Using Siebel Tools

Siebel Innovation Pack 2014

November 2014

Documentation Accessibility

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

Access to Oracle Support

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

Contents

Chapter 1: What’s New in This Release

Chapter 2: Using Siebel Tools

Overview of Using Siebel Tools 13

Roadmap for Setting Up and Using Siebel Tools 14

- Process of Setting Up the Development Environment 14

Using the Object Explorer and Object List Editor 21

- Overview of the Object Explorer and Object List Editor 21
- Locating and Modifying Object Definitions in the Object List Editor 24
- Displaying Object Types in the Object Explorer 26
- Using Lists and Tabs in the Object Explorer 27
- Using the Object List Editor 28

Using Menus and Running Queries 31

- Using the Menu Bar 31
- Using a Right-Click Menu 31
- Running Queries 31
- Getting Documentation About an Object 34

Using Windows, Wizards, and Toolbars 34

- Using the Properties Window 35
- Using the Applets Window 35
- Using the Controls and Columns Window 36
- Using the Palettes Window 37
- Using the Web Template Explorer Window 37
- Using the Multi Value Property Window 37
- Using the Bookmarks Window 38
- Displaying, Docking, and Stacking Windows 38
- Using a New Object Wizard 39
- Using the Toolbars 40

Using Editors in Siebel Tools 41

- Using a Layout Editor 42
- Specifying the Web Template Editor 42
- Using an Editor That Includes a Canvas 43
- Using a Script Editor 45

Using the Command Line 45

Chapter 3: Managing Repositories

Viewing, Renaming, Comparing, and Configuring Repositories 47
- Overview of Managing Repositories 47
- Files That You Use to Manage Repositories 48
- How Siebel Tools Minimizes the Data That it Processes during Exports and Imports 49
- Viewing Information About the Current Repository 49
- Renaming Repositories 50
- Comparing Two Repository Files 52
- Configuring Siebel CRM to Read Data from a Single Repository 53

Managing Modifications That Developers Make to Repositories 54
- How Siebel Tools Associates Repository Modifications with One or More Developers 55
- Using Object Tags to Manage Modifications That Multiple Developers Make 56
- Restricting the Objects That Developers Can Modify 62
- Adding Identification Numbers to Repository Modifications 65

Exporting and Importing Repositories 66
- Overview of Exporting and Importing Repositories 67
- Importing Repositories in Windows Environments 68
- Importing a Repository in UNIX Environments 69
- Exporting Repositories 70
- Importing or Exporting Repositories at a Later Time 71

Upgrading Repositories 71
- Identifying Conflicts That Occur During Upgrades 72

Process of Migrating Repositories 73
- Preparing to Migrate Repositories 73
- Using the Database Configuration Wizard to Migrate Repositories 75
- Updating Siebel Remote Databases 78
- Migrating Non-repository Configurations and Data 78
- Using the Repository Import and Export Utility 80
- Guidelines for Migrating Repositories 82

Deleting Repositories 83

Chapter 4: Checking Out and Checking In Projects and Objects

Overview of Checkout and Checkin 85
- About the Project Object Type 85
- Guidelines for Using Checkout and Checkin 86

Getting Projects from the Server Repository 86

Getting All Projects from the Server Repository 87
Getting Some Projects from the Server Repository 88
Getting Locale Data from the Server Repository 89

Using Checkout and Checkin 89
Allowing Object Locking for a Project 89
Checking Out Projects from the Server Repository 90
Checking Out Objects from the Server Repository 91
Checking In Projects or Objects to the Server Repository 92
Locking a Project in the Local Repository 93
Locking an Object in the Local Repository 93
Unlocking a Project 94
Undoing Project Checkout 94
Preventing Object Checkin and Checkout 95
Viewing Locked Objects in a Project 95
Viewing Differences Between Local Repositories and Server Repositories 95

Setting Options for Checkout and Checkin 96
Configuring Third-Party Code Control 96
Configuring Siebel Tools to Restart Editors After Checkout 101

Customizing Projects 101
Creating New Projects 101
Renaming Projects 102
Modifying the Project That an Object References 103
Modifying the Location Where Siebel Tools Saves Objects and Projects 103

Chapter 5: Customizing Objects

Overview of Customizing Objects 105

Creating, Modifying, Copying, Validating, and Deleting Objects 107
Creating Objects 107
Modifying Objects 108
Copying Objects 108
Validating Objects 109
Deleting Objects 110

Examining Objects 111
Searching the Repository 111
Viewing Object Relationships 113
Comparing and Synchronizing Objects Between Repositories and Archives 114
Comparing Different Versions of a Workflow Process or Task UI 117
Determining When Siebel CRM Created or Updated a Record 118

Chapter 6: Using Siebel Script Editors

Using the Siebel Script Editor 119

- Overview of Using Siebel Script Editor 119
- Setting Options for the Siebel Script Editor 121
- Opening the Siebel Script Editor 122

Using the ST eScript Engine 123

- Overview of Using the ST eScript Engine 123
- Enabling the ST eScript Engine 127
- Setting Options for the ST eScript Engine 127
- Opening the Script Assist Window 128
- Using Fix and Go 128
- Using Running Tool Tip 129
- Using Script Libraries with the ST eScript Engine 130

Using the Siebel Debugger 132

- Overview of Using the Siebel Debugger 132
- Setting Debug Options 133
- Accessing the Siebel Debugger 134
- Validating Script Syntax 134
- Using Breakpoints 135
- Using the Calls Window 135
- Using the Watch Window While it Monitors a Script 136
- Tracing a Script 136

Using the Script Profiler 138

- Overview of the Script Profiler 139
- Enabling the Script Profiler and Line Profiler 141
- Running the Script Profiler 142

Chapter 7: Localizing Strings and Locale Data

Configuring Symbolic Strings 145

- Overview of Configuring Symbolic Strings 145
- Modifying the Configuration File to Support Symbolic Strings 147
- Creating a Project for Your Symbolic Strings 147
- Modifying a Predefined Symbolic String 148
- Creating a New Symbolic String 148
- Modifying a Symbolic String to Globally Update Display Values 149
- Setting a Symbolic String Reference 150
- Entering a String Override 152

Converting Symbolic Strings 153

- Consolidating Symbolic Strings 158
- Using a Batch File to Convert Strings 160

Configuring Locale Data 161
- Configuring Nontranslatable Locale Object Properties 161
- Displaying Controls and List Columns According to Locale 162
- Fixing Orphaned String References After an Upgrade 164

Using the Locale Management Utility 165
- Finding Untranslated Text Strings 166
- Finding Existing Translations 167
- Finding Objects That the Locale Management Utility Modifies 168
- Finding Objects Modified Since the Last Export 169
- Exporting Text Strings and Locale Properties to a File 170
- Importing Text Strings and Locale Properties 171
- Modifying the Value in All Strings for a Language 172
- Using the Command Line to Run the Locale Management Utility 172

Using Advanced Compile to Localize a Repository 175
- Overview of Using Advanced Compile 175
- Using Advanced Compile 176

Chapter 8: Compiling, Testing, and Troubleshooting Your Customizations

Compiling Your Modifications 179
- Overview of Compiling 179
- Compiling Projects 180
- Compiling Objects 182
- Using an Incremental Repository Upgrade Kit 183
- Using the Command Line to Import, Export, or Compile Objects 183

Testing and Troubleshooting Your Modifications 187
- Setting Debug Options to Open the Siebel Client 187
- Recovering from a Failed Development Server 188

Chapter 9: Archiving Objects

Overview of Archiving Objects 189

Exporting Objects to an Archive 190
- Using the Command Line to Export Objects to an Archive 190
- Exporting Only Modified Objects to an Archive 191

Importing Objects from an Archive 195
- Preparing the Siebel Tools Environment to Import Objects from an Archive 195
- Importing an Archive 196
- Importing Multiple Archives 198
- Using the Command Line to Import an Archive 199
- Importing Only Modified Objects from an Archive 200

Using the Review Conflicts and Actions Dialog Box of the Import Wizard 202

Using the Application Deployment Manager 204

Using a Hotfix to Deploy Fixes to the Production Environment 205

Using the Command Line to Export Objects to a Hotfix 205

Exporting Objects to a Mid-Level Release 207

Appendix A: Reference Materials for Siebel Tools

Menus and Menu Items on the Menu Bar 209

File Menu 210

Edit Menu 211

View Menu 212

Screens Menu 214

Go Menu 215

Query Menu 215

Format Menu 216

Debug Menu 216

Tools Menu 217

Window Menu 220

Help Menu 221

Buttons on the Toolbars 222

Edit Toolbar 222

List Toolbar 223

History Toolbar 224

Debug Toolbar 225

Simulate Toolbar 226

Format Toolbar 226

WF/Task Editor Toolbar 228

Configuration Context Toolbar 228

Buttons in the Palettes Window 229

Dialog Boxes That Check Out and Check In Projects and Objects 230

Elements of the Project Check Out Dialog Box 231

Elements of the Object Check Out Dialog Box 234

Elements of the Check In Dialog Box 235

Dialog Boxes That Validate Objects 237

Elements of the Validate Dialog Box 237

Elements of the Validation Options Dialog Box 238

Dialog Boxes That Compare Objects 239

Elements of the Compare Objects Dialog Box 239

Dialog Boxes That Set Development Options 241

Development Options for Project Checkin and Project Checkout 241
Development Options for Visualization Views 242
Development Options for Scripting 242
Development Options for Debugging 244

Parameters That Convert Symbolic Strings 245
Parameters You Use with the Conversion Export Utility 246
Parameters That You Can Use with the Conversion Import Utility 247

Glossary

Index

1 What’s New in This Release

What’s New in Using Siebel Tools, Innovation Pack 2014

Table 1 lists the changes in this revision of the documentation to support this release of the software.

NOTE: Siebel Innovation Pack 2014 is a continuation of the Siebel 8.1/8.2 release.

Table 1. What’s New in Using Siebel Tools, Innovation Pack 2014

Topic	Description
“How Object Tagging Indicates Modifications at the Object Definition Level” on page 55	New topic. Describes how an object tag uses an object definition to track any modifications that the developer makes to this definition. It does not use the properties of an object definition to track these modifications.
“Tagging Objects When Using Siebel Remote” on page 62	New topic. Describes how to configure your environment so that you can use object tagging with Siebel Remote.
“Restricting the Objects That Developers Can Modify” on page 62	New topic. You can restrict the objects that developers can modify, according to a date that you specify.
“Adding Identification Numbers to Repository Modifications” on page 65	New topic. You can associate a feature number or a bug number for each modification that a developer makes to the repository.
“Identifying Conflicts That Occur During Upgrades” on page 72	Modified topic. Provides additional information about the hierarchy report.
“Using the Command Line to Import Only Modified Objects” on page 185	New topic. You can use the command line to import only modified objects according to the objects that an SDF file defines.

What’s New in Using Siebel Tools, Version 8.1/8.2

Table 2 lists changes in this version of the documentation to support this release of the software. The new features described in Table 2 are available in Siebel CRM versions 8.1.1.11, Siebel CRM version 8.2.2.4, and later.

Table 2. What’s New in Using Siebel Tools, Version 8.1/8.2

Topic	Description
“Fixing Orphaned String References After an Upgrade” on page 164	Modified topic. If you set the FixReferences parameter to TRUE, then you must include the PriorRepository parameter.
“Importing Only Modified Objects from an Archive” on page 200	New topic. You can import only the modifications to an object instead of the entire definition of an object.

2 Using Siebel Tools

This chapter describes how to use Oracle’s Siebel Tools. It includes the following topics:

- Overview of Using Siebel Tools on page 13
- Roadmap for Setting Up and Using Siebel Tools on page 14
- Using the Object Explorer and Object List Editor on page 21
- Using Menus and Running Queries on page 31
- Using Windows, Wizards, and Toolbars on page 34
- Using Editors in Siebel Tools on page 41
- Using the Command Line on page 45

Overview of Using Siebel Tools

Siebel Tools is an integrated development environment that you can use to customize Siebel CRM. You can use it to modify predefined Siebel objects or to create custom objects that meet your business requirements. For example, you can use Siebel Tools to modify the data model, modify business logic, or customize the user interface that the Siebel client shows. Siebel Tools allows you to develop a single configuration that you can use to do the following:

- Deploy Siebel CRM across multiple types of clients
- Support multiple Siebel Business Applications and languages
- Upgrade Siebel product releases
- Maintain Siebel CRM

Declarative configuration is a type of programming technique that uses objects and object properties in the Siebel repository to implement the logic that your business requires. Siebel Tools uses declarative configuration to create and modify the object definitions that define a Siebel application. Siebel Tools is not a programming environment. You do not modify the source code or write SQL. Siebel CRM uses the terms object and object definition differently than developers who use programming languages that use similar terms, such as object, object class, or object instance. For more information about the Siebel repository, see Chapter 3, “Managing Repositories.” For more information about the objects, object definitions, and the object hierarchy that Siebel CRM uses, see *Configuring Siebel Business Applications*.

NOTE: The *Siebel Bookshelf* is available on Oracle Technology Network (http://www.oracle.com/technetwork/indexes/documentation/index.html) and Oracle Software Delivery Cloud. It might also be installed locally on your intranet or on a network location.

Roadmap for Setting Up and Using Siebel Tools

To set up and use Siebel Tools, do the following tasks:

1. Process of Setting Up the Development Environment on page 14
2. Specifying the Web Template Editor on page 42
3. Getting All Projects from the Server Repository on page 87
4. Allowing Object Locking for a Project on page 89
5. Checking Out Projects from the Server Repository on page 90
6. Creating, Modifying, Copying, Validating, and Deleting Objects on page 107
7. Checking In Projects or Objects to the Server Repository on page 92
8. Compiling Your Modifications on page 179
9. Testing and Troubleshooting Your Modifications on page 187
10. Exporting Objects to an Archive on page 190

Process of Setting Up the Development Environment

To set up the development environment, do the following tasks:

1. Installing Siebel Tools on page 14
2. Specifying the Data Source That Siebel Tools Uses on page 15
3. Setting Database Options on page 17
4. Choosing the Browser That the Applet Layout Editor Uses on page 17
5. Controlling How Siebel Tools Handles Versions for a Workflow Process or Task UI on page 18
6. Setting the Language Mode on page 19
7. Controlling How Siebel Tools Displays the Confirmation Dialog Box on page 20
8. Resetting Development Tools Options on page 20

This process is a step in "Roadmap for Setting Up and Using Siebel Tools" on page 14.

Installing Siebel Tools

You can install Siebel Tools for different product releases on the same computer. For more information about:

- How to install Siebel Tools, see the *Siebel Installation Guide* for the operating system you are using. For example, see *Siebel Installation Guide for Microsoft Windows*.

- System requirements, such as supported versions of Microsoft Windows, see *Siebel System Requirements and Supported Platforms* on Oracle Technology Network.

  NOTE: For Siebel CRM product releases 8.1.1.9 and later and for 8.2.2.2 and later, the system requirements and supported platform certifications are available from the Certification tab on My Oracle Support. For information about the Certification application, see article 1492194.1 (Article ID) on My Oracle Support.

In this guide, *SIEBEL_TOOLS_ROOT* represents the folder where you install Siebel Tools. This folder is `C:\Program Files\Siebel\`*release_number*`\Tools`, by default.

This task is a step in "Process of Setting Up the Development Environment" on page 14.

Specifying the Data Source That Siebel Tools Uses

This topic describes how to specify the data source so that Siebel Tools and the Siebel Web Client connect to the same local repository. It is recommended that you maintain only one local repository for use with Siebel Tools and with this client. This configuration allows you to use this client to view and test modifications that you make in Siebel Tools. Starting with Siebel CRM version 8.0, if you extract a local repository, and if you use the default settings, then Siebel CRM encrypts the local repository. If you use Siebel Tools to download the local repository, then Siebel CRM does not automatically use the Siebel Web Client to connect to this repository.

This book uses the term Siebel Web Client. You can use the Siebel Web Client or the Siebel Mobile Web Client in most topics that this book describes. For more information about these client types, see *Siebel Deployment Planning Guide*.

This task is a step in "Process of Setting Up the Development Environment" on page 14.

To specify the data source that Siebel Tools uses

1. Open Siebel Tools, click the View menu, and then click Options.
2. In the Development Tools Options dialog box, click the Check In/Out tab.
3. Modify the ODBC data source that the server repository uses:
   a. In the Data Sources section, click the Change button that Siebel Tools shows next to the Server window.

b. In the Change Data Source dialog box, define the ODBC data source parameters. Use information from the following table.

Window	Description
ODBC Data Source	Full string of the ODBC data source that communicates with the server repository.
User Name	User logon Id to access the server repository. Use all upper case.
Password	User password to access the server repository. Use all upper case.
Table Owner	Table owner name to access the server repository.

