
 - EIM interface table column for attribute columns 621
 - EIM Interface Table Column object type 512
 - EIM interface table column specifications 617
 - EIM interface table columns for foreign key processing 619
 - EIM interface table columns for foreign keys 620
 - EIM interface table columns for table mapping 618
 - EIM interface table object specifications 617
 - EIM Interface Table object type 512
 - EIM object types 512
 - EIM Table Mapping object type 513
 - EIM table mapping objects that reference the target table 622
 - Foreign Key Mapping Column object type 517
 - foreign key mapping for foreign key column 622
 - foreign key mapping for foreign key object 623
 - Foreign Key Mapping object type 515
 - Interface Table User Key Usage object type 514
 - labeling data as NULL 523
 - User Key Attribute object type 517
 - User Key Column object type 517
 - User Key Join Attribute object type 517
 - User Key object type 517
 - using EIM Table Mapping Wizard 518
- EIM object specifications**
 - attribute mapping objects 622
 - EIM interface column 617
 - EIM interface table 617
 - EIM interface table column for attribute columns 621
 - EIM interface table columns for foreign key processing 619
 - EIM interface table columns for foreign keys 620
 - EIM interface table columns for table

- mapping 618
 - EIM table mapping objects that reference target table 622
 - foreign key mapping for foreign key column 622
 - foreign key mapping for foreign key object 623
 - EIM object types**
 - about and mapping restrictions 512
 - Attribute Mapping object type 514
 - EIM Interface Table Column object type 512
 - EIM Interface Table object type 512
 - EIM Table Mapping object type 513
 - Foreign Key Mapping Column object type 517
 - Foreign Key Mapping object type 515
 - Interface Table User Key Usage object type 514
 - labeling data as NULL 523
 - User Key Attribute Join object type 517
 - User Key Attribute object type 517
 - User Key Column object type 517
 - User Key object type 517
 - EIM Table Mapping object type** 513
 - EIM Table Mapping Wizard**
 - about 518
 - using to map new table 518
 - EIM Table User Key Usage object type** 514
 - elbows and trees** 373
 - enterprise dock object** 525
 - Enterprise Integration Manager**
 - See EIM interface tables, configuring
 - entities**
 - associating attributes to business component fields 225
 - binding to business components 224
 - defining entity attributes 225
 - modifying entity or relationship properties 228, 229
 - viewing entities for an ERD 226
 - Entity Relationship Designer**
 - about 221
 - associating attributes to business component fields 225
 - binding entities to business components 224
 - binding relationships to links or joins 225
 - copying diagrams 228
 - creating diagrams 224
 - defining entity attributes 225
 - modifying entity or relationship properties 228, 229
 - modifying relationship properties 227
 - moving shapes 230
 - resizing shapes 231
 - showing grid lines 232
 - turning snap to grid on 232
 - viewing entities for an ERD 226
 - viewing relations list for an ERD 226
 - zooming in/out 231
 - eScript**
 - using to enter data for customer dashboard 583
 - explicit login, configuring views for** 278
 - explorer view**
 - See tree applets
 - exporting, applet preview** 308
 - extension columns**
 - about 55, 56, 57, 598
 - creating type LONG 64, 237
 - deleting 239
 - modifying 238
 - renaming 239
 - extension tables**
 - about 43
 - creating one to one extension tables 66, 238
 - deleting 239
 - modifying 238
 - one-to-many extension tables 48
 - one-to-one extension tables 44
 - using predefined one-to-many extension tables 62
 - using predefined one-to-one extension tables 60
- ## F
- FavoritesLabel Web page item** 149, 472
 - Field Read Only Field user property** 86
 - restricting Competitor field 248
 - field values, displaying images for** 465
 - FieldLabel control** 614
 - fields**
 - about and business components 78
 - about calculated fields 81
 - about creating sequence fields 82
 - about data types 81
 - about system fields 89
 - adding sequence fields 246
 - joined, configuring 258
 - file attachment applets, about** 382
 - file control** 614
 - Fixup Administration view** 349
 - Foreign Key Mapping Column object type** 517
 - Foreign Key Mapping object type** 515
 - Form Applet Wizard, using** 325
 - form applets**
 - about 111, 132