   For security reasons, Siebel Tools does not store the server credentials between Siebel Tools sessions. You must specify these credentials each time you start a new Siebel Tools session. You must do this before you do a checkin or checkout. For more information, see Chapter 4, "Checking Out and Checking In Projects and Objects."

c. Click OK.

4. Modify the ODBC data source that the local repository uses:

   a. In the Data Sources section, click the Change button that Siebel Tools shows next to the Client window.

   b. Make sure the ODBC data source parameters are identical to the parameters that you set in Step 3, and then click OK.

   For more information, see "Running Simultaneous Siebel Tools Sessions After You Modify the ODBC Data Source" on page 100.

5. Locate the Local section of the following files:

   - The .cfg file that the Siebel client uses. For example, uagent.cfg.
   - The tools.cfg file.

   For more information, see "About the Configuration Files" on page 17.

6. Make sure the Local section of the files you locate in Step 5 use the same values for the following parameters:

   - DSDockEncryptDB
   - DSHashUserPwd
   - DSHashAlgorithm

   If you use Siebel Tools to initialize the local repository, then it sets these parameters in the tools.cfg file. You can copy them to the .cfg file that the Siebel client uses.

For more information about connecting Siebel Tools and the Siebel Web Client to the same local repository, see 475398.1 (Article ID) on My Oracle Support. This document was previously published as Siebel FAQ 229. For more information about extracting a repository, see *Siebel Remote and Replication Manager Administration Guide*.

About the Configuration Files

You can use the following configuration files when you use Siebel Tools:

- **Application configuration file.** A Siebel application references a configuration file when it starts. For example, the configuration file for Siebel Call Center is uagent.cfg. Siebel CRM typically locates this configuration file in the following folder:

  client_root`\bin\`*language_code*

  For example, `C:\Siebel\8.1\Client\BIN\ENU\uagent.cfg`.

  You can customize some of the parameters that this configuration file specifies.

- **Siebel Tools configuration file.** Siebel Tools references the tools.cfg configuration file. You can use it to set various options, such as enabling object checkout or setting default directories. It resides in the following folder:

  SIEBEL_TOOLS_ROOT`\bin\`*language_code*

Setting Database Options

This topic describes how to set database options.

This task is a step in "Process of Setting Up the Development Environment" on page 14.

To set database options

1. In Siebel Tools, click the View menu, and then click Options.
2. In the Development Tools Options dialog box, click the Database tab.
3. Set the options. Use the information in the following table, and then click OK.

Option	Description
Developing for Deployment on DB2 for zSeries	For more information, see *Implementing Siebel Business Applications on DB2 for z/OS*.
Limit Schema Object Names to 18 Characters	For more information, see *Implementing Siebel Business Applications on DB2 for z/OS*.
Allow to Create Column of Type 'Character' Being Greater Than 1	Allows columns of type CHAR to be greater than one character in length. If a column is a CHAR type column, and if the data that this column stores is variable in length, then Siebel CRM pads the data with empty spaces in the database.

Choosing the Browser That the Applet Layout Editor Uses

You can choose the browser that Siebel Tools uses to display applets in the preview mode of the Applet Layout Editor. You can include conditional tags in Web templates that it shows for some browsers but not for others. Choosing a browser allows you to preview how Siebel CRM displays an applet layout for a specific browser. For more information, see "Using Editors in Siebel Tools" on page 41.

This task is a step in "Process of Setting Up the Development Environment" on page 14.

To choose the browser that the Applet Layout Editor uses

1 In Siebel Tools, click the View menu, click Toolbars, and then click Configuration Context.

   For more information, see "Using the Toolbars" on page 40.

2 On the Configuration Context toolbar, click the Target Browser drop-down list, and then click Target Browser Config.

   Siebel Tools displays the Target Browser Configuration dialog box. It includes the following items:

Section	Description
Available Browsers	Lists browser groups that are available.
Selected Browsers for Layout Editing	Lists browser groups that the Web Layout Editor affects.
Capability Name and Value	Lists the capabilities and values for the browser that you choose in the Selected Browsers for Layout Editing section.

3 To add a browser group to the list browsers, click it in the Available Browsers list, and then click the right arrow.

   You can also double-click a browser in the Available Browsers list.

4 Click OK.

   Siebel Tools displays each browser group that you add as a value in the Target Browser drop-down list.

Controlling How Siebel Tools Handles Versions for a Workflow Process or Task UI

You can control how Siebel Tools handles versions for a workflow process or task UI that you open with an editor, depending on the value of the Status property.

This task is a step in "Process of Setting Up the Development Environment" on page 14.

To control how Siebel Tools handles versions for a workflow process or task UI

1 In Siebel Tools, click the View menu, and then click Options.

2 In the Development Tools Options dialog box, click the General tab.

3 In the Workflow and Task Configurations section, set the options, and then click OK. Use descriptions in the following table to help determine the option to set.

Option	Description
Automatic Revision in WF/Task Editor and Version Check	If this check box: ■ Includes a check mark, and if you use an editor to open a workflow process or task UI, and if the Status property of this workflow process or task UI is Completed, then Siebel Tools creates a copy of this workflow process or task UI, sets the status for this copy to In Progress, and then opens an editable version of it in the editor. ■ Does not include a check mark, then Siebel Tools opens the editor, and then displays a version of the workflow process or task UI that you cannot edit.
Automatically Close All the Previous WF/ Task Versions	If this check box includes a check mark, and if you use an editor to open a workflow process or task UI, and if the Status property of this workflow process or task UI is Completed, Not In Use, or Expired, then Siebel Tools closes any prior versions of this workflow process or task UI.

Setting the Language Mode

The *language mode* is a type of mode that allows you to configure Siebel CRM to display text in a language other than English. For example, the German (DEU) language mode allows you to view text that Siebel Tools shows in some objects in German, such as the text that a string contains or object definitions that the Object List Editor shows. The language mode determines the set of locale data that Siebel Tools uses when it compiles the repository, and during checkin and checkout. If you add a language to the Siebel database that does not come predefined with Siebel CRM, then this language must use all capital letters. For more information, see “About Predefined Objects” on page 24 and *Siebel Global Deployment Guide*.

This task is a step in “Process of Setting Up the Development Environment” on page 14.

To set the language mode

1 Make sure the repository includes the language data that Siebel Tools must show.

2 In Siebel Tools, click the View menu, and then click Options.

3 In the Development Tools Options dialog box, click the Language Settings tab.

4 In the Tools Language Mode section, choose a value from the Language drop-down list, and then click OK.

5 (Optional) To enable language override, make sure the Enable and Use Language Override check box includes a check mark.

For more information, see “Enabling Language Override” on page 20.

6 Click OK.

Enabling Language Override

A *language override* is a nontranslatable, locale property that you can configure differently for different locales. For example, you can configure Siebel CRM to use a specific height for the address field in French and to use a different height in English. If you enable language override, then Siebel CRM might create more locale records in the repository. To avoid unnecessary records, it is recommended that you enable language override only if your deployment requires it. For more information, see "Configuring Nontranslatable Locale Object Properties" on page 161 and *Configuring Siebel Business Applications*.

Controlling How Siebel Tools Displays the Confirmation Dialog Box

You can control how Siebel Tools displays the confirmation dialog box.

This task is a step in "Process of Setting Up the Development Environment" on page 14.

To control how Siebel Tools displays the confirmation dialog box

1. In Siebel Tools, click the View menu, and then click Options.
2. In the Development Tools Options dialog box, click the General tab.
3. In the Editing Confirmation Dialogs section, configure Siebel Tools to do one of the following:
   - **Display a confirmation dialog box.** Add a check mark to a check box.
   - **Do not display a confirmation dialog box.** Remove a check mark from a check box.
4. Click OK.

Resetting Development Tools Options

If Siebel Tools behavior is not consistent with the preferences you set, then you can reset them.

This task is a step in "Process of Setting Up the Development Environment" on page 14.

To reset development tools options

- Do one of the following:
  - Reset preferences in the Development Tools Options dialog box:
    - ❑ Click the View menu, and then click Options.
    - ❑ In the Development Tools Options dialog box, click the various tabs and reset preferences, as necessary.
  - Delete the file that stores your preferences:
    - ❑ Close Siebel Tools.
    - ❑ Navigate to the *SIEBEL_TOOLS_ROOT*`\BIN` folder, and then delete the devtools.prf file.
    - ❑ Log in to Siebel Tools, click the View menu, and then click Options.

- Set preferences in the Development Tools Options dialog box.

  Siebel Tools stores the preferences that you set in the Development Tools Options dialog box. It stores these preferences in the devtools.prf file. If you delete this file, then Siebel Tools resets all your preferences to their default values.

Using the Object Explorer and Object List Editor

This topic describes how to use the Object Explorer and Object List Editor. It includes the following information:

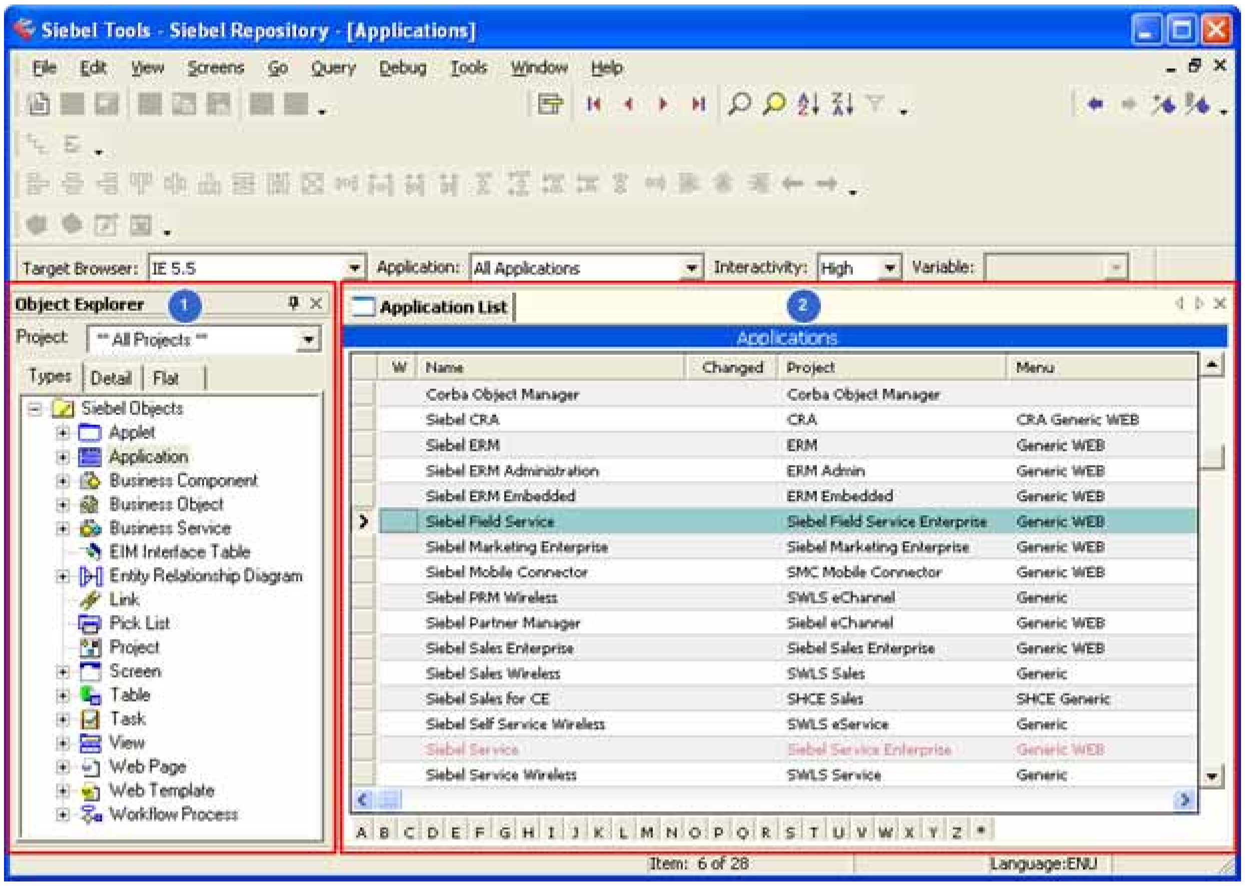
- Overview of the Object Explorer and Object List Editor on page 21
- Locating and Modifying Object Definitions in the Object List Editor on page 24
- Displaying Object Types in the Object Explorer on page 26
- Using Lists and Tabs in the Object Explorer on page 27
- Using the Object List Editor on page 28

Overview of the Object Explorer and Object List Editor

The *Object Explorer* is a window in Siebel Tools that displays the Siebel object hierarchy. It allows you to use a tree structure to navigate through the object types that the hierarchy contains. The *Object List Editor* is a window in Siebel Tools that displays the object definitions of the object type that you choose in the Object Explorer. It allows you to view and modify the object definitions.

How Siebel Tools Displays Object Types and Object Definitions

Figure 1 illustrates how Siebel Tools includes an Object Explorer window and one or more Object List Editor windows. In this figure, Siebel Tools displays the Applications list in the Object List Editor. For more information about the object hierarchy that Siebel CRM uses, see *Configuring Siebel Business Applications*.



Target Browser: IE 5.5 Application: All Applications Interactivity: High Variable:

Project: ** All Projects **

W	Name	Changed	Project	Menu
	Corba Object Manager		Corba Object Manager	
	Siebel CRA		CRA	CRA Generic WEB
	Siebel ERM		ERM	Generic WEB
	Siebel ERM Administration		ERM Admin	Generic WEB
	Siebel ERM Embedded		ERM Embedded	Generic WEB
	Siebel Field Service		Siebel Field Service Enterprise	Generic WEB
	Siebel Marketing Enterprise		Siebel Marketing Enterprise	Generic WEB
	Siebel Mobile Connector		SMC Mobile Connector	Generic WEB
	Siebel PRM Wireless		SWLS eChannel	Generic
	Siebel Partner Manager		Siebel eChannel	Generic WEB
	Siebel Sales Enterprise		Siebel Sales Enterprise	Generic WEB
	Siebel Sales Wireless		SWLS Sales	Generic
	Siebel Sales for CE		SHCE Sales	SHCE Generic
	Siebel Self Service Wireless		SWLS eService	Generic
	Siebel Service		Siebel Service Enterprise	Generic WEB
	Siebel Service Wireless		SWLS Service	Generic

Item: 6 of 28 Language:ENU

Figure 1. Object Explorer and Object List Editor

Explanation of the Callouts

Figure 1 includes the following items:

1 **Object Explorer.** A window that displays object types. Allows you to navigate between each group of object definitions of an object type.

2 **Object List Editor.** A window that displays object definitions. For more information, see "Using the Object List Editor" on page 28.

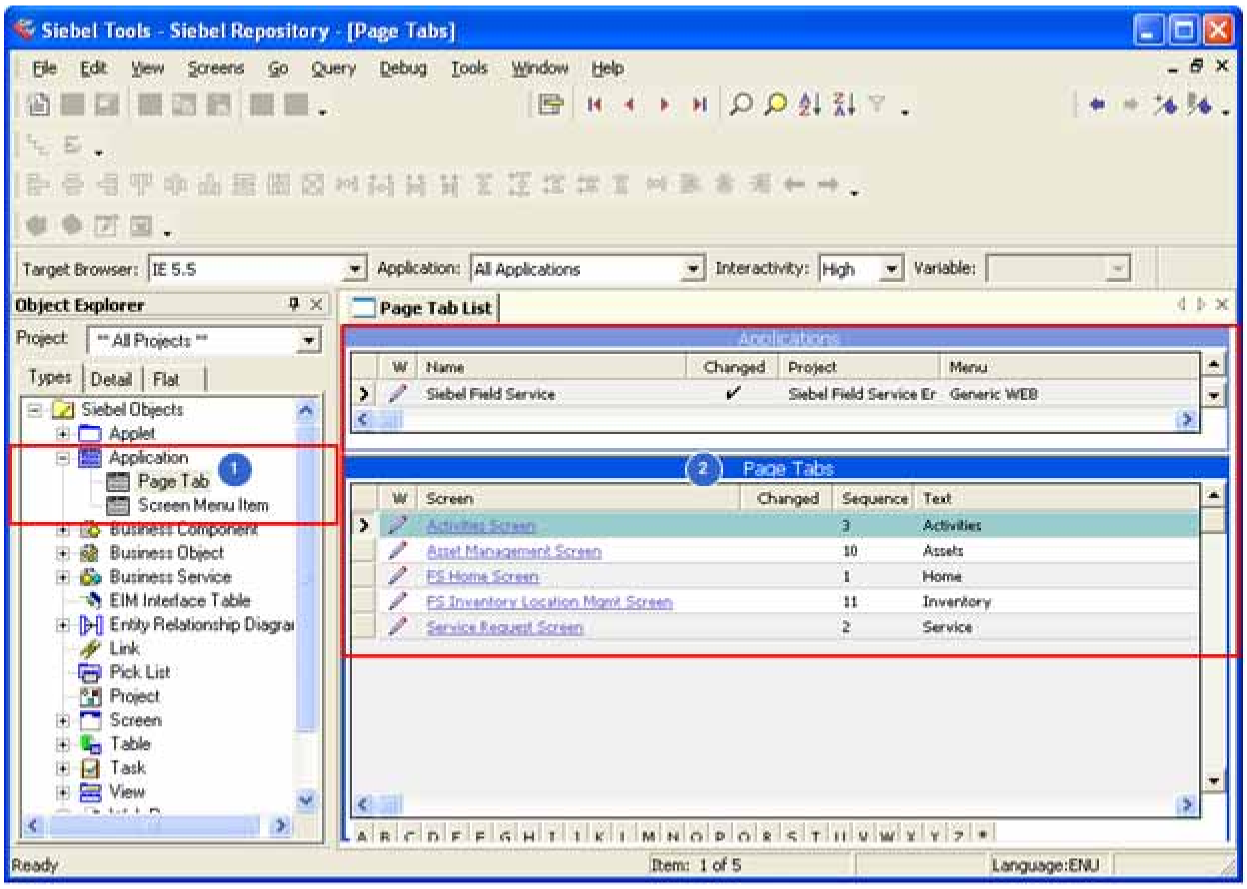
An *object type* is an entity that includes a predefined set of properties. You can use it as a template to create an object definition. An application is one kind of object type. For more information, see "About Predefined Objects" on page 24.

An *object definition* implements one piece of the software. This object definition consists of *object properties*, which are characteristics of this piece of the software. The Object Explorer and Object List Editor display all the object definitions that the repository contains. Siebel Field Service is an example of an object definition of an application. Siebel CRM uses the terms *attribute* and *property*. These terms have the same meaning.

If an object type is not visible in the Object Explorer, then you can display it. For more information, see “Displaying Object Types in the Object Explorer” on page 26.

How Siebel Tools Displays the Siebel Object Hierarchy

Figure 2 illustrates a *parent and child relationship*, which is a type of hierarchical relationship between one object type and another object type.



Target Browser: IE 5.5 | Application: All Applications | Interactivity: High | Variable:

Project: ** All Projects **

Applications

W	Name	Changed	Project	Menu
	Siebel Field Service	✓	Siebel Field Service Er	Generic WEB

Page Tabs

W	Screen	Changed	Sequence	Text
	Activities Screen		3	Activities
	Asset Management Screen		10	Assets
	FS Home Screen		1	Home
	FS Inventory Location Mgmt Screen		11	Inventory
	Service Request Screen		2	Service

Figure 2. How Siebel Tools Displays Relationships Between Object Types

Explanation of Callouts

Figure 2 includes the following items:

1 **Object type hierarchy.** In this example, the Page Tab object type and the Screen Menu Item object type are child objects of the parent Application object type. The Object Explorer displays this relationship as a hierarchical tree that you can expand or collapse. For more information, see “Using the Types Tab” on page 27.

2 **Object definition hierarchy.** The Object List Editor uses two windows to display this parent and child relationship. In this example, the Applications list displays the object definition for the parent Siebel Field Service application. The Page Tabs list displays the object definitions of the child page tabs that the repository contains for the Siebel Field Service application.

If you click the Types tab in the Object Explorer, then Siebel Tools arranges folder icons in a hierarchy. An object type that the Object Explorer shows beneath and slightly to the right of another object type is the child in a parent and child relationship. The object type that the Object Explorer shows above the child object type is the parent. A parent object type can include multiple child object types.

About Predefined Objects

This book uses the term *predefined* to describe an object or configuration that comes already defined when you first install Siebel CRM or Siebel Tools. Each object that the Object List Editor shows after you install Siebel Tools, but before you make any modification, is a predefined object. A *custom object* is a new object that you create. A *modified object* is a predefined object or custom object that you modified.

Locating and Modifying Object Definitions in the Object List Editor

The example in this topic locates the object definitions for a single object type in the Object List Editor, and then modifies the value that resides in a property of this object definition. You use the Object Explorer and Object List Editor to modify the Comment property of the Siebel Call Center application, and then examine the page tabs that this application uses. For information about locating object definitions for multiple object types, see "Searching the Repository" on page 111.

To locate and modify object definitions in the Object List Editor

1 Log in to Siebel Tools.

2 Click the View menu, and then click Object Explorer.

   You can also press CTRL+E.

3 In the Object Explorer, click Application.

   For more information, see "How Siebel Tools Displays Object Types and Object Definitions" on page 21.

4 In the Applications list, query the Name property for Siebel Call Center.

   For more information, see "Using the Query Menu to Run a Query" on page 32.

   If the object definition exists, then the Object List Editor displays it in the Applications list. Querying for a record in this way causes Siebel Tools to display a single, isolated record. This technique helps to make sure that you choose the correct object definition and that it remains chosen while you use the Object Explorer and Object List Editor to navigate the object hierarchy, or if you do other work, such as publishing or revising. It reduces the possibility that you might mistakenly modify another object definition. For more information, see "How Siebel Tools Displays Object Types and Object Definitions" on page 21.

5 Right-click the Siebel Call Center record, and then click Lock Object.

   For more information, see "Locking an Object in the Local Repository" on page 93.

6 Locate the property you must modify, and then type a new value in this property, choose a value from a drop-down list, or add a check mark to a check box.

   In this example, locate the Comments property. Use the scroll bar at the bottom of the Object List Editor to scroll toward the right.

7 Enter the following text in the Comments property:

   ```
   This is the object definition for the Siebel Call Center application.
   ```

   For more information, see "Using the Object List Editor" on page 28.

8 To save your modifications, click anywhere outside of the row.

9 In the Object Explorer, expand the Application tree, and then click Page Tab.

10 Notice the records that the Object List Editor shows in the Page Tabs list.

   For more information, see "How Siebel Tools Displays the Siebel Object Hierarchy" on page 23.

Modifying Multiple Records in the Object List Editor

You can modify multiple records in the Object List Editor.

To modify multiple records in the Object List Editor

1 In the Object List Editor, locate the objects you must modify.

2 To choose multiple records, hold down the CTRL key while you click each record you must modify.

3 Click the Edit menu, and then click Change Records.

4 In the Change Selected Records dialog box, choose the field you must modify, and then enter a value for that field.

5 Click OK.

Displaying Object Types in the Object Explorer

This topic describes how to display object types in the Object Explorer.

To display object types in the Object Explorer

1. Log in to 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 object type you must display.
5. Make sure the object type you must display includes a check mark. Use values from the following table.

Check Box State	Description
☑	Display this object and all child objects of this object.
☐	Hide this object and all child objects of this object.
☑	Display this object. Display some child objects and hide some child objects.

   If you add a check mark to a top-level object type, such as Applet, then Siebel Tools adds a check mark to all child objects that the Applet contains. You can expand the parent tree to display only some child objects. A *top-level object type* is an object type that resides at the top of the object hierarchy. The object types that the Object Explorer shows immediately after you install Siebel Tools, but before you expand any object types in the Object Explorer, are top-level object types.

6. (Optional) To restore the Object Explorer to the settings when you first install Siebel Tools, click Default.
7. Click OK.

Displaying Hidden Object Types and Properties

This topic describes how to display hidden object types and properties. A *hidden object type* is an object type that Siebel CRM does not show in the Siebel Web Client. For more information, see *Siebel Object Types Reference*.

To display hidden object types and properties

1. Close Siebel Tools.
2. Use a text editor to open the `tools.cfg` file.

   For more information, see "About the Configuration Files" on page 17.

3 In the Siebel section, modify the following parameter to `All`:

```
    ClientConfigurationMode
```

4 Save the file, and then restart Siebel Tools.

Using Lists and Tabs in the Object Explorer

This topic describes how to use lists and tabs in the Object Explorer.

Using the Project Drop-Down List

The Object Explorer includes the Project drop-down list that allows you to filter objects according to project. The example in this topic describes how to use the Project drop-down list so that the Object Explorer displays only the object types that reference the Account project. For more information about projects, see Chapter 4, "Checking Out and Checking In Projects and Objects."

To use the Project drop-down list

- In the Object Explorer, click the Project drop-down list, and then choose Account.

  You can choose All Projects in the Project drop-down list to reset the Object Explorer so that it displays objects types for all projects.

Using the Types Tab

Siebel Tools displays the Types tab, by default. You can use it to navigate the object hierarchy. The Object Explorer does not display all object types, by default. For more information, see "Displaying Object Types in the Object Explorer" on page 26. For more information about the object hierarchy, see *Configuring Siebel Business Applications*.

To use the Types tab

1 In the Object Explorer, click the Types tab.

  Siebel Tools displays the top-level object types in alphabet order.

2 To display child object types, expand the tree for an object type.

  To expand the tree, you click the plus sign (+) that the Object Explorer shows to the left of an object type. For example, expand the Business Component tree.

3 To collapse an object type, click the minus sign (–).

Using the Detail Tab

The Object Explorer includes a Detail tab that allows you to view and expand an object type. If you use the Detail tab, then Siebel Tools displays object properties in the Object List Editor only for the object that you choose.

Using the Flat Tab

The Object Explorer includes a Flat tab that displays all object types in a single, alphabetic list. It does not display a hierarchy. You can use it to do the following:

- Find a child object with an unknown parent. For example, assume you create a new field but do not remember the business component that this field references. You can click the Flat tab, click the Field object type, and then query the Name property for the field name. Each record that Siebel Tools returns includes a parent property that displays the business component name.
- Determine how Siebel CRM typically uses objects and properties, such as how it uses a predefault value or the format that it uses in a calculated field.

To use the Flat tab

1. In the Object Explorer, click the Flat tab.

   The Object Explorer displays a list of all the object types in the repository. It displays this list in alphabetic order and with no hierarchy.

2. In the Object Explorer, click Field.

   The Object List Editor displays all the object definitions in the repository for the field object. This example describes how to use the flat tab to identify all the business components that include a field named Account Address.

3. In the Fields list, query the Name property for Account Address.

   The Object List Editor displays a list of all the field object definitions in the repository with the name Account Address.

4. Right-click in the Fields list.
5. In the Columns Displayed dialog box, scroll to the bottom of the Displayed Columns window, choose Parent Business Component, click the move to top arrow, and then click OK.

   The Parent Business Component column in the Object List Editor lists all the business components that include a field named Account Address.

6. In the Parent Business Component property, click Order Entry - Orders.

   The Object List Editor displays the object definition for the Order Entry - Orders business component.

Using the Object List Editor

You can use the Object List Editor to edit the properties of an object definition. For example, if you choose the Application object type in the Object Explorer, then the Object List Editor displays a list of all the applications that the repository contains. One row in the Object List Editor represents one object definition. For example, the values that the Object List Editor shows in the properties of the Siebel Call Center application represent one object definition.

The Object List Editor displays each object property as a column in the list. Information that you enter in each column represents a *property value*. You can modify the property values in an object definition. You cannot modify the set of properties that constitute an object definition. You can also use the Properties window to edit these properties. Modifying the value of a property in the Properties window also modifies the corresponding value of that property in the Object List Editor window. For more information, see "Using the Properties Window" on page 35.

Controlling How Siebel Tools Displays the Object List Editor

You can control how Siebel Tools displays the Object List Editor.

To control how Siebel Tools displays the Object List Editor

1. In Siebel Tools, click the View menu, and then click Options.
2. In the Development Tools Options dialog box, click the List Views tab, and then set the options. Use information from the following table.

Option	Description
Small, Normal, or Large	Sets the size of the font that Siebel Tools shows.
Tight, Normal, or Loose	Sets the spacing that Siebel Tools uses between rows.
Horizontal Grid Lines	Displays or hides horizontal grid lines in the list.
Vertical Grid Lines	Displays or hides vertical grid lines in the list.
Alternating Row Color	Displays a different color for every other row.
Mouse Focus Rectangle	Displays or hides a dotted line around the record that you choose.

Using the W Property

If an object is locked and editable, then Siebel Tools displays a pencil icon in the W property in the Object List Editor. The pencil icon in Figure 2 on page 23 indicates that all objects in the Object List Editor are locked and editable. If you lock a parent object, then Siebel Tools locks all child objects of this parent throughout the hierarchy. If you lock a child object, then Siebel Tools locks all objects that are parents of this child throughout the hierarchy.

Using the Changed Property

If you edit a record, then Siebel Tools adds a check mark in the Changed property. This check mark indicates that you modified this object definition since a specific date and time. If the Changed property does not include a check mark, then no modification has occurred since the date and time that you specify in the General tab of the Development Options dialog box. If you modify a child object, then Siebel Tools also adds a check mark to the Changed property of all objects that are parents of this child in the hierarchy. You can control how Siebel Tools displays the check mark in the Changed property.

To use the Changed property

1. In Siebel Tools, click the View menu, and then click Options.
2. In the Development Tools Options dialog box, click the General tab.
3. In the Changed Date section, set the Date and Time fields, and then click OK.

   If a modification occurs on a record that the Object List Editor shows, and if this modification occurs:

   - **On or after the date you specify.** Siebel Tools displays a check mark in the Changed property.
   - **Before the date you specify.** Siebel Tools does not display a check mark in the Changed property.

Using the Link

If the property value in the Object List Editor references the name of another object, then Siebel Tools displays this value as a link that you can click. If you click this link, then Siebel Tools displays the object definition of the item you clicked.

To use the link

- Make sure you are assigned the Developer responsibility in the Administration - Application screen, Responsibilities view in the Siebel client.

  For more information, see *Siebel Security Guide*.

Using the Inactive Property

If the Inactive property is TRUE, then Siebel Tools colors the item red in the Object List Editor, and then deactivates the record in the repository the next time you compile your modifications. For more information, see "Compiling Your Modifications" on page 179.

To use the Inactive property

- If an object definition becomes obsolete due to an update or due to a new requirement, then you must not delete the unused objects. Instead, it is recommended that you set the Inactive property of this object definition to TRUE.

  Siebel CRM does not reference an inactive object.

Defining the Name of the User Property Object Type

Starting with Siebel Tools version 8.0, it is not necessary to enter the name of a valid user property if you add a user property to an object, such as an applet or business component. Instead, you can use a drop-down list in the Name field of the user property. Siebel Tools then displays a dialog box that lists the valid user properties that you can define for the object. For more information about user properties, see *Siebel Developer's Reference*.

Using Menus and Running Queries

This topic describes how to use menus and run queries.

Using the Menu Bar

The menu bar resides along the top of the Siebel Tools interface. It includes the same menus that a typical Microsoft Windows applications include, such as File, Edit, and Help. You click a menu on the menu bar to display menu items. Siebel Tools disables the menu items that are not available. For more information, see "Menus and Menu Items on the Menu Bar" on page 209.

To use the Menu Bar

- Log in to Siebel Tools, and then click a menu on the menu bar.

Using a Right-Click Menu

A *right-click menu* is a type of context-sensitive menu that Siebel Tools shows if you right-click an object or an element. It allows you to do the following:

- Create, copy, or delete a record.
- Undo modifications that you make to a record.
- Open the Applet Layout Editor or the View Layout Editor from the Object List Editor. You right-click an applet or a view, and then click Edit Web Layout.
- View and edit a Web template. You right-click a Web template in the Object List Editor, and then click View Web Layout.
- Display the name and status of a toolbar. You right-click any toolbar. You can also customize how Siebel Tools displays a toolbar.
- Check out or lock an object.
- Add an object to an archive.
- Access wizards.

Running Queries

This topic describes how to run a query in Siebel Tools. If you use the Query Menu, then Siebel Tools searches for objects according to the conditions that you enter in one or more properties in the Object List Editor. You can use a simple query or a compound query. You can create, refine, or activate a query from the Query menu or from the List toolbar. You can also use shortcut keys instead of using the Query menu.

Using the Query Menu to Run a Query

This topic describes how to use the Query Menu to run a query.

To use the Query Menu to run a query

1. In the Object Explorer, click the object type you must locate.
2. Click the Query menu, and then click New Query.

   Siebel Tools displays an empty record in the Object List Editor.
3. In the Object List Editor, enter the query.

   For more information, see "Using a Simple Query" on page 32 and "Using a Compound Query" on page 33.
4. Click the Query menu, and then click Execute Query.

   Siebel Tools displays the records that meet the query criteria that you entered.
5. (Optional) To refine the query, do the following:
   a. Click the Query menu, and then click Refine.
   b. In the Object List Editor, add more query conditions.
   c. Click the Query menu, and then click Execute Query.

Using Shortcut Keys to Run a Query

You can use a shortcut key to run a query.