- applets that cannot be converted 319
- converting to grid layout 318
- converting using Conversion Wizard 316
- creating 325
- troubleshooting conversions to grid layout 320

form templates

- about (grid) 156
- about that do not use a grid 157

FormSection control 615

G

GetCurrentContactId dashboard

- command 588

GetDashboardFieldValue dashboard

- command 589

global deployment

- about deploying help 541
- language folders 542
- localizing help 541

GoTo

- configuring view 579
- defining list 577

graphics

- about indentation 373
- adding to templates 472

grid layout

- about 315
- applets that cannot be converted 319
- converting form applets to grid layout 318
- converting using Conversion Wizard 316
- copying and pasting items 313
- setting field tab order 312
- troubleshooting conversions 320
- working with 321

grid lines

- showing 232
- turning snap to grid on 232

GROUP BY, about supported for business

- components 123

guidelines

- application configuration guidelines 143
- business object configuration 109
- configuration guidelines 189
- configuration join 94
- configuring applets 125
- configuring business components 76
- configuring keyboard shortcuts 300
- configuring MLOVs 552
- customizing the data objects layer 62
- MLOV configuration and coding 553
- view naming conventions 137

H

help development

- about global deployment 541
- language folders 542
- localizing help 541

hidden control 615

hierarchical list applets, how they are rendered 375

hierarchical lists

- about 405
- configuring 414

hierarchical objects, defining images 467

High Interactivity Enabled property

- settings for views 276

Host Administration view 346

HTML content controls

- administration views 346
- configuring to use Web Content Assets 350
- control properties 348
- controlling behavior 349
- defining hosts 346

HTML control type

- about formats 485
- creating custom type 477
- when SWE uses a custom HTML type 485

HTML file, exporting applet preview 308

HTML frames

- about using in Container Page 150
- using in view web templates 153

I

ImageButton control 615

images

- about displaying images 462
- configuring button 464
- creating bitmap categories and objects 462
- creating bitmap object of type GIF 464
- defining images used in hierarchical objects 467
- displaying images for field values 465
- using images as links in controls 465

implicit joins, about 44

indentation graphics 373

index columns

- about and indexes 57, 598
- about index column object type 57

indexes

- about index column object type 57
- and index columns 57, 598
- creating custom indexes 64, 235

inheritance, about upgrade inheritance 190

input method editors (IMEs), setting mode 312

interface tables, about 511
intersection business components, about 72

intersection tables
 about 50
 how tables are used 51
 intersection data 54
 object definitions 51

iterator tags, about 171

J

Java

about integration with Java EE 35

Java EE, about integration with 35

JavaApplet control 615

JavaScript

using to customize toolbars 460

joined tables, about 71

joins

about 91
 about implicit joins 44
 about implied joins 44
 binding relationships to 225
 configuration guidelines 94
 defining in a business component 258
 join construction diagram 92
 predefault value, using for a join field 260
 roles of objects in joins 92
 using predefault value for join field 260

K

key sequence

hiding 302
 modifying 302

keyboard shortcuts

about 300
 adding 301
 guidelines for configuring 300
 hiding key sequence 302
 modifying key sequence 302

L

label controls 615

label text, changing styles 338

Language Independent Code column, about 543

language mode, about 540

language-independent code, about need for uniqueness 554

leaf icon 373

limited dock object 525

Link Object definition

about and the link object 104

links

about and example 615
 about multi-value group applet and properties 104
 binding relationships to 225
 using images as links 465

links, configuring

about 104
 about Cascade Delete property 261
 construction of multi-value links 96
 defining many-to-many relationship 263
 defining one-to-many relationship 263
 link construction 96
 making multiple associations between the same parent and child records 264
 multi-value links 95
 object definitions 96
 Search Specification property 121
 Visibility Rule property 104

links, defining

allowing users to Set Primaries 251
 defining the Primary ID Field 504

list applet templates

about 159
 about indentation graphics 373
 about tree applet templates 162
 displaying totals of list column values 338
 elbows and trees 373
 enabling current record selection 343
 example list applet template 159
 multi-value group and pick applet 430
 persistently editable list applets 159
 text style parameters 373