To use shortcut keys to run a query

1. In the Object Explorer, click the object type that you must locate.
2. In the Object List Editor, enter CTRL + Q.
3. In the Object List Editor, enter the query.

   For more information, see "Using a Simple Query" on page 32 and "Using a Compound Query" on page 33.
4. Press the ENTER key.

   Siebel Tools displays the query results.

Using a Simple Query

Table 3 describes the operators that you can use to create a simple query. A *simple query* is a type of query that locates records according to one condition. A check mark in a property that can contain a Boolean value represents TRUE. The Changed property is an example of a property that contains a Boolean value. For more information about query operators and Siebel data types, see *Siebel Developer's Reference*.

Table 3. Simple Query Operators

Operator	Description
=	Equal to.
<	Less than.
>	Greater than.
< >	Not equal to.
< =	Less than or equal to.
> =	Greater than or equal to.
*	A wildcard. An asterisk (*) can represent any number of characters, including no characters.
?	A wildcard. A question mark (?) can represent any single character.
IS NOT NULL	Queries for a property that is not empty.
IS NULL	Queries for an empty property.
LIKE	Queries for a value that begins with the string that you enter.
NOT LIKE	Queries for a value that does not start with the string that you enter.
" "	Queries for a string that includes a special character. For example, to query for the MyQuery's Text string, you enter "MyQuery's Text". If your query text includes a special character, then you must use quotes to enclose the query.
EXISTS ( )	Queries for a value in a multi-value group.
~	Forces the case of the text string to use the case that follows the tilde (~).

Using a Compound Query

A *compound query* is a type of query that locates records according to more than one condition. You can use parentheses to control the order that Siebel Tools uses to do a compound query. Siebel Tools runs the query according to the expression that you enter. It starts with the inside parentheses first, in left to right order.

Table 4 describes operators that you can use for a compound query. You can combine simple conditions and compound conditions in a single query. For more information, see *Siebel Developer's Reference*.

Table 4. Compound Query Operators

Operator	Description
AND	All the conditions that the AND operator connects must be true.
OR	At least one of the conditions that the OR operator connects must be true.
NOT	The condition that the NOT operator precedes must be false.

To use a compound query

- Do one of the following:

  a. Enter conditions in two or more properties.

     When you run the query, Siebel Tools uses an AND operator between the conditions that you enter.

  b. Use OR, AND, and NOT operators in a single property.

  c. Use any combination of the conditions that you enter for Step a and Step b.

Removing Query Results from the Object List Editor

You can remove query results from the Object List Editor.

To remove query results from the Object List Editor

1. Click the Query menu, and then click New Query.
2. Do not enter any values in the Object List Editor.
3. Click the Query menu, and then click Execute Query.

Getting Documentation About an Object

This topic describes how to get documentation about an object.

To get documentation about an object

1. Open Siebel Tools.
2. In the Object Explorer, click an object type, such as Business Component, and then press the F1 key.

Using Windows, Wizards, and Toolbars

This topic describes how to use windows, wizards, and toolbars. It includes the following information:

- Using the Properties Window on page 35
- Using the Applets Window on page 35
- Using the Controls and Columns Window on page 36
- Using the Palettes Window on page 37
- Using the Web Template Explorer Window on page 37
- Using the Multi Value Property Window on page 37
- Using the Bookmarks Window on page 38

- Displaying, Docking, and Stacking Windows on page 38
- Using a New Object Wizard on page 39
- Using the Toolbars on page 40

Using the Properties Window

The *Properties window* is a window that displays the properties and the property value for the object that you choose in the Object List Editor. It displays the name of this object near the top of the window. It displays the name of the property in the left column, and the property value in the right column. Siebel Tools displays properties in the Properties window in alphabetic order. You can click the Categorized tab to view these properties grouped according to category. The Properties window does not display the Project property or the Changed property.

In this example, you use the Properties window to modify the Comments property of the Siebel Call Center application.

To use the Properties window

1. Locate the Siebel Call Center application in the Applications list.

   For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

2. Click the View menu, Windows, and then click Properties Window.
3. In the Properties window, enter the following text in the Comments property:

   ```
   I can modify values in the Properties window or in the Object List Editor.
   ```

4. To verify your modifications, examine the Comments property in the Applications list. It includes the values you entered.

Using the Applets Window

The *Applets window* is a window that displays information about a view and allows you to add applets to that view. It includes the List tab and the Icons tab. The Icons tab performs the same function as the List tab except it displays a generic visual representation of the applet type. For more information about editing views and applets, see *Configuring Siebel Business Applications*.

The Applets window includes the same items as the Controls and Columns window except it displays the name of the business object that the view references instead of the applet. For more information, see "Using the Controls and Columns Window" on page 36.

The example in this topic describes how to use the Applets window to add a form applet to the Account Address view.

To use the Applets window

1 In the Views list, locate the Account Address view.

For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

2 Right-click the view that you located in Step 1, and then click Lock Object.

3 Right-click the view that you located in Step 1, and then click Edit Web Layout.

Siebel Tools displays the Applets window and the View Layout Editor. For more information, see "Using a Layout Editor" on page 42.

4 Drag a form applet from the Applets window, and then drop it on the following placeholder in the View Layout Editor:

Child Applet (Hi Display Only)

5 In the Pick Record dialog box, click Name in the drop-down list, enter the following query in the Starting With window, and then click Go:

```
*Address*
```

6 Click Account Service Address Map, and then click Pick.

Siebel Tools adds the Account Service Address Map applet to the View Layout Editor.

7 (Optional) In the View Layout Editor, right-click the applet that you added in Step 4, and then click Edit Applet Web Layout.

You can now edit the Account Service Address Map applet. You can use the Controls/Columns window to add items. For more information, see "Using the Controls and Columns Window" on page 36.

Using the Controls and Columns Window

The *Controls/Columns window* is a window that displays the controls and list columns that you can add to an applet layout if you use the Applet Layout Editor. You drag a control or list column from the Controls/Columns window, and then drop it in a placeholder in the Applet Layout Editor in the same way that you drag an applet from the Applets window.

If you choose a control or a list column in the Controls/Columns window, then the Properties window displays the properties for the object that you choose. The Controls/Columns window includes the following items:

- **Applet.** A field that displays the name of the applet.
- **Template.** A field that displays the Web template that the view references.
- **Change Template.** A button that opens the Choose Template dialog box that allows you to choose another Web template.
- **Edit Template.** A button that opens the Web template editor. For more information, see "Specifying the Web Template Editor" on page 42.
- **Mode.** A drop-down list that displays the view mode, such as Base or Edit.

For an example that describes usage that is similar to usage for the Controls/Columns window, see "Using the Applets Window" on page 35.

Using the Palettes Window

The *Palettes window* is a window that allows you to add items to an object. For more information, see "Editors That Include a Canvas" on page 44.

To use the Palettes window

1. Open the editor that you use to edit an object.

   For example, to edit an applet layout, you open the Applet Layout Editor. When you open an editor, Siebel Tools displays the Palettes window. For an example that uses the Applet Layout Editor, see "Using the Applets Window" on page 35.

2. Drag, and then drop items from the Palettes window to the editor.

Using the Web Template Explorer Window

The *Web Template Explorer window* is a window that displays the HTML code of a Siebel Web Template (.swt) file.

To use the Web Template Explorer Window

1. In Siebel Tools, click the View menu, Windows, and then click Web Templates Window.

   To filter the templates, you can use the Web Template drop-down list in the Web Template Explorer.

2. In the Web Templates window, choose the Web template that you must modify.

   Siebel Tools displays the HTML code for the .swt file in a separate window.

3. To edit the HTML code, right-click the window that displays this code, and then click Edit Template.

Using the Multi Value Property Window

The Multi Value Property Window allows you to view and set values for multi-value properties when you use one of the following designers:

- Entity Relationship Designer
- Task Designer
- Workflow Process Designer

For more information about multi-value properties and how to use the Multi Value Property Window, see *Siebel Business Process Framework: Task UI Guide* and *Siebel Business Process Framework: Workflow Guide*.

Using the Bookmarks Window

The *Bookmarks window* is a window that allows you to navigate directly to an object that you use frequently. For more information, see "History Toolbar" on page 224.

Displaying, Docking, and Stacking Windows

This topic describes how to display, dock, and stack windows.

Displaying a Window

You can display or hide a window.

To display a window

- In Siebel Tools, click the View menu, click Windows, and then click the window that you must display.

  You can also click the Go menu, and then click Bookmarks List to display the Bookmarks window.

- To hide a window, do one of the following:
  - Click the Close button that Siebel Tools shows in the upper right corner of the window.
  - Right-click the window that you must hide, and then click Hide.

Docking a Window

You can dock a window in a corner of the main window.

To dock a window

- Drag the window to the area of the main window where you must dock the window.
- To undock a window, you can right-click the window, and then click Docked.
- To prevent Siebel Tools from docking a window when you move it, you can hold down the CTRL key while you move the window.

Displaying a Window as a Tab

You can display a docked window, including the Object Explorer, as a tab. You can open, close, or dock a tabbed window.

To display a window as a tab

- Right-click the window, and then click Hide.

  Siebel Tools collapses the window and displays a tab for it along the far left margin in Siebel Tools.

- To display a tabbed window:
  - **Temporarily.** Mouse over the tab that represents the window. Siebel Tools displays the window.
  - **Permanently.** Mouse over the tab that represents the window, and then click the push pin that Siebel Tools shows in the window title bar.

Stacking Windows

You can stack windows on top of each other if they are floating.

To stack windows

- Drag, and then drop a floating window onto another floating window.

  Siebel Tools creates a single window that includes both windows, and it adds a tab that represents each window. You can click the tabs to navigate between these stacked windows.

Resetting Windows

You can reset windows to the display that Siebel Tools uses immediately after you first install it.

To reset windows

- In Siebel Tools, click the View menu, and then click Reset Windows.

  Siebel Tools closes all windows except the Object Editor and the Object List Editor. It pins the Object Editor on the left, and the Object List Editor on the right, and resizes them to their original size.

Using a New Object Wizard

A *new object wizard* is a feature that you can use that guides you through the steps of creating a new object. It prompts you to configure the properties and objects that Siebel Tools requires according to the object type that you are creating. You can use a new object wizard to create the following objects:

- **General object.** For example, a business component, table, view, or applet method menu item.
- **Applet object.** For example, a list applet, form applet, MVG Applet, or chart applet.
- **EAI object.** For example, an integration object.
- **Task object.** For example, a task, task applet, task view, or transient business component.

For more information about how to use a wizard for a particular object type, see *Configuring Siebel Business Applications*.

To use a new object wizard

1 Do one of the following:

   - Click the File menu, and then click New Object.

     Siebel Tools displays the New Object Wizards dialog box.

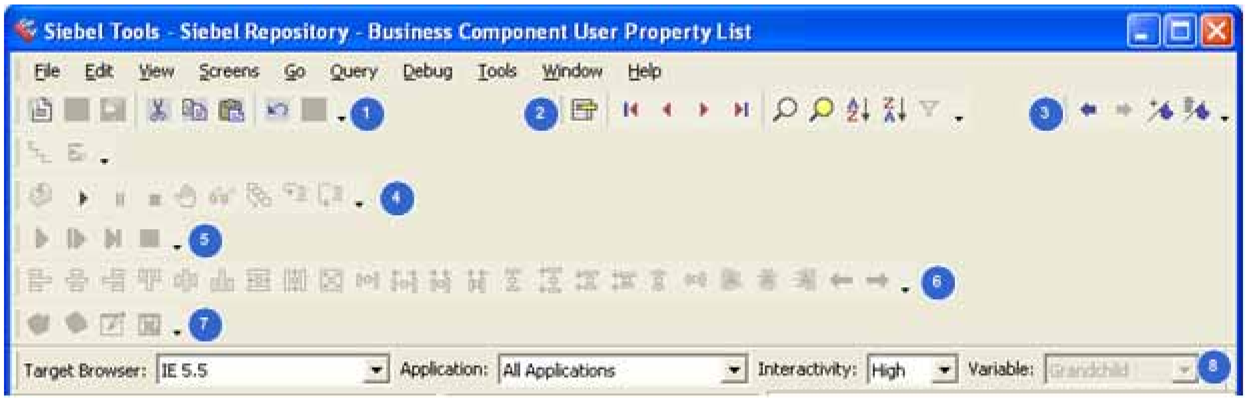
   - Right-click an object in the Object List Editor, and then click New Object Wizards.

     Siebel Tools displays a list of wizards that are specific to the object type that you choose.

2 Choose a wizard.

Using the Toolbars

Figure 3 illustrates the toolbars that you can use in Siebel Tools. A toolbar is active only if the object type or window that uses it is active. To rearrange toolbars, you can drag and drop them. You can display and hide toolbars.


Siebel Tools - Siebel Repository - Business Component User Property List
File Edit View Screens Go Query Debug Tools Window Help
1
2
3
4
5
6
7
8

Target Browser: IE 5.5 Application: All Applications Interactivity: High Variable: Grandchild

Figure 3. Toolbars You Can Use in Siebel Tools

Explanation of Callouts

Siebel Tools includes the following toolbars:

1 **Edit Toolbar.** Allows you to use edit tools, the New Object wizard, and undo and redo options. You can right-click a field in the Object List Editor to display a menu of edit tools. For more information, see "Using a Layout Editor" on page 42.

2 **List Toolbar.** Allows you to insert a new record, move forward and backward through records, query records, and sort records. It manages records that the Object List Editor shows.

3 **History Toolbar.** Allows you to retrace your steps, and to create and navigate to bookmarks.

4 **Debug Toolbar.** Allows you to access debug tools that you can use with Siebel VB and Siebel eScript. For more information, see Chapter 6, "Using Siebel Script Editors."

5 **Simulate Toolbar.** Allows you to simulate a workflow process.

6 **Format Toolbar.** Allows you to apply a format to an applet control. This control uses a Web template that uses a grid layout.

7 **WF/Task Editor Toolbar.** Allows you to publish, activate, revise, or expire a workflow process or task UI.

8 **Configuration Context Toolbar.** Allows you to define settings for Web browser layout and scripting.

Displaying a Toolbar

You can display a toolbar.

To display a toolbar

- In Siebel Tools, click the View menu, click Toolbars, and then click the menu item that represents the toolbar that you must display.

  If Siebel Tools displays a check mark next to the menu item that represents the toolbar, then the toolbar is visible.

Displaying the Status Bar

You can display the status bar.

To display the status bar

- In Siebel Tools, click the View menu, and then click Status Bar.

  If Siebel Tools displays a check mark next to the Status Bar menu item, then the Status Bar is visible.

Using Editors in Siebel Tools

This topic describes how to use some of the editors that you can use in Siebel Tools. It includes the following information:

- Using a Layout Editor on page 42
- Specifying the Web Template Editor on page 42
- Using an Editor That Includes a Canvas on page 43
- Using a Script Editor on page 45

Using a Layout Editor

Siebel Tools includes the following layout editors:

- Applet Layout Editor
- View Layout Editor
- Web Page Layout Editor
- Applet Menu Layout Editor

A layout editor allows you to do the following:

- Add or map a control or list column to an applet layout.
- Modify an existing view or create a new view. You can drag and drop an applet to the View Layout Editor. You can view a list applet, a form applet, or the container page in Preview mode.
- Add or delete a control from a Web page template, modify a control property, or map a control to a placeholder.
- Edit the menu structure of a Siebel application. You right-click an applet in the Object List Editor, and then click Edit Web Menus.
- Preview an approximation of how Siebel CRM displays an applet or a Web page at run time.

For more information about using layout editors, see *Configuring Siebel Business Applications*.

To use a layout editor

1. In the Object List Editor, locate the object that you must modify.

   You can open a layout editor from an applet, view, or Web page. For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

2. Right-click the object that you located in Step 1, and then click Edit Web Layout or Edit Web Menus.

Specifying the Web Template Editor

The *Web template editor* is an application that is external to Siebel Tools that allows you to edit a Web template. For example, if you right-click a Web template in the Web template explorer, then Siebel Tools opens the editor that you specify and displays this Web template in the editor. You can specify the external application that Siebel Tools opens.

This task is a step in "Roadmap for Setting Up and Using Siebel Tools" on page 14.

To specify the editor for Web template files

1. In Siebel Tools, click the View menu, and then click Options.
2. In the Development Tools Options dialog box, click the Web Template Editor tab.
3. In the Folder full path field, enter the full path to where your Web template files reside.

4 Under External Web template editor, do the following:

   a Use the Browse button in the Executable full path field to navigate to and choose the executable for the external Web editor.

   b In the Optional parameters field, enter the parameters that Siebel Tools must send to the executable when it opens the external editor.

5 Click OK.

Using an Editor That Includes a Canvas

A *canvas* is a background that Siebel Tools shows in different designers. It includes a work area that allows you to drag, and then drop objects. For example, the Task Designer allows you to drag a Siebel Operation step from the Palette window, and then drop it on the canvas. The canvas also indicates whether or not you can edit an object.