List Applet Wizard, using 323

list applets

about 112
 creating 323

list columns

configuration guidelines 127
 configuring Show More/Less button 310
 setting input method editor (IME) mode 312

list-form views 130

Lists of Values

See LOVs, creating and administering

lists, configuring

about creating dynamic list objects 412
 about dynamic lists 395
 about dynamic lists originating applet 400
 about hierarchical lists 405
 about list originating business component 391
 about pick applets 393
 about PickList Generic business component 392

- about static list object 392
 - about static list originating applet 391
 - about static lists 389
 - configuring hierarchical lists 414
 - constraining dynamic lists 403
 - crating static 407
 - data flow 394
 - determining if bounded 546
 - dynamic list object types 397
 - pick applet details 394
 - types of lists 389
 - Locale Management Utility, about** 540
 - Locale object types, about** 540
 - localization, about** 540
 - log file**
 - about MLOV converter log file 560
 - about recompiling and deploying SRF file 541
 - fix LOV types 560
 - login, configuring for explicit login** 278
 - LONG column type, creating** 64, 237
 - LOV, creating and administering**
 - Administration view LOV 548
 - configuring MLOV in Siebel Tools 544
 - process of enabling MLOV 544
 - LOVs, creating and administering**
 - See also* MLOVs
 - about language-independent code 543
 - about lists of values 416
 - about multilingual LOVs 542
 - about organization-enabled LOVs 418
 - about recompiling and deploying SRF file 541
 - adding records for MLOV fields 549
 - adding translated display values 545
 - checking for visibility rules 545
 - configuration considerations 553
 - configuring Anywhere to use MLOV fields 566
 - configuring Assignment Manager to use MLOV fields 565
 - configuring to use MLOV-enabled fields 561
 - configuring Workflow Policy Column 564
 - configuring Workflow Policy Program argument 562
 - creating a language-independent code list 563
 - creating a new language-independent code applet 562
 - creating language-independent code applet for Workflow Policy Program Argument 562
 - creating LIC list for Workflow Policy Program Argument 562
 - creating using Siebel Tools 417
 - deleting/deactivating MLOV records 552
 - determining if list is bounded 546
 - fixing LOV types in log file 560
 - guidelines for configuring MLOVs 552
 - identifying which columns to enable 544
 - integration considerations 554
 - language-independent code, about need for uniqueness 554
 - making sure LOV is translatable 544
 - MLOV configuration and coding guidelines 553
 - MLOV Converter Utility, using to upgrade existing data 555
 - querying MLOVs 551
 - repicking values, workflow policy conditions 564
 - repicking values, workflow program policy arguments 562
- M**
- Mailto control** 615
 - mappings, application specific** 123
 - menus**
 - about 449
 - about applet method menu item 451
 - about Invoke Method Targeting 451
 - about the Class Method Menu item 451
 - about the Command object type 450
 - activating/deactivating 457
 - creating applet menus 458
 - creating command objects 455
 - toolbar and menu-related object types 450
 - Microsoft Windows**
 - resuming MLOV Converter Utility 558
 - running MLOV Converter Utility 555
 - MiniButtons, types of** 114
 - MLOV**
 - Administration views LOVs 548
 - configuring in Siebel Tools 544
 - process of enabling 544
 - MLOV Converter Utility**
 - about log file 560
 - parameters 557
 - resuming in UNIX 558
 - resuming in Windows 558
 - running in a UNIX environment 555
 - running in Windows 555
 - using 555
 - MLOVs**
 - about 542
 - about MLOV converter log file 560
 - about recompiling and deploying SRF