The example in this topic uses the canvas that Siebel Tools shows in the Task Designer for the FS Asset To Contract Task.

To use an editor that includes a canvas

1 In the Object Explorer, click Task.

2 In the Tasks list, locate the FS Asset To Contract Task.

   Siebel Tools displays two records for the FS Asset To Contract Task. For more information, see "How Siebel Tools Displays Object Types and Object Definitions" on page 21.

3 Use multiselect to choose both records that Siebel Tools shows in the Tasks list.

   For more information, see "Choosing More Than One Record in the Object List Editor" on page 44.

4 Right-click the tasks that you chose in Step 3, and then click Lock Object.

   For more information, see "Locking an Object in the Local Repository" on page 93.

5 Use the canvas that Siebel Tools shows for an object that you can edit:

   a Right-click the task that includes In Progress in the Status property, and then click Edit Task Flow.

      Siebel Tools displays the canvas as a white grid. A white grid indicates that you can edit the task. It displays task objects on top of this grid.

   b To verify that you can edit the task, do the following:

      ❏ Drag a Siebel Operation step from the Palette window to the canvas. For more information, see "Using the Palettes Window" on page 37.

      ❏ Move an existing object.

   c Click the File menu, and then click Close.

   d In the Confirm dialog box, click No.

6 Use the canvas that Siebel Tools shows for an object that you cannot edit:

a. Right-click the task that includes Completed in the Status property, and then click Edit Task Flow.

   Siebel Tools displays the canvas as a solid white background with no grid. This solid white background indicates that you cannot edit the task. Some editors, such as the Workflow Process Designer, use a pale yellow canvas to indicate that you cannot edit the object.

b. To verify that you cannot edit the task, do the following:

   - Attempt to drag a Siebel Operation step from the Palette window to the canvas.
   - Attempt to move an existing object.

c. Click the File menu, and then click Close.

7. Right-click the tasks that you chose in Step 3, and then click Unlock Object.

Choosing More Than One Record in the Object List Editor

You can use multiselect to choose more than one record in the Object List Editor.

To choose more than one record in the Object List Editor

- Do one of the following:
  - To choose records that are not consecutive, hold down the CTRL key while you click each record.
  - To choose multiple records that are consecutive, click a record, hold down the SHIFT key, and then click another record.

Editors That Include a Canvas

Table 5 describes the editors that include a canvas. You can use these editors with the Palettes Window.

Table 5. Editors That Include a Canvas

Editor	Description
Applet Layout Editor	Allows you to add interface controls to an applet layout. Button, Label, and CheckBox are examples of controls that the Palettes window shows. For more information, see "Buttons in the Palettes Window" on page 229.
Entity Relationship Designer	Allows you to diagram the business entities that your configuration uses and represent relationships between these entities. For more information, see *Configuring Siebel Business Applications*.

Table 5. Editors That Include a Canvas

Editor	Description
Task Designer	Allows you to create a user interface that assists the user in completing a job task. Allows you to add steps and connectors to a task UI. For more information about task UI and using the Task Designer, see *Siebel Business Process Framework: Task UI Guide*.
Workflow Process Designer	Allows you to define, manage, and enforce the business processes that your company uses. Allows you to add steps and connectors to a workflow process. For more information about workflow processes, using the Workflow Designer, and using the Workflow Simulator, see *Siebel Business Process Framework: Workflow Guide*.

Using a Script Editor

If you cannot use declarative configuration to meet your requirements, then you can use the Server Script Editor and the Browser Script Editor in Siebel Tools to create a script. You use these editors to add a script to a Siebel object. You can use the following types of script:

- Siebel VB
- Siebel eScript
- Browser Script

For information about declarative configuration, see "Overview of Using Siebel Tools" on page 13. For information about scripting, see Chapter 6, "Using Siebel Script Editors."

Using the Command Line

For more information about using the command line, see the following information:

- Using the Command Line to Validate Objects on page 110
- Using the Command Line to Run the Locale Management Utility on page 172
- Using the Command Line to Import, Export, or Compile Objects on page 183
- Using the Command Line to Export Objects to an Archive on page 190
- Using the Command Line to Import an Archive on page 199
- Using the Command Line to Export Objects to a Hotfix on page 205

3 Managing Repositories

This chapter describes how to manage repositories. It includes the following topics:

- Viewing, Renaming, Comparing, and Configuring Repositories on page 47
- Managing Modifications That Developers Make to Repositories on page 54
- Exporting and Importing Repositories on page 66
- Upgrading Repositories on page 71
- Process of Migrating Repositories on page 73
- Deleting Repositories on page 83

Viewing, Renaming, Comparing, and Configuring Repositories

This topic describes how to view, rename, and compare repositories. It includes the following information:

- Overview of Managing Repositories on page 47
- Viewing Information About the Current Repository on page 49
- Renaming Repositories on page 50
- Comparing Two Repository Files on page 52
- Configuring Siebel CRM to Read Data from a Single Repository on page 53

Overview of Managing Repositories

A *Siebel Repository* is a set of tables that includes Siebel objects and server scripts. These objects and scripts define a Siebel application, such as Siebel Call Center. It provides Siebel CRM with the metadata that it requires to interact with enterprise data and to interact with the people who use the Siebel client. Siebel CRM stores these tables in a compressed Siebel Repository File (SRF) during installation, and then reads the repository at run time. You can use Siebel Tools to view and modify the information that this repository contains. For more information, see *Configuring Siebel Business Applications*.

CAUTION: You must use only one repository in a production environment. You must synchronize the compiled repository and repository table data. If you configure Siebel CRM to use multiple Siebel repositories, then unpredictable behavior might result.

Siebel CRM includes *Incremental Repository Merge*, which is a feature that allows you to merge multiple repositories to a single repository during an upgrade. It automatically does some of this upgrade work for you, such as importing SIF files and seed data, applying schema modifications, and compiling the SRF. For more information, see *Siebel Database Upgrade Guide*.

Files That You Use to Manage Repositories

A *Siebel ArchIve File* (SIF) is a type of file that you can use to import object definitions into or export objects from an SRF. You can use Siebel Tools to create an SIF file, and you can use SIF files to move your modifications from a source environment to a destination environment. Siebel Tools writes SIF files in XML format. A SIF file uses the following hierarchy, and it can include property values and scripts:

1. Repository
2. Project
3. Object
4. Child Objects

For example, the following code is part of an SIF file that Siebel Tools creates when it adds objects to a hotfix. It includes a definition for the Account Assoc Applet:

```
<REPOSITORY
  NAME="Siebel Repository"
  ... >
  <PROJECT
     ...
     NAME="Account (SSE)"... >
     <APPLET
        ASSOCIATE_APPLET="Account Assoc Applet"
        BUSINESS_COMPONENT="Account"
        CLASS="CSSFrameListBase"
        ...
        NAME="Account List Applet"
        ... >
        <APPLET_METHOD_MENU_ITEM... >
        </APPLET_METHOD_MENU_ITEM>...
     </APPLET>
     <BUSINESS_COMPONENT
        CACHE_DATA="N"
        CLASS="CSSBusComp"
        ... >
     </BUSINESS_COMPONENT>
     ...
  </PROJECT>
</REPOSITORY>
```

For more information about:

- This example, see “Using a Hotfix to Deploy Fixes to the Production Environment” on page 205

- Using Siebel Tools to import or export an SIF file as an archive file, see Chapter 9, “Archiving Objects.”

How Siebel Tools Minimizes the Data That it Processes during Exports and Imports

You can use Siebel Tools to create a *Siebel Delta File* (SDF), which is a type of file that is similar to an SIF file except that an SIF file contains the entire definition of an object, while the SDF file contains only information about modifications to an object. The SDF file contains enough information about the object hierarchy so that it can get the modified information.

For example, assume you use Siebel Tools to modify the contents of a comment field that resides in the Contact business component. Assume you use one of the following file types to move this modification from a source environment to a destination environment:

- **SIF file.** Siebel CRM exports the entire Contact business component into an SIF file, and then imports this file into the destination environment. It must compare each part of data that the SIF file contains to the counterpart data that the repository contains that resides in the destination environment.
- **SDF File.** Siebel CRM exports only the contents of the comment field. It does not export the entire Contact business component. This export is similar to the export that Siebel Tools does with an SIF file except that Siebel Tools does the export according to the object tag or object tag and time. For more information about object tags, see “Managing Modifications That Developers Make to Repositories” on page 54. For more information about using SDF files, see “Exporting Only Modified Objects to an Archive” on page 191 and “Importing Only Modified Objects from an Archive” on page 200.

Viewing Information About the Current Repository

Only one repository is available to your local database, and only one repository is available to the server database for your development workgroup. If you log in to Siebel Tools, and if you connect to the local database or the server database, then Siebel Tools opens this repository, by default. In some situations, multiple repositories might reside on the Siebel Server. For example, if you upgrade to a new version of Siebel CRM.

To view information about the current repository

1. In Siebel Tools, click the File menu, and then click Open Repository.

   Siebel Tools displays the Open Repository dialog box. This dialog box lists the repositories in the database that Siebel Tools uses. Siebel Tools highlights the repository that it is currently using.

2. Click the Help menu, and then click About SRF.

3 Examine the information that the About Repository File dialog box shows. Use information from the following table.

Field	Description
Internal Version	Version number that Oracle maintains. Siebel CRM modifies this value only if the internal format of the repository is modified. For example, during an upgrade.
User Version	Reserved for use by Oracle's Siebel Anywhere. It maintains this number when Oracle creates kits that upgrade the repository. Siebel CRM reads this value when it does a version check.
Full Compile	Choose this option to display information about the most recent full compile.
Last Incremental Compile	Choose this option to display information about the most recent incremental compile. If no incremental compile has occurred since the last full compile, then this option is not available.
When	Date of the last compile.
Machine Name	Name of the computer that Siebel CRM uses to compile the repository.
Language	Language code.
User Name	The Microsoft Windows logon name of the user who compiled the repository.
Repository	Repository name of the repository that was current when Siebel CRM performed the compile. This value is typically Siebel Repository.
Tools Version	The version number and build number of the Siebel Tools software that performed the compile. Global Customer Support can use this information to help you resolve a problem.
Schema Version	Database schema version of the Siebel database that Siebel CRM uses to compile the repository.
File Name	File name and path of the repository that Siebel Tools is currently using. The default folder is *SIEBEL_TOOLS_ROOT*\OBJECTS.

Renaming Repositories

It is recommended that you use Siebel Repository as the name of the repository that you use in your production environment. If you must rename the repository, then it is recommended that you use the procedure that this topic describes.

To rename repositories

1 Make sure all developers check in all projects that are currently checked out from the repository that you must rename.

For more information, see Chapter 4, "Checking Out and Checking In Projects and Objects."

2 Log in to Siebel Tools, connected to the server database.

3 Display the Repository object type.

   For more information, see "Displaying Object Types in the Object Explorer" on page 26.

4 In the Object Explorer, click Repository.

5 In the Repositories list, locate the repository that you must rename.

   For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

6 Enter the new name in the Name property, and then step off the record to save your modifications.

   For more information, see "Guidelines for Naming Repositories" on page 51.

7 Compile your modifications.

   For more information, see "Compiling Your Modifications" on page 179.

8 Communicate the name of the new repository to your developers and request that each developer get all projects from the server repository.

   For more information, see "Getting All Projects from the Server Repository" on page 87.

9 Modify the value of the Siebel Repository enterprise parameter to the new name of the repository.

   For information about how to modify an enterprise parameter, see *Siebel System Administration Guide*.

10 Modify the Application Main Repository Name parameter in the Object Manager.

Guidelines for Naming Repositories

If you name a repository, then it is recommended that you use the following guidelines:

- Use a naming convention for all repositories that your Siebel CRM implementation uses. Siebel Servers reference a repository by name. The procedure that you use to upgrade to a new version of Siebel CRM depends on the repository name. A consistent naming convention promotes successful configuration and testing and minimizes the work required to migrate a new repository or to do an upgrade.
- Use the default Siebel Repository name, where possible. You can modify this name only if it is absolutely necessary. The default configuration that Siebel CRM uses assumes that the repository name is Siebel Repository.
- Use the same repository name in your test environment that you use in your production environment. Using the same name simplifies the process of migrating a repository from development to test and from test to production. It also eliminates the need for you to modify the client configuration or server configuration during a migration. For more information, see "Process of Migrating Repositories" on page 73.

- If your development environment uses multiple repositories, then use a unique and descriptive name for each repository. For example, you might use Siebel v8.2 Original as the name of the repository when you install Siebel CRM. You can use another descriptive name for the repository that you use during an upgrade and for a repository from a prior custom configuration.

Comparing Two Repository Files

You can use the SRFDiff utility to compare two repository files. The differences between these files can include new, deleted, or modified objects. The SRFDiff utility saves any modifications that it finds in an XML file. If it does not find any differences, then it does not create this file.

To compare two repository files

1. Log in to Siebel Tools.
2. Click the Help menu, and then click Technical Support.
3. Make sure the repository files that you must compare use the same locale and language.
4. Make sure the repository files that you must compare are part of the same major schema version.

   The SRFDiff utility validates only repository files that are part of the same major schema version. For example, it can validate repositories that each use a different minor schema version, such as 44.39.0.248 and 44.40.0.1, where 44 identifies the major schema version.
5. Click the Tools menu, Utilities, and then click Compare SRFs.
6. In the Compare SRFs dialog box, enter the following information:
   - **Full path to the first repository file.**
   - **Full path to the second repository file.**
   - **Full path for the output file.** If you do not specify a location, then the SRFDiff utility saves the output file to the *SIEBEL_TOOLS_ROOT*\OBJECTS folder.
7. Click Compare.

   The SRFDiff utility does the following work:
   - Validates each repository for file parameters, such as file existence, version, and language.
   - Creates one or more of the following output files:
     - **diff_srf1_srf2.xml.** Includes a description of the differences between the repository files. It includes a list of new and modified objects. It might also include new objects that it finds in one repository but not in the other repository. It uses an attribute name and value list to display the differences that it finds during the comparison. You can use this information to identify a modified or new object.
     - **deleted_records_diff_srf1_srf2.xml.** Includes a description of the deleted objects. In includes the definitions of objects in one repository but not in the other repository.
8. Analyze the differences in the xml file.

   You can manually inspect the results. A modified object includes the following comment:

```
<!-- Changed
```

A new object includes the following comment:

```
<!-- Start: New object section
-->
```

Using the Server Manager Command-Line Interface to Start the SRF Differences Utility

You can use the Server Manager command-line interface to start the SRFDiff Utility.

To use the Server Manager command-line interface to start the SRF Differences Utility

1. Open the Server Manager (srvrmgr) command-line interface.
2. Enter the following command:

   ```
   siebdev /srfdiff "pathname1" "pathname2" "output_pathname"
   ```

   where:

   - *pathname1* is the full path and file name of one of the repository files that you must compare.
   - *pathname2* is the full path and file name of one of the repository files that you must compare.
   - *output_pathname* is the full path where the utility saves the output file.

   For example, assume you must compare the siebel.srf file to the siebel_1.srf file, and that these files reside in the `C:\Programs Files\Siebel Tools 8.1\OBJECTS\ENU\` folder. In this example, you enter the following command:

   ```
   siebdev /srfdiff "C:\Programs Files\Siebel Tools 8.1\OBJECTS\ENU\siebel.srf"
   "C:\Programs Files\Siebel Tools 8.1\OBJECTS\ENU\siebel_1.srf" "C:\Programs
   Files\Siebel Tools 8.1\OBJECTS\"
   ```

Configuring Siebel CRM to Read Data from a Single Repository

The example in this topic describes how to configure Siebel CRM to read data from a single repository when the Siebel database contains multiple repositories.

To configure Siebel CRM to read data from a single repository

1. Log in to the Siebel client with administrative privileges.
2. Navigate to the Administration - Order Management screen, and then the Message Types view.
3. Click Payload, and then click New.

4 Click the Response Field drop-down list.

If the Response Field drop-down list:

- **Does not display duplicate values.** The Siebel database does not contain multiple repositories, and you can exit this task.
- **Displays duplicate values.** The Siebel database contains multiple repositories, and you must configure Siebel CRM to read data only from a single repository. Continue to Step 5.

5 Configure Siebel CRM to read data only from a single repository:

a Log in to Siebel Tools.

b In the Object Explorer, click Pick List.

c In the Pick Lists list, query the Name property for the following value:

```
UMS PickList Response Field
```

d Modify the Search Specification property. Use values from the following table.

Property	Value
Search Specification	[Buscomp] = 'UMS Response' and [Repository Id] = RepositoryId()

Siebel CRM displays the UMS Type Variables List Applet in Step 5. This applet references the UMS Type Variable business component. It displays the Response Field Name field and uses the UMS PickList Response Field picklist for this field. This picklist uses the following search specification when it references the UMS Pick List Field business component:

```
[Buscomp] = 'UMS Response'
```

The UMS Pick List Field business component references the S_FIELD table. It does not include a search specification, so Siebel CRM does not filter the records that it gets according to the specifications that the repository contains. If the Siebel database contains multiple repositories, then Siebel CRM gets all the field names from all repositories, and then displays them in the picklist. This configuration might result in Siebel CRM displaying duplicate values that you cannot choose in the picklist. To correct this situation, you create a search specification that configures Siebel CRM to read data only from the current repository.

6 Compile your modifications.

7 Repeat Step 1 through Step 4.

8 Make sure the Response Field drop-down list does not display duplicate values.

Managing Modifications That Developers Make to Repositories

This topic describes how to manage the modifications that developers make. It includes the following information:

- How Siebel Tools Associates Repository Modifications with One or More Developers on page 55
- Using Object Tags to Manage Modifications That Multiple Developers Make on page 56
- Restricting the Objects That Developers Can Modify on page 62
- Adding Identification Numbers to Repository Modifications on page 65

How Siebel Tools Associates Repository Modifications with One or More Developers

Object tagging is a feature that Siebel Tools uses to associate repository modifications with one or more developers. You can use it to export all the work that a group of developers performs. For example, assume you work on a development team named ABC that includes three developers. Object tagging allows you to extract all the work that the ABC team performs. To identify a developer as part of this group, each developer chooses the ABC tag in the Tag Name field in the login dialog box when the developer logs in to Siebel Tools. For information about exporting the objects that a developer tags, see "Exporting Only Modified Objects to an Archive" on page 191.

How Object Tagging Indicates Modifications at the Object Definition Level

An object tag uses an object definition to track any modifications that the developer makes to this definition. It does not use the properties of an object definition to track these modifications. For example, assume a developer uses different tags to modify the properties of the following items:

- **Same object definition.** If this developer creates an SDF for either tag, then this SDF includes all the modifications for this object definition. For more information about the SDF, see "How Siebel Tools Minimizes the Data That it Processes during Exports and Imports" on page 49.
- **Different object definitions.** If this developer creates an SDF for either tag, then this SDF includes only the modifications for the tag that the developer was using when this developer did the modification.

In the following examples, assume that Siebel Tools assigns the Tag1 and Tag2 tags to Developer1.

User Modifies Properties of the Same Object Definition

If a user uses two different tags to modify properties of the same object definition, then the SDF includes each of these modifications. For example, assume Developer1 does the following work:

- Logs in to Siebel Tools and chooses Tag1 during the login.
- Sets the Force Active property of the First Name field of the Account business component to Y, and then logs out.
- Logs in to Siebel Tools and chooses Tag2 during the login.
- Sets the Required property of the First Name field of the Account business component to Y, and then logs out.

- Creates an SDF for Tag1.

  In this situation, Developer1 used different tags to modify different properties of the same object definition, so this SDF indicates that the Force Active property of the First Name field is Y, and that the Required property of the First Name field is Y.

User Modifies Properties of Different Object Definitions

If a user uses two different tags to modify properties of different object definitions, then the SDF includes only one of these modifications. For example, assume Developer1 does the following work:

- Logs in to Siebel Tools and chooses Tag1 during the login.
- Sets the Force Active property of the First Name field of the Account business component to Y, and then logs out.
- Logs in to Siebel Tools and chooses Tag2 during the login.
- Sets the Required property of the Last Name field of the Account business component to Y, and then logs out.
- Creates an SDF for Tag1.

  In this situation, Developer1 used different tags to modify the properties of two different object definitions, where one object definition exists for the First Name field, and another, different object definition exists for the Last Name field. So this SDF indicates only that the Force Active property of the First Name field is Y. It includes no information about the Last Name field.

Using Object Tags to Manage Modifications That Multiple Developers Make

This topic describes how to use object tags to manage the modifications that multiple developers make. It includes the following information:

- Administering Object Tags on page 56
- Enabling Object Tagging in Siebel Tools on page 57
- Tagging Objects on page 58
- Tagging Objects That You Do Not Modify on page 59
- Tagging Objects When Using Siebel Remote on page 62

Administering Object Tags

This topic describes how to administer object tags.

To administer object tags

1. Consult with your team members to identify the tags that your deployment requires.

   For example, ABC team - developer 1, ABC team - developer 2, and so forth.

2. Log in to the Siebel client with administrator privileges.
3. Enable object tagging:
   a. Navigate to the Administration - Application screen.
   b. Click System Preferences.
   c. Query the Name field for Enable Object Tagging.
   d. Set the value in the System Preference Value field to TRUE.
4. Navigate to the Administration - Tag screen.

   The Tags list displays all the tags that the repository contains.
5. Click New to create one of the tags that you identified in Step 1:
   a. Add information in the Name and Description field that describes that tag.
   b. Click Users, and then add developers to the tag.
6. Repeat Step 5 for each tag that your deployment requires.
7. Administer an existing tag:
   a. Choose the tag that you must administer in the Tags list.
   b. Click All Objects to identify the objects that are associated with a tag.

      The All Objects list displays information about all the objects that developers have created, modified, or deleted. For example, if you choose the Siebel Sales tag in the Tags list, then it displays objects only for developers who use the Siebel Sales tag when they log in to Siebel Tools. For a description of the format that this list uses in the Object Path field, see "Example of an Object List File" on page 60.
   c. Click Conflicts Only to view conflicts.

      The Conflicts Only list displays all the objects that are in conflict and details for each of these objects. The grandchild list displays the details of other tags that are associated with the conflict.

Enabling Object Tagging in Siebel Tools

This topic describes how to enable object tagging.

To enable object tagging in Siebel Tools

1. Make sure you use a version of Siebel Tools that supports object tagging.
2. Set up the shortcut that you use to start Siebel Tools to start in iPack mode:
   a. Right-click the shortcut that you use to open Siebel Tools, and then click Properties.
   b. In the Properties dialog box, click the Shortcut tab.
   c. In the Target field, add the iPackMode switch.

      For example:

```
C:\Siebel\8.1\Tools_2\BIN\siebdev.exe /c
"C:\Siebel\8.1\Tools_2\bin\enu\tools.cfg" /u SADMIN /p SADMIN /d Sample /
iPackMode
```

Bold font indicates the switch that you add.

3 Log in to Siebel Tools.

4 Click the Screens menu, System Administration, and then click System Preferences.

5 In the System Preferences list, create a new record. Use values from the following table.

Property	Value
System Preference Name	Enable Object Tagging
System Preference Value	TRUE

To disable object tagging, you can set the System Preference Value to FALSE.

6 Log out of Siebel Tools.

Tagging Objects

This topic describes how to tag objects.

To tag objects

1 Make sure you enable and administer object tagging.

If you enable object tagging, then you must choose a tag when you log in to Siebel Tools or when you use the command-line interface. For more information, see "Administering Object Tags" on page 56 and "Enabling Object Tagging in Siebel Tools" on page 57.

2 Log in to Siebel Tools.

If you associate:

- More than one tag with a developer, then Siebel Tools displays the Select a Development Tag Name dialog box during the login, and allows you to pick a tag.
- Only one tag with a developer, then Siebel Tools automatically chooses this tag during the login.

You associate developers to tags in Step 5 on page 57.

After you finish the login, Siebel Tools automatically tags objects if you do any of the following work:

- Create or modify an object
- Import an SIF file
- Check in or check out

- Do a manual update

To use a different tag after you log in, you must log out of Siebel Tools and then choose another tag when you log back in to Siebel Tools.

To identify the tag that you are currently using, do one of the following:

- View the tag name that Siebel Tools shows in the Title bar.
- Click the View menu, click Options, click the General tag, and then view the value that Siebel Tools shows in the Development Tag Name window.

Tagging Objects That You Do Not Modify

Touch is a feature in Siebel Tools that allows you to tag an object that Siebel Tools has not tagged as a result of you importing an SIF file, checking in or checking out an object, or doing a manual update. It uses this tag information to identify the objects that it extracts from an SDF file during a repository merge. If you touch an object, then you cannot remove the tag that Touch adds.

Siebel Tools automatically tags the objects that it imports or checks in. If Siebel Tools requires an object during an SDF export, and if this object is not tagged, then you can touch the object to make it available to the export. For information about how to export the objects that a developer tags, see "Exporting Only Modified Objects to an Archive" on page 191. For more information about SDF files, see "How Siebel Tools Minimizes the Data That it Processes during Exports and Imports" on page 49.

To tag objects that you do not modify

1. Make sure you enable object tagging.

   For more information, see "Enabling Object Tagging in Siebel Tools" on page 57.

2. In the Object List Editor, locate the object that you must tag.
3. Right-click the object that you located in Step 2, and then click Touch Object.

   Siebel Tools does the following:

   - If this object is:
     - **Not already tagged.** It adds an entry in the Tag Object table.
     - **Already tagged.** It updates the Last Updated date in the Tag Object table.
   - Displays the following message:

     `The Object is Touched`

Using a Batch Operation to Tag Objects That You Do Not Modify

This topic describes how to tag objects that you do not modify in batch.

To use a batch operation to tag objects that you do not modify

1. Create an object list file.

   For more information, "Example of an Object List File" on page 60.

2 Open a command-line interface in the following folder:

```
SIEBEL_TOOLS_ROOT\bin
```

3 Enter the following command:

```
siebdev /c tools.cfg /l language_code /u user /p password /d database /iPackMode
/t "tag_name" /BatchTouchObj object_list_file_name log_file_name
```

where:

- *language_code* identifies the language code. For example, enu.
- *user* identifies the Siebel Tools user.
- *password* identifies the password that you use to log in to Siebel Tools.
- *database* identifies the database where the object resides.
- *tag_name* identifies the name of the tag. This is the same name that you typically click in the Tag Name field in the log-in dialog box that you use to open Siebel Tools.
- *object_list_file_name* identifies the full path to and the name of a text file that includes a list of the objects that Siebel Tools must tag. For more information, "Example of an Object List File" on page 60.
- *log_file_name* identifies the file that Siebel Tools uses to store the log entries that it creates during the batch operation.

For example:

```
siebdev.exe /c tools.cfg /d ServerDataSrc /u sadmin /p sadmin /iPackMode /t
"Siebel Mobile" /BatchTouchObj C:\Touch\TouchObjList.txt C:\Touch\touchobj.log
```

Example of an Object List File

The Object List file is a text file that includes a list of the objects that Siebel Tools must tag. You can use any file name that meets Windows file name requirements. You can locate this file anywhere on the computer where Siebel Tools resides. The Object List file uses the following format:

```
object hierarchy,object type
.
.
.
object hierarchy,object type
```

where:

- *object hierarchy* identifies the full object hierarchy of the object that Siebel Tools must tag.
- *object type* identifies the type of object that Siebel Tools must tag.
- , (a comma) separates the object hierarchy from the object type. You must not include a space before or after the comma.

The file must not include any empty lines.

For example, you use the following format to tag the control user property of a control that resides in an applet:

```
applet_name||control_name||control_user_property_name,Control User Prop
```

where:

- Two vertical bars (||) separate two different levels in the object hierarchy.

For example, the following code specifies the control user property named Url of the CancelQuery control that resides in the ABO Bulk Request Component Product Pick Applet:

```
ABO Bulk Request Component Product Pick Applet||CancelQuery||Url,Control User Prop
```

If you specify an object in the TouchObjList.txt file, and if this object does not exist in the SRF, then Siebel Tools creates an error message in the log file.

Examples of Specifying Objects That Do Not Include the Name Property

You use the Name property to specify most objects. If an object does not include a Name property, then you can use the next significant property. For example, you use the following Field property for a pick map:

```
business_component_name||field_name||pick_map_field_property,Pick Map
```

The following code specifies the Account Status pick map of the Account Status field that resides in the Account business component:

```
Account||Account Status||Account Status,Pick Map
```

For another example, you use the following Table property for a join:

```
business_component_name||table,Join
```

The following code specifies the S_CONTACT table of a join that resides in the Account business component:

```
Account||S_CONTACT,Join
```

Tagging Objects When Using Siebel Remote

This topic describes how to configure your environment so that you can use object tagging with Siebel Remote. Siebel Tools does not synchronize any modifications that you make when you use Siebel Remote. It communicates only with the repository that resides on the Siebel Server through project Check Out, project Check In, or a Get. You do not use Siebel Tools to synchronize the modifications that you make through object tagging in a Siebel Remote client. Instead, you must use the synchronization features that Siebel Remote provides. For more information, see Chapter 4, “Checking Out and Checking In Projects and Objects” and *Siebel Remote and Replication Manager Administration Guide*.

To tag objects when using Siebel Remote

1. Make sure every developer who uses the Siebel Remote client uses the MOBILE CLIENT - STANDARD routing model when this client extracts the local database.

   You must not use the MOBILE CLIENT - EXTRACT ONLY routing model because it initializes only the local database. It does not synchronize it. For more information about how to use these routing models and Siebel Remote, *Siebel Remote and Replication Manager Administration Guide*.

2. Make sure the tools.cfg configuration file and the Local ODBC data source that the Siebel Web Client uses each reference the same database file, such as sse_data.dbf. For more information about configuring this file and data source, see “Specifying the Data Source That Siebel Tools Uses” on page 15.

3. Make sure each developer synchronizes any modifications that they make through object tagging to the Siebel Server, and that each developer periodically synchronizes their local databases with the Siebel Server so that their clients include the object tagging modifications that other developers make.

Restricting the Objects That Developers Can Modify

In some situations, it might be helpful to restrict the objects that developers can modify, according to a date that you specify. If you enable this feature, then Siebel Tools applies these restrictions when the developer does the following work:

- Check in or check out a project or object. For more information, see Chapter 4, “Checking Out and Checking In Projects and Objects”.
- Import or export an SIF or SDF file. For more information, see “Files That You Use to Manage Repositories” on page 48.
- Directly modify the Siebel Database.

To restrict the objects that developers can modify

1. Enable the system preferences:

a. Log in to Siebel Tools, connected to the Siebel Server.

   You must make sure that you log in to Siebel Tools, connected to the Siebel Server. If the values that you set for the system preferences in this procedure do not match between the Siebel Server and the local computer, than Siebel Tools does not allow you to check in your modifications.

b. Click the Screens menu, click System Administration, and then click System Preferences.

c. In the System Preferences list, set values for the following system preferences.

System Preference Name	System Preference Value
EnablePerfFieldModification	Set the value to TRUE. This system preference applies edit and copy restrictions to objects.
Enable Object Version Control	Set the value to TRUE.
ServerEditRecordTimeStamp	Enter the time when Siebel Tools must start using this configuration. For more information, see "Setting the Server Edit Record Time Stamp Parameter" on page 63.

2. Log out of Siebel Tools.

Setting the Server Edit Record Time Stamp Parameter

When you set the ServerEditRecordTimeStamp parameter, you enter the time when Siebel Tools must start using this configuration. Use the following format:

```
MM/DD/YYYY
```

You can specify only the month, day, and year. You cannot specify minutes and seconds.

For example, to start using the configuration on October 9, 2014, enter the following value:

```
10/09/2014
```

In this example, Siebel Tools does the following:

- Allows you to modify any object that includes a timestamp that occurs after October 9, 2014.
- Does not allow you to modify any object that includes a timestamp that occurs before October 9, 2014. Developers can view or copy these objects, but not modify them.
- Allows you to create new records, and then to modify these new records.
- Allows you to copy any record, regardless of the timestamp.

  If you set a property value in the source record, if this property affects database performance, and if you connect Siebel Tools to the:

  - **Siebel Server,** then Siebel Tools resets the property value in the copied record, and you cannot reset this value.

- **Local database,** then you cannot check in this record.

For more information, see "Properties That Affect Database Performance" on page 64.

Restricting Access to an Object

This topic describes how to restrict access to an object.

To restrict access to an object

1. Determine the date that Siebel Tools most recently updated the object:
   a. In the Object List Editor, choose the record where you must restrict access.
   b. Click Help, and then click About Record.
   c. In the Siebel Tools dialog box, in the Updated section, note the date that the On window shows.
2. Set the value for ServerEditRecordTimeStamp to the date that you noted in Step c.

Properties That Affect Database Performance

Table 6 lists the properties that affect database performance.

Table 6. Properties That Affect Database Performance

Object Type	Property
business component	The following properties affect database performance: ■ Force Active
field	The following properties affect database performance: ■ Force Active ■ Link Specification ■ Immediate Post Changes
multi value link	The following properties affect database performance: ■ Check No Match
multi value field	The following properties affect database performance: ■ Force Active ■ Link Specification
single value field	The following properties affect database performance: ■ Force Active ■ Link Specification

Overriding Restrictions That Prevent Developers from Modifying Objects

This topic describes how to log in to Siebel Tools so that the developer can modify any object, regardless of the timestamp restrictions that you place in Step 1 on page 62.