- file 541
 - adding records for MLOV fields 549
 - adding translated display values 545
 - checking for visibility rules 545
 - choosing values, workflow policy conditions 564
 - columns that cannot be enabled 547
 - configuration and coding guidelines 553
 - configuration considerations 553
 - configuring Anywhere to use MLOV fields 566
 - configuring Assignment Manager to use MLOV fields 565
 - configuring the Workflow Policy Column 564
 - configuring to use MLOV-enabled fields 561
 - configuring Workflow Policy Program argument 562
 - creating a language-independent code list 563
 - creating a new language-independent code applet 562
 - creating language-independent code applet for Workflow Policy Program Argument 562
 - creating LIC list for Workflow Policy Program Argument 562
 - deleting/deactivating MLOV records 552
 - fixing LOV types in log file 560
 - guidelines for configuring 552
 - integration considerations 554
 - MLOV Converter Utility parameters 557
 - querying MLOVs 551
 - re picking values, workflow program policy arguments 562
 - upgrading existing data 555
 - multilingual lists of values. See MLOVs**
 - multipart tags, about** 170
 - multivalued group applet**
See MVG applets
 - multi-value group applets**
about originating business component 428
about the detail business component 429
 - multi-value links**
about 95
construction 96
construction of cascade copy 250
 - MVG applets**
about and properties 423
about defining 426
about originating applet 428
about the Link object definition 104
configuring MVG applets 432
configuring shuttle applets 443
creating 430
 - object definitions 425
 - shuttle applets 441
 - MVG Wizard, using** 430
- ## N
- naming conventions**
table prefix naming conventions 42
 - naming formats**
for applets 126
for applets titles 127
suffix table naming formats 42
 - non-licensed objects, usage and configuration** 183
- ## O
- object definitions**
about 21
architecture 23
Business Object layer 28
data object layer 30
Logical User Interface Objects layer 26
summary of object types 31
 - object interfaces**
and scripting 197
browser scripting 198
generating browser scripts 200
server scripting 197
 - object layers**
architectural layers diagram and layers 23
Business Object layer 28
data object layer 30
Logical User Interface Objects layer 26
physical user interface layer 25
summary of object types 31
 - object reuse**
deciding when to reuse objects 213
guidelines 203
reusing applets 209
reusing business component fields and table columns 252
reusing business components 219
reusing tables 205
reusing views 210
 - objects, about modifying** 204
 - objects, copying**
copying business objects and business components 209
copying user interface objects 210
problems caused 204
 - one-to-many extension tables** 48
 - online help development**
language folders 542
 - open/closed folder icon** 373

organization enabled LOVs
 configuring and scripting guidelines 420
 guidelines 420

originating applets
 about detail diagram and property settings 400
 multivalue group applet properties 428
 Static list, property settings 391

originating business components
 about list and property settings 391
 about multi-value group applet and properties 428

P

page tabs, defining 282
Parent Read Only Field user property 87
Parent Read Only Field: user property 88
parent-child relationships, defined 21
parent-child views

about 131
 calling association applets 439

password, interface element 615
performance considerations, and chart applets 509

persistently editable list applets 159
personal layout control, configuring 276
phone number fields, formatting 575
physical user interface layer 25

Pick List Wizard
 creating dynamic list objects 412
 using to create static list 407
PickList Generic business component 392
Policy Conditions, enabling 562, 565

pop-up controls 328
pop-up windows
 configuring 328
 configuring pop-up wizards 331
 creating 327

PositionOnRow, about 615
predefined database extensibility options 59

predefined queries
 about 180
 about Siebel conditional tags 172
 conditional tag: <swe:if> 175
 conditional tag: <swe:if-var> 176
 conditional tags: <swe:switch>, <swe:case>, and <swe:default> 175
 query management commands 181

previewing applet layout
 exporting to HTML file 308
 previewing 307

Pricer, about 39

Primary Id Field
 allowing users to Set Primaries 251
 defining 504

private dock object 525
Properties window, using 227

Q

queries
 about Siebel conditional tags 172
 conditional tag: <swe:if> 175
 conditional tag: <swe:if-var> 176
 conditional tags: <swe:switch>, <swe:case>, and <swe:default> 175
 predefined 180
 query management commands 181

R

RadioButton control 616
read-only behavior
 about 85
 BC Read Only Field user property 86
 Field Read Only Field user property 86
 Parent Read Only Field user property 87
 Parent Read Only Field: user property 88
 restricting Competitor field 248
 warnings on user properties 88

Recent records functionality, configuring 289

RecNavNxt control 616
RecNavPrv control 616
records, adding for MLOV fields 549

recursive trees, about 371
registering views 269
remote users, setting up developers as responsibility, associating to view 269
rich list templates, about 165
roof icon 373
RTCEmbedded text editor 616
RTCEmbeddedLinkField, about 616

S

S_Party table 58
Screen Home Pages
 applets, creating 292
 business object, creating 291
 creating, process of 285
 screen view, creating 296
 view, creating 294

screen menu items
 associating with screens 282
 defining screen menu items 282
screen view objects, defining 283
screen views

- about screen view 133
- hierarchy example 136
- screens**
 - about 133
 - about screen view 133
 - configuring 281
 - creating screen views 283
 - defining sequence for screen view objects 283
 - managing unused screens 508
 - process of creating screens and screen views 281
 - screen view hierarchy example 136
- scripting**
 - and object interfaces 197
 - browser scripting 198
 - generating browser scripts 200
 - server scripting 197
- Search Specification property**
 - about 121
 - defining 599
- search syntax**
 - sorting versus searching, differences 123
- sequence fields**
 - about creating 82
 - adding sequence field 246
 - object definitions 83
- Sequence property, Dock Object object type** 526
- server database**
 - applying data objects layer changes to 242
- server scripting** 197
- Set Primaries, setting** 251
- shapes**
 - modifying shape properties 228, 229
 - moving shapes 230
 - resizing 231
- shortcuts**
 - about 300
 - adding new keyboard shortcut 301
 - guidelines for configuring 300
 - hiding key sequence 302
 - modifying key sequence 302
- Show More/Less button**
 - about and adding to applets 310
 - configuring controls for More mode 311
- shuttle applets**
 - about 441
 - association applets as known as 433
 - configuring 443
- Siebel Anywhere, configuring to use MLOV fields** 566
- Siebel application, personalizing** 37
- Siebel applications**
 - about configuring 184
 - about copying objects 204
 - about modifying objects 204
 - about structuring development work 185
 - about upgrade inheritance 190
 - adding view 269
 - configuration goals and objectives 185
 - configuration guidelines 189
 - configuration strategy 185
 - copying business objects and business components 209
 - copying user interface objects 210
 - deciding when to reuse objects 213
 - development process overview 36
 - object reuse guidelines 203
 - reusing applets 209
 - reusing business component fields and table columns 252
 - reusing business components 219
 - reusing tables 205
 - reusing views 210
 - usage and configuration on non-licensed objects 183
- Siebel Assignment Manager**
 - configuring assignment criteria and skills for MLOVs 565
 - configuring to use MLOV-enabled fields 565
 - configuring Workload rules 566
- Siebel eScript**
 - using dashboard commands with Siebel eScript 589
- Siebel object definitions**
 - about 21
 - architecture 23
 - Business Object layer 28
 - data object layer 30
 - Logical User Interface Objects layer 26
 - physical user interface layer 25
 - summary of object types 31
- Siebel operating architecture**
 - about 32
 - about integration with Java EE 35
 - how Siebel Web Engine generates app 35
 - Siebel Web Engine infrastructure 33
- Siebel Pricer, about** 39
- Siebel Spell Check, configuring**
 - about invoking 334
 - adding button to Web template 335
 - associating business component to business object 335
 - creating a spell check button 334
 - creating spell check menu item 335
 - defining spell check button user properties 334