To override restrictions that prevent developers from modifying objects

1 In Microsoft Windows, click Start, All Programs, and then click the Siebel Tools installation folder.

2 Right-click the Siebel Tools icon, and then click Properties.

3 In the Siebel Tools Properties window, click the Shortcut tab, add the EditPerfFields switch to the Target field, and then click OK.

   For example:

   ```
   C:\Siebel\8.1\Tools_2\BIN\siebdev.exe /c
   "C:\Siebel\8.1\Tools_2\bin\enu\tools.cfg" /u SADMIN /p SADMIN /d Sample /
   EditPerfFields
   ```

   Bold font indicates the switch that you add.

4 Log in to Siebel Tools.

Adding Identification Numbers to Repository Modifications

In some development environments, you might find it useful to associate an identification number with each modification that a developer makes to the repository. You can use this feature to help manage and maintain development work, particularly when your organization includes multiple developers who work across multiple development initiatives. For example, you can use this identification number to track all the modifications that your development team makes for a feature, or to help track the resolution of a bug that uses a unique identification number.

To add identification numbers to repository modifications

1 Enable the system preferences. Do Step 1 on page 62.

2 Log out of, and then log back into Siebel Tools.

3 Modify an existing object, or create a new one.

4 Check in the project or object:

   a Right-click the object that you modified in Step 3, and then click Check In.

   b In the Check In dialog box, click Check In.

c. In the Repository Object Check-in Description dialog box, in the Enter Feature No/Bug No/RF No field, enter a number.

   You can specify any alphanumeric number up to a maximum length of 15 characters. Siebel Tools assigns this number to all objects that you check in.

For more information, see Chapter 4, “Checking Out and Checking In Projects and Objects”.

5. (Optional) Locate all the modifications that are associated with an identification number:

   a. Start SQLPlus, connected to the Siebel Database that resides on the Siebel Server.

   b. Run the following SQL query:

```
SELECT column REL_FEATURE_TXT FROM S_TAG_OBJECT WHERE
REL_FEATURE_TXT = 'identification_number'
```

where:

- *identification_number* is the number that you entered in Step 4. You must enclose this number with single quotes.
- S_TAG_OBJECT is the Siebel CRM table that contains all of the identification numbers.
- REL_FEATURE_TXT is the column of the S_TAG_OBJECT table that contains the identification numbers.

For example, if you specify an identification number of myObjects123 in Step 4, then you run the following SQL query:

```
SELECT column REL_FEATURE_TXT FROM S_TAG_OBJECT
WHERE REL_FEATURE_TXT = 'myObjects123'
```

Siebel Tools adds the identification number to the REL_FEATURE_TXT column the first time that you check in the object. For each subsequent checkin of this object, it saves a unique number in the same column. To do this, it appends the subsequent identification number to the existing identification number. It uses two vertical bars (||) to separate these numbers.

To locate all identification numbers, run the following SQL query:

```
SELECT * FROM S_TAG_OBJECT
```

You must run this query against the Siebel database that resides on the Siebel Server.

For more information about SQLPlus, see the SQL*Plus® User's Guide and Reference on the Oracle Documentation Web site at

http://docs.oracle.com/cd/B19306_01/server.102/b14357/toc.htm

Exporting and Importing Repositories

This topic describes how to export and import repositories. It includes the following information:

- Overview of Exporting and Importing Repositories on page 67
- Importing Repositories in Windows Environments on page 68
- Importing a Repository in UNIX Environments on page 69

- Exporting Repositories on page 70
- Importing or Exporting Repositories at a Later Time on page 71

Overview of Exporting and Importing Repositories

You can use the Database Configuration Wizard to export or import a repository. You can use this utility to do the following work:

- Back up or restore a repository.
- Move all repository objects to another environment that uses the same physical database schema that the source environment uses.
- Export objects to an archive so that you can export or import only some objects. For more information, see "Exporting Objects to an Archive" on page 190.

For more information about how to use the Database Configuration Wizard, see *Siebel Database Upgrade Guide* and the *Siebel Installation Guide* for the operating system you are using.

Guidelines for Exporting and Importing Repositories

If you use the Database Configuration Wizard to import or export a repository, then it is recommended that you use the following guidelines:

- If you import a custom repository, then the Database Configuration Wizard restores all the languages that are part of the predefined repository when you do the import. For example, if you archive repositories weekly, and if your development repository includes ENU and DEU, then the wizard includes ENU and DEU when it imports one of the archived repositories. For more information, see "About Predefined Objects" on page 24.
- If you modify the repository, then make sure you compile all projects. Make a backup copy of the repository file in case you must compare it to the contents of the updated repository. If necessary, you can use this comparison to verify that the import is identical to the backup. For more information, see "Compiling Your Modifications" on page 179.
- You can use the database utilities that your RDBMS vendor provides to back up the entire contents of the Siebel database.
- If you customize the source repository, then you can use the Migrate option of the Database Configuration Wizard. For more information, see "Process of Migrating Repositories" on page 73.

Character Encoding That Siebel CRM Supports to Import or Export Repositories

Table 7 describes the character encoding that Siebel CRM supports when it imports or exports a repository. These databases must use the same Siebel version.

Table 7. Character Encoding That Siebel CRM Supports to Import or Export a Repository

Source Database	Target Database
Code Page	Code Page
Unicode	Unicode
Code Page	Unicode

Where the Database Configuration Wizard Saves Log Files

If you export a repository in a Windows or UNIX environment, and if you use the Export Repository option of the Database Configuration Wizard, then this wizard saves log files in following directories:

- *SIEBSRVR_ROOT*`\log\exprep\output`
- *SIEBSRVR_ROOT*`\log\exprep\state`

where:

- exprep is the default process name for the exprep utility. You can modify this value.

Importing Repositories in Windows Environments

This topic describes how to import a repository in a Windows environment.

To import repositories in Windows environments

1. Make sure Siebel CRM supports the databases that you intend to use.

   For more information, see "Character Encoding That Siebel CRM Supports to Import or Export Repositories" on page 67.

2. In Microsoft Windows, navigate to Start Programs menu, Settings, Control Panel, and then click Services.
3. Stop all Siebel Servers.

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

4. Click the Start Programs menu, Programs, Siebel Enterprise Server Configuration 8.0, and then click Database Server Configuration.
5. In the Database Configuration Wizard, enter information when this wizard prompts you, and then click Next to continue.
6. Choose Import Repository when the wizard prompts you to specify a database operation.

   For important caution information, see "Caution About Migrating a Repository" on page 77.

7. Specify the following items:

- To import your custom 8.x repository
- The location of where the custom CustRep.dat file resides

8 When the wizard displays the Configuration is Complete window, choose one of the following options, and then click Next:

- **Yes Apply Configuration Changes Now.** The wizard saves the configuration information that you entered, and you can open the Siebel Upgrade Wizard in Step 10.
- **No I Will Apply Configuration Changes Later.** The wizard saves the configuration information that you entered, but you cannot open the Siebel Upgrade Wizard in Step 10.

9 In the Configuration Parameter Review window, review the configuration that you entered. To modify a value, click Back to return to the window that includes the parameter you must modify, modify the parameter, and then click Next.

10 When the wizard prompts you to run the configuration, click one of the following:

- **No.** The wizard does not save the configuration information that you entered. You must enter the database configuration parameters again.
- **Yes.** The wizard saves the configuration information that you entered.

11 Click OK.

The Database Configuration Wizard does one of the following, depending on the choice that you make in Step 8:

- **Apply now.** Opens the Siebel Upgrade Wizard, and then calls the SQL generator to create the SQL scripts.
- **Apply later.** Saves the configuration information, but does not open the Siebel Upgrade Wizard. You can restart the configuration, and then run the Upgrade Wizard later. For more information, see "Importing or Exporting Repositories at a Later Time" on page 71.

Importing a Repository in UNIX Environments

This topic describes how to import a repository in a UNIX environment.

To import a repository in UNIX environments

1 Do Step 1 on page 68 and Step 3 on page 68.

2 Make sure $SIEBEL_ROOT is the current folder.

3 Run a script, depending on the following UNIX shell that you are using:

- **Korn shell.** Run siebenv.sh.
- **C shell.** Run siebenv.csh.

4 Make sure that the following environment variables use the following values:

- **SIEBEL_ROOT.** This path must end in siebsrvr. For example, /usr/siebel/siebsrvr.

- **LANGUAGE.** Determines the language that the Database Configuration Wizard uses. The value of this variable is a text string that identifies the language. For example, enu for English.

5 Start the Database Configuration Wizard. You run the following command:

```
$SIEBEL_ROOT/bin/ssincfgw -args MODEL_FILE=$SIEBEL_ROOT/admin/dbsrvr.scm
MODE=LIVE
```

6 In the Database Configuration Wizard, do the following:

- Enter information when the wizard prompts you. Use the Next and Back button to navigate between dialog boxes.
- Choose Import Repository when the wizard prompts you to choose a database operation.
- Specify to import your 8.x repository.
- Identify the location of where the custom CustRep.dat file resides.

7 After you enter all the requested information, the wizard displays a message that is similar to the following. Click Next to continue:

```
Configuration is complete: configuration parameters will be saved to $Masterfile
file when the wizard completes. Please run the following command line after you
exit from this configuration wizard. This command will deploy the process you
configured to the database.

$SIEBEL_ROOT/siebsrvr/bin/srvrupgwiz /m $SIEBEL_ROOT/siebsrvr/bin/$Masterfile
```

8 The wizard displays the values that you entered in the Parameter Review window. To modify a value, click Back to return to the appropriate window.

9 The wizard prompts you to click one of the following values:

- **Yes**. The wizard saves the configuration in a master file in the $SIEBEL_ROOT/bin folder. It does not start the Upgrade Wizard. For information about how to start the Upgrade Wizard, see *Siebel Database Upgrade Guide*.
- **No.** The wizard does not save the configuration that you entered.

Exporting Repositories

To export a repository when you use:

- **Microsoft Windows.** Use the same procedure that you use to import a repository, but choose Export Repository in Step 6 on page 68.
- **UNIX.** Use the same procedure that you use to import a repository, but choose Export Repository in Step 6 on page 70.

Importing or Exporting Repositories at a Later Time

If you use the Database Configuration Wizard to export or import a repository, then it saves the values that you specify to the master_exprep.ucf file that resides in the *SIEBSRVR_ROOT*\ folder. After the wizard finishes collecting information from you, it prompts you to export or to not export. If you choose not to export or not to import, then you can run the following command to do the export or import at a later time:

```
siebupg.exe /m master_exprep.ucfs
```

Upgrading Repositories

The *Siebel Application Upgrader* is a utility that you can use to get new features from the latest software release while preserving the custom configuration that you created in the current repository. It allows you to do the following:

- Compare your custom configuration to the modifications that a new Siebel CRM release contains.
- Receive notifications about conflicts between the customizations that you make and the new release.
- Merge any differences between objects.
- Choose the modifications to apply and manually override modifications.
- Merge objects with versions, including task UI objects and workflow processes. It copies version 1 through version *n* from the prior custom repository to the new custom repository. It merges version 0 from the prior repository with the new custom repository. This configuration results in version *n* + 1 in the new custom repository.
- Reduce the time required to upgrade Siebel CRM.

You can use the Application Upgrader to merge an entire custom repository with another repository. To merge only part of a repository, you must import the repository. For more information, see "Exporting and Importing Repositories" on page 66. For more information about the Application Upgrader, see *Siebel Database Upgrade Guide*.

To upgrade repositories

1. Log in to Siebel Tools.
2. Click the Tools menu, Upgrade, and then click Upgrade Application.
3. In the Merge Repositories dialog box, choose the repositories to merge, and then click Merge.

   Siebel Tools does the following:

   - Starts the upgrade.
   - Displays any differences between objects and properties.
   - Displays any differences between objects and properties for different versions of task UIs and workflow processes.

Identifying Conflicts That Occur During Upgrades

This topic describes how to identify merge conflicts between objects that Siebel Tools added or modified during a repository merge. A *merge conflict* is a scenario in which a customer changed an object in the current customer repository, and Siebel CRM changed that same object to a different value in a new version of the Siebel repository. Consequently, the attribute for the object is different when the following three repositories are compared: the current customer repository, the current Siebel repository, and the new version of the Siebel repository.

To identify conflicts that occur during upgrades

1. Make sure you finish upgrading repositories.

   For more information, see "Upgrading Repositories" on page 71.

2. In Siebel Tools, click the Screens menu, Application Upgrader, and then click Application Upgrade Object List.

3. In the Application Upgrades list, right-click the record of the merge that you want to analyze, and then click Hierarchy Reports.

   Siebel Tools displays the hierarchy report, which includes the objects that Siebel Tools added or modified during a merge. The report includes the following types of objects:

   - **Objects that include a valid value for each field value.** Siebel Tools modified these objects during the merge. An N or a Y in a binary field is an example of a valid value.

     For these objects, the Status field designates added objects or objects with modified attributes. For added objects, an N appears in the Attributes field because none of the attributes are modified. For objects with modified attributes, a Y appears in the Attributes field, and the modified attributes appear in the Attributes pane.

   - **Objects that include an asterisk (*) for each field value.** Siebel Tools modified children of these objects during the merge, or Siebel Tools modified dependent objects of these objects during the merge.

4. To view information about merge details, complete the following steps:

   a. Expand and navigate through the hierarchical tree in the Object Types pane to select a parent or child object type.

      The top List of Objects pane shows the objects for the parent object type that you select, and the bottom List of Objects pane shows the objects for the child object type that you select.

   b. Click an object in a List of Objects pane.

      In the Attributes pane, Siebel Tools displays any merge conflicts for the object.

5. (Optional) To filter the objects in the report, select one of the following values in the Filter drop-down list, and then click Go:

   - **Siebel and Customer Modified.** Siebel Tools displays only objects that Oracle or you modified. These objects have a Y in the In Prior Standard, In Prior Customized, and In New Standard fields.

   - **Siebel Modified.** Siebel Tools displays only objects that Oracle modified or added. These objects have an N in the In Prior Standard and In Prior Customized field and a Y in the In New Standard field.
   - **All.** Siebel Tools displays all modified or added objects.

6. (Optional) To filter the merge conflicts in the Attributes pane, select or clear the Critical Only check box as follows, and then click Go:
   - Select the check box to show only critical merge conflicts.
   - Clear the check box to show all merge conflicts.
7. (Optional) To display the dependencies for an object, complete the following steps:
   a. Click an object in a List of Objects pane.

      If you want to display the dependencies for an additional object, then hold down the CTRL key, and click the additional object in the List of Objects pane.

   b. Click Show Dependencies.

      If the object has no dependencies, then the Show Dependencies button is disabled.

      Siebel Tools displays the dependent objects that it modified and the attributes for these objects.

   c. Click Back to return to the hierarchy report.

Process of Migrating Repositories

To migrate a repository, do the following tasks:

1. Preparing to Migrate Repositories on page 73
2. Using the Database Configuration Wizard to Migrate Repositories on page 75
3. (Optional) Updating Siebel Remote Databases on page 78
4. Migrating Non-repository Configurations and Data on page 78

This topic also includes the following information:

- Using the Repository Import and Export Utility on page 80
- Guidelines for Migrating Repositories on page 82

It is sometimes necessary to migrate the Siebel Repository and any application customizations that between databases before you deploy modifications to a development, test, or production environment. Doing this migration makes sure that the database schema for the user data, the business objects, and the user interface remain synchronized.

Preparing to Migrate Repositories

In this topic, you prepare to migrate the repository.

This task is a step in "Process of Migrating Repositories" on page 73.

To prepare to migrate repositories

1. Make sure you complete testing.

   You must verify that any customizations you make work correctly, and that they meet your business requirements.

2. Make sure the target database configuration meets the database requirements.

   For more information, see the *Siebel Installation Guide* for the operating system you are using and *Siebel System Requirements and Supported Platforms* on Oracle Technology Network.

   NOTE: For Siebel CRM product releases 8.1.1.9 and later and for 8.2.2.2 and later, the system requirements and supported platform certifications are available from the Certification tab on My Oracle Support. For information about the Certification application, see article 1492194.1 (Article ID) on My Oracle Support.

3. Check in all projects in the source database and in the target database.

   If a project is checked out, and if you migrate a database, then the migration works but Siebel CRM does not lock the project in the target database. For more information, see "Checking Out and Checking In Projects and Objects" on page 85.

4. If your environment includes Remote users, then make sure all these users do a full synchronization.

5. Do a full backup of the target database.

6. Delete old repositories from the target database.

   For more information, see "Deleting Repositories" on page 83.