- process for configuring 334
- Siebel Tags**
 - about 168
 - about iterator tags 171
 - about singleton and multipart tags 170
 - about This tag 170
- Siebel Templates**
 - types of templates 146
- Siebel Tools**
 - about language mode 540
 - configuring MLOV 544
 - creating LOVs 417
- Siebel VB**
 - using to enter data for customer
 - dashboard 583
- Siebel Visual Basic**
 - using dashboard commands Siebel Visual Basic 590
 - using dashboard commands with Siebel Visual Basic 590
- Siebel Web Engine**
 - how generates application 35
 - infrastructure 33
- Siebel Web template**
 - about 145
 - about applet templates 154
 - about catalog-style list applets 165
 - about form templates (grid) 156
 - about form templates (non-grid) 157
 - about HTML frames in Container Page 150
 - about indentation graphics 373
 - about list applet templates 159
 - about rich list templates 165
 - about the Container Page 149
 - about tree applet templates 162
 - about view templates 152
 - about Web Page templates 149
 - catalog-style list applet example 167
 - Container Page areas 149
 - displaying totals of list column values 338
 - elbows and trees 373
 - enabling current record selection in list applets 343
 - example list applet template 159
 - how SWE generates HTML files 147
 - HTML frame in Container Page templates 151
 - multi-value group and pick applet 430
 - persistently editable list applets 159
 - roof, leaf, and open/closed folder icons 373
 - support for multiple views 472
 - text style parameters 373
- Siebel Web templates**
 - about HTML frames view web templates 153
- Siebel Workflow** 38
- singleton tags, about** 170
- SmartScript, configuring for dynamic data capture** 38
- SmartScripts**
 - activating SmartScripts Player applet 586
 - configuring 587
 - configuring to enter data into customer dashboard 586
 - mapping SmartScripts variables to dashboard fields 586
- SmartScripts Player applet, activating** 586
- sort specifications, defining** 246
- sort syntax**
 - sorting versus searching, differences 123
- sorting capabilities, adding** 298
- spell check, configuring** 336
 - about invoking 334
 - adding button to Web template 335
 - associating business component to business object 335
 - creating a spell check button 334
 - creating spell check menu item 335
 - defining spell check button user properties 334
 - process for configuring 334
- SSNxt control** 616
- SSPrv** 616
- state model, about using** 39
- static lists**
 - about 389
 - about originating applet 391
 - about PickList Generic business component 392
 - architecture and object types 393
 - creating static list 407
 - difference from dynamic list 389
 - Pick List object 392
- style sheets**
 - about cascading style sheets 477
- SWE templates**
 - about cascading style sheets 477
 - about formats 485
 - about search and find 172
 - about Search and Find applet tags 172
 - about Search Result applet tags 173
 - about Siebel conditional tags 172
 - adding graphics to templates 472
 - adding sorting capabilities 298
 - conditional tag: <swe:if> 175
 - conditional tag: <swe:if-var> 176
 - conditional tags: <swe:switch>, <swe:case>, and <swe:default> 175
 - creating custom HTML control types 477

- how hierarchical list applets are rendered 375
- predefined queries 180
- query management commands 181
- search result tag:
 - <swe:srchResultField> 174
- search result tag:
 - <swe:srchResultFieldList> 174
- search result tag: <swe:this> 174
- search tag: <swe:srchCategory> 173
- search tag:
 - <swe:srchCategoryControl> 174
- search tag: <swe:srchCategoryList>; 173
- search tag: <swe:srchCategoryText> 174
- when SWE uses a custom HTML type 485

system columns, about 56**system fields**

- about 89
- displaying 351

T**table columns, reusing** 252**Table Wizard**

- columns generated 234
- using to create new tables 233

tables

- about 41
- about columns 55
- about data columns 55
- about database extension options 59
- about extension columns 55, 56, 57, 598
- about extension tables 43
- about implied joins 44
- about including joined tables data in business components 71
- about indexes and index columns 57, 598
- about intersection tables 50
- about S_Party table 58
- about system columns 56
- about user keys 57
- adding columns to predefined tables 63, 236
- adding dock object table to object 535
- applying data objects layer to server database 242
- applying database customization to the local database 240
- creating custom indexes 64, 235
- creating one-to-one extension tables 66, 238
- creating tables using Table Wizard 233
- creating type LONG columns 64, 237
- customizing data objects layer guidelines 62
- deleting extension tables or columns 239

- Dock Object Table object 526
- how intersection tables are used 51
- intersection data 54
- modifying extension tables or columns 238
- one-to-many extension tables 48
- one-to-one extension tables 44
- prefix naming conventions 42
- process for customizing the data objects layer 233
- propagating changes to other local databases 243
- properties 594
- reusing 205
- suffix naming formats 42
- Table Wizard actions 234
- using predefined one-to-many extension tables 62
- using predefined one-to-one extension tables 60

templates

- about displaying toolbars 479

text control 115**text style parameters** 373**TextArea control** 616**This tag, about** 170**thread bar**

- configuring 271

Toolbar Item object type, about 451**Toolbar object type, about** 450**toolbars**

- about 449
- about displaying in templates 479
- about Invoke Method Targeting 451
- about the Command object type 450
- about the Toolbar Item object type 451
- about the Toolbar object type 450
- activating/deactivating 457
- adding new toolbar icon 456
- creating a new toolbar 456
- creating command objects 455
- customizing toolbars using JavaScript 460
- toolbar and menu-related object types 450

translated display values, adding 545**tree applets**

- about 363
- about recursive trees 371
- about tree applet templates 162
- configuring and explorer views 367
- creating tree applets in Applet Layout Editor 370
- file attachment applets 382
- using Tree Applet Wizard 367

troubleshooting

- conversions to grid layout 320

view configuration 279

U

UNIX

resuming MLOV Converter Utility 558

running MLOV Converter Utility 555

unused business components

configuring business components 78

unused screens, managing 508

Update Dashboard command 589

upgrade inheritance, about 190

URL control 616

user interface navigation model 129

user interface object, copying 210

User Key Attribute Join object type 517

User Key Attribute object type 517

User Key Column object type 517

User Key object type 517

user keys, about 57

user properties, adding to customer dashboard 570

V

values, choosing (Workflow Policy Column) 564

verifying dock objects 536

view layout, editing 268

View links, configuring functionality 288

view templates

about 152

about HTML frames in web templates 153

View Wizard, using 267

views

about applet toggles 141

about drilldowns 138

adding view to Siebel application 269

applet toggles example 273

associating to responsibility 269

configuring for explicit login 278

configuring personal layout control 276

configuring secure views 278

configuring the thread bar 271

creating screen views 283

creating views process 267

defining sequence for screen view objects 283

dynamic drilldowns 140

editing view layout 268

list-form views 130

new view, about providing access to 269

parent-child views 131

process of creating screens and screen views 281

registering views 269

reusing 210

settings for High Interactivity Enabled

property 276

static drilldowns 139

troubleshooting view configuration 279

using the View Wizard 267

view naming conventions 137

Virtual business components

rapid search and rapid add, creating 285

virtual business components, about 72

visibility

checking for visibility rules for LOVs 545

Visibility Rule property 104

visibility rules

Dock Object Visibility Rule 526

types 525

Visibility Strength property

Dock Object object type 526

Dock Object Visibility Rule object type 526

W

Web Content Assets, configuring fields 350

Web Page Layout editor, accessing 471

Web Page Objects, configuring

about 471

editing 471

Web Page templates

about 149

about applet templates 154

about catalog-style list applets 165

about form templates (grid) 156

about form templates that do not use a grid 157

about HTML Frames in Container Page 150

about indentation graphics 373

about list applet templates 159

about rich list templates 165

about the Container Page 149

about tree applet templates 162

about view templates 152

catalog-style list applet example 167

Container Page areas 149

displaying totals of list column values 338

elbows and trees 373

enabling current record selection in list applets 343

example list applet template 159

HTML frames in Container Page

templates 151

multi-value group and pick applet 430

persistently editable list applets 159

roof, leaf, and open/closed folder icons 373

- support for multiple views 472
- text style parameters 373

Web page templates

- about HTML frames in view templates 153

Web templates

- about cascading style sheets 477
- about formats 485
- about Search and Find applet tags 172
- about search and find in SWE templates 172
- about Search Result applet tags 173
- about Siebel conditional tags 172
- adding graphics to templates 472
- adding sorting capabilities 298
- adding spell check button 335
- conditional tag: `<swe:if>` 175
- conditional tag: `<swe:if-var>` 176
- conditional tags: `<swe:switch>`, `<swe:case>`,
and `<swe:default>` 175
- creating custom HTML control types 477
- how hierarchical list applets are
rendered 375
- predefined queries 180
- query management commands 181
- search result tag:
 - `<swe:srchResultField>` 174
- search result tag: `<swe:srchResultFieldLis
>` 174

- search result tag: `<swe:this>` 174
- search tag: `<swe:srchCategory>` 173
- search tag:

- `<swe:srchCategoryControl>` 174

- search tag: `<swe:srchCategoryList>` 173

- search tag: `<swe:srchCategoryText>` 174

- when SWE uses a custom HTML type 485

Web templates applets, adding 345

Windows

- resuming MLOV Converter Utility 558

- running MLOV Converter Utility 555

Workflow Policy Column

- choosing 564

- configuring 564

- creating a language-independent code
list 563

- creating a new LOC applet 562

Workflow Policy Program Argument

- configuring 562

- creating LIC list 562

- creating new LIC applet 562

Workload Rules, configuring 566

Z

- zooming in/out 231