7. Update database statistics, if necessary.

8. Export the source repository to a file.

   It is recommended that you use the Database Configuration Wizard to export the source repository to a file. You use this file later during a migration. For more information, see "Exporting Repositories" on page 70.

   You can also migrate directly from the source database. If you do this, then you do not need to export the source repository to a file.

9. Stop all Siebel Server tasks and disconnect all database access.

   If you do not need to minimize downtime during a migration, then it is recommended that you stop all Siebel Server tasks and disconnect all database access until the migration finishes. This downtime occurs in the Siebel Enterprise where the target repository resides.

Using the Database Configuration Wizard to Migrate Repositories

In this topic, you use the Database Configuration Wizard to migrate repositories.

This task is a step in "Process of Migrating Repositories" on page 73.

To use the Database Configuration Wizard to migrate repositories

1. Make sure all users, including the database administrator, disconnect from the Siebel application.
2. Start the Database Configuration Wizard according to the following operating system that you are using:
   - **Windows.** Do Step 1 on page 68 through Step 5 on page 68.
   - **UNIX.** Do Step 1 on page 69 through Step 5 on page 70.
3. In the Database Configuration Wizard, click Migrate Repository when the wizard prompts you to choose a database operation, and then click Next.

   For important caution information, see "Caution About Migrating a Repository" on page 77.
4. Choose to get the source repository from one of the following items, and then click Next:
   - **Get from a file.** If you choose this option, then the wizard displays the Repository File Selection window. Enter the path and the file name of the source repository file that you created in Step 8 on page 74, and then click Next.
   - **Get from the database**.
5. Specify whether or not the target enterprise is online or offline, and then click Next:
   - Online.
   - Offline.

   If you choose the online option, then Siebel CRM keeps the Siebel Enterprise where the target repository resides online during the migration until it must shut down this environment. When it must do this shut down, it displays a dialog box that informs you that it will shut down the target environment during the remainder of the migration. This option allows you to minimize the amount of downtime during a migration. If you run this migration unattended, then do not choose this online option. Running this migration from a script is an example of running it unattended.
6. Choose one of the options, and then click Next:
   - **There are new schema changes to be applied.** Choose this option if the source repository includes any physical or logical schema modifications. Examples of schema modifications include adding an extension column to an existing table, or adding a new extension table. If you are not sure whether or not any schema modifications exist, then it is recommended that you choose this option.
   - **There are no new schema changes.** Choose this option if the source repository does not include any physical or logical schema modifications. If you choose this option, then the Database Configuration Wizard requires less time to finish the migration.

7 If in Step 4 on page 75 you choose to get the source repository:

- **From a file.** The wizard displays the Target Repository Name window. Enter the target repository name, and then click Next.
- **From the database.** The wizard displays the following series of dialog boxes that require you to define the source database. Complete each of these dialog boxes. Use information from the following table.

Dialog Box	Description
ODBC Data Source Name	Enter the ODBC data source name.
Database User Name	Enter the user name and password for the source database.
Database Table Owner	Enter the table owner and table owner password for the source database.
Source Database Repository Name	Enter the source database repository name and the target database repository name.

8 Complete the information in the following windows, and then click Next.

The sequence that the wizard uses to display these windows and the information that you enter might vary slightly depending on whether or not your environment uses Windows or UNIX, or the type of database that your environment uses. The wizard does not display any windows for Microsoft SQL Server.

Window	Description
RDBMS Platform	For UNIX only. Choose an RDBMS platform.
Target Database Encoding	Choose whether or not the target database is Unicode. Consult with your DBA regarding how to set up the target database.
Target Database Repository Name	Enter the repository name of the target database.
Target RDBMS Platform	Choose the target RDBMS platform.
Target Database ODBC Datasource	Enter the ODBC data source for the target database.
Target Database User Name	Enter the user name and password that the server components that reside on the target database use.
Target Database Table Owner	Enter the table owner and the table owner password for the target database.
Index Table Space Name	If you choose IBM DB2 for the target database platform, then the wizard displays the Index Table Space Name dialog box. Enter the index tablespace name and 4KB tablespace name.

Window	Description
16K Table Space Name	If you choose IBM DB2 for the target database platform, then the wizard displays the 16K Table Space Name dialog box. Enter the 16KB tablespace name and the 32KB tablespace name.
Index Table Space Name	If you choose Oracle Database for the target database platform, then the wizard displays the Index Table Space Name dialog box.

9 If the wizard displays a Language Selection window, then choose the appropriate language, and then click Next.

The wizard migrates strings from the repository that uses the language that you choose. If you migrate a repository in a multilingual environment, and if the source repository includes duplicate strings in multiple languages, then the wizard copies all of these strings to the target repository.

10 Finish the migration, depending on the following operating system that your environment uses:

- **Windows.** Do Step 8 on page 69 through Step 11 on page 69.
- **UNIX.** Do Step 7 on page 70 through Step 9 on page 70.

11 Wait for the wizard to finish the migration.

The Database Configuration Wizard does the following work:

- Exports the source repository from a file. You can also export the repository for the database.
- Imports the repository into the target database.
- If you choose to apply new schema modifications in Step 6, then the wizard does the following work:
  - Exports the logical schema definition from the source repository to a control file.
  - Synchronizes the physical schema of the target database with this logical schema definition.

Caution About Migrating a Repository

If you migrate a custom repository, then you must use the Migrate Repository option in the Database Configuration Wizard.

CAUTION: If you migrate a custom repository and schema from one environment to another environment, such as migrating from a development environment to a test environment, then do not use the Export/Import option in the Database Configuration Wizard. Instead, you must use the Migrate Repository option.

This Migrate Repository option calls the Repository Migration Utility (dev2prod). If you use the Repository Migration Utility directly instead of calling it from the Database Configuration Wizard, then you must reset the Locked and Allow Object Locking columns after you finish the migration. For more information, see "Allowing Object Locking for a Project" on page 89.

Updating Siebel Remote Databases

If your environment includes Remote users, then you must update Siebel Remote databases. For more information about performing the steps in this topic, see *Siebel Remote and Replication Manager Administration Guide*.

This task is a step in "Process of Migrating Repositories" on page 73.

To update Siebel Remote databases

1. Stop and restart Siebel Remote server components.
2. Wait for the Transaction Processor (TxnProc) and the Transaction Router (TxnRoute) server components to finish processing all pending transactions.
3. Regenerate the local database templates.

   Use the Generate New Database server component (GenNewDb) to regenerate the local database template file. This step updates this template file so that it uses the same schema and version that the server database uses.
4. Reextract the local database for all Remote users.

   If you do not use Siebel Anywhere to update Remote clients, then you must use the Database Extract (DbXtract) server component to extract the local database for all remote users.

   If you do not extract these databases, then Remote users can synchronize but Siebel CRM does not create any error message even though these databases might not include the migration modifications. This configuration allows users who use Siebel Anywhere to upgrade their Remote databases and continue working.

Using Siebel Anywhere to Update Remote Databases

You can use Siebel Anywhere to update some Remote databases. Importing a repository and then synchronizing the schema definition in a target environment is equivalent to migrating the repository. The Remote users must synchronize the next time they log in to their local database after the migration finishes. Synchronizing downloads new schema modifications from the Siebel Server to the local database that the Remote user uses. Before synchronization occurs, you must rename the older repository to a temporary name, and rename the imported repository to the correct name. If a Remote user does not synchronize, then the data that the local database contains does not match the data that the server database contains. For more information, see *Siebel Anywhere Administration Guide*.

Migrating Non-repository Configurations and Data

The Database Configuration Wizard does not migrate non-repository configurations and data. This topic describes how to migrate this information.

This task is a step in "Process of Migrating Repositories" on page 73.

To migrate non-repository configurations and data

1. Migrate Web templates, image files, and style sheets.

   For more information, see "Migrating Web Templates, Image Files, and Style Sheets" on page 79.

2. Migrate any customizations that you have made to the Siebel application or to runtime data.

   You must migrate any customizations that your Siebel Enterprise data contains, such as custom views, responsibilities, assignment rules, and so on. For more information, see *Siebel Application Deployment Manager Guide* and *Siebel Assignment Manager Administration Guide*.

3. Migrate custom configurations, such as parameter values and server component definitions.

   For more information, see "Migrating Custom Configurations" on page 80.

4. Migrate database triggers.

   Siebel CRM does not migrate custom database triggers. If your Siebel base tables include custom database triggers, then you must disable them before you migrate the repository, and then recreate them on the target repository after the migration finishes.

5. Update database statistics.

Migrating Web Templates, Image Files, and Style Sheets

A Siebel Server installation includes the following files that determine how Siebel CRM displays the Siebel client in a Web browser:

- Siebel Web template files (SWT files)
- Image files (GIF, JPG, and PNG files)
- Cascading style sheet files (CSS files)

If you modify any of these files in the source environment, then you must copy them to the target environment. Note the following:

- You can use Siebel Application Deployment Manager (ADM) to automate this work. For more information, see *Siebel Application Deployment Manager Guide*.
- For information about how to specify the Siebel Enterprise Security Token and how to migrate content from the Siebel Server to the Web server, see the *Siebel Installation Guide* for the operating system you are using and *Siebel Security Guide*.

To migrate Web templates, image files, and style sheets

- Copy any new or modified files from the source Siebel Server to the target Siebel Server. Use values from the following table to determine where to copy files.

File Type	File Location
Web templates	*SIEBSRVR_ROOT*\WEBTEMPL
Images	*SIEBSRVR_ROOT*\webmaster\images\ *language_code*
Cascading style sheets	*SIEBSRVR_ROOT*\webmaster\files\ *language_code*

Migrating Custom Configurations

If you migrate a repository, then you must also migrate any custom environment configurations in the source environment to the target environment. For example, custom server component definitions. Siebel CRM stores this information in the siebns.dat file on the Siebel Gateway Name Server. The Database Configuration Wizard does not migrate this information.

To migrate custom configurations

1. Recreate any custom environment configurations.

   Use the Server Manager command-line interface (srvrmgr) and scripts that provide input to srvrmgr that automate this work in the target environment. It is recommended that you use scripts because you can save them and reuse them for future migrations. If you reuse a script, then make sure you edit it if the target environment is a new Siebel version. For example, if a Siebel upgrade includes a new Siebel version. New parameters or srvrmgr commands might be available with this new version. For more information, see *Siebel System Administration Guide*.

   You can also use the Server Manager GUI or command-line interface (srvrmgr) to manually create these custom configurations in the target environment. For information, see *Siebel System Administration Guide*.

2. Recreate parameter values.

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

Using the Repository Import and Export Utility

The Repository Import and Export Utility imports or exports a repository, or creates a file dump of a repository. It can also import an INTL table. INTL tables contain language-specific information and are a part of the repository. You must use this utility only if your migration requires special parameters settings that a batch file cannot access, or if you must create a file dump. In all other situations, you must use the Database Configuration Wizard. For information "Using the Database Configuration Wizard to Migrate Repositories" on page 75.

To use the Repository Import and Export Utility

1. Open a Windows command line.
2. To import a repository, enter the following command:

   ```
   repimexp /A I /G language_codes
   ```

   where:

   - A I specifies to import data from a file.
   - *language_codes* is a list of codes.

3. To export a repository, enter the following command:

```
repimexp /A E parameters
```

where:

- A E specifies to export data to a file.
- *parameters* specify the parameters.

4 To display a complete list of command-line usage options for this utility, enter the following command:

```
repimexp
```

You can save this output to a text file.

For more information, see "Parameters You Can Use with the Repository Import and Export Utility" on page 81.

Parameters You Can Use with the Repository Import and Export Utility

Table 8 describes the parameters that you can use with the Repository Import and Export Utility.

Table 8. Parameters You Can Use with the Repository Import and Export Utility

Parameter	Description
You can use one of the following values: ■ `/a e` ■ `/a i` ■ `/a x`	Required. You can use one of the following parameters: ■ **a e.** Specifies to export all data from the Siebel Repository tables to a file. ■ **a i.** Specifies to import data from a file into the Siebel Repository tables. ■ **a x.** Specifies to import locale-specific repository data from a file into the INTL tables. If you use a x, then you must also use the /g option.
`/u` *`user_name`*	Required. Specifies the Siebel administrator user name.
`/p` *`password`*	Required. Specifies the password for the Siebel administrator user name.
`/c` *`ODBC_data_source`*	Required. Specifies the ODBC data source. The value that resides in the SIEBEL_DATA_SOURCE environment variable is the default value. If this value includes a space, then you must enclose the value in quotes.
`/d` *`table_owner`*	Required. Specifies the Siebel Database table owner. The value of the SIEBEL_TABLE_OWNER environment variable is the default value.

Table 8. Parameters You Can Use with the Repository Import and Export Utility

Parameter	Description
`/g` *`language_codes`*	For import only. Required only if you use the /a x parameter to import to INTL tables. Specifies to import a repository that contains locale-specific data for one or more languages. For example: FRA, DEU, ITA. You can specify ALL to import all the languages that the repository file contains. To import locale objects, you must use the /g parameter, and you must specify at least one language code. If you do not do this, then the utility does not import any locale objects, and the Siebel client does not include any text after you compile the imported repository. If you export repository data to a file, then this utility includes locale-specific data for all languages.
`/r` *`repository_name`*	Required. Specifies the name of the repository that you are importing or exporting. The default value is Siebel Repository. If this value includes a space, then you must enclose the value in quotes.
`/f` *`data_file`*	Required. Specifies the data file for the repository, including the path to this file. Specify the file to which you are exporting, or from which you are importing, repository data. The path must not include spaces.
You can use one of the following values: ■ /v y ■ /v n	Specifies to verify data: ■ The default value for export is n. ■ The default value for import is y.
You can use one of the following values: ■ `/n 0` ■ `/n 1` ■ `/n 2`	For export only. Specifies to modify information about creating and updating records. You can use one of the following values: ■ **n 0.** Make no modifications. ■ **n 1.** Update the CREATED_BY, UPDATED_BY, and OWNER_BRANCH columns. The default value is 1. ■ **n 2.** Update the CREATED_BY, UPDATED_BY, and OWNER_BRANCH columns, and update the date columns.

Guidelines for Migrating Repositories

If you migrate a repository, then it is recommended that you use the following guidelines:

- Isolate development and test environments from the production environment.
- Do not migrate a repository between two databases that use different releases or patch levels. To avoid an inconsistent environment, do not migrate repositories between different versions of Siebel applications.
- If your migration requires special parameter settings that batch files cannot access, or if you must do a file dump, then use the Repository Import and Export Utility. For more information, see "Using the Repository Import and Export Utility" on page 80.
- Use a consistent naming convention for the source repository and for the target repository. You can rename the target repository to indicate that you replaced it during a migration. For more information, see "Guidelines for Naming Repositories" on page 51.
- Do not modify ODBC parameters or settings for the data sources that you created when you configured the Siebel Server when you installed Siebel CRM. A repository migration references one ODBC data source for the source database, and another ODBC data source for the target database. This requirement applies to all servers and databases. For more information about how to verify the ODBC data source, see the *Siebel Installation Guide* for the operating system you are using.
- Consider the time required to export over a WAN. It is more efficient to export the file on the source environment, and then copy the exported file to the target environment. If you migrate a repository over a wide area network (WAN), and if you run the Database Configuration Wizard from the target environment, then Siebel CRM exports only the source repository to a flat file. All other processing occurs on the local area network (LAN) of the target environment.
- If you defined a custom tablespace on IBM DB2 for z/OS, then this tablespace might affect the Database Configuration Wizard. For more information, see *Implementing Siebel Business Applications on DB2 for z/OS*.

Deleting Repositories

This topic describes how to delete a repository.

To delete repositories

1. Back up the repository that you plan to delete.

   If you delete a repository, then Siebel CRM removes all records that the repository contains.
2. In Siebel Tools, display the Repository object type.

   For more information, see "Displaying Object Types in the Object Explorer" on page 26.
3. In the Object Explorer, click Repository.
4. In the Repositories list, click anywhere in the row for the repository that you must delete.

5. Click the Edit menu, and then click Delete Record.

   CAUTION: To improve performance, it is recommended that you delete obsolete repositories. Deleting a repository requires a significant amount of time and requires resources, such as rollback segments, cursors, tablespace, and so on. Consult your DBA before you delete a repository.

6. Click outside the record to save the deletion.

4 Checking Out and Checking In Projects and Objects

This chapter describes how to check out and check in projects and objects. It includes the following topics:

- Overview of Checkout and Checkin on page 85
- Getting Projects from the Server Repository on page 86
- Using Checkout and Checkin on page 89
- Setting Options for Checkout and Checkin on page 96
- Customizing Projects on page 101

Overview of Checkout and Checkin

You can use checkout and checkin to do the following:

- Allow multiple developers to work on objects in a single project.
- Improve check-out and check-in times.
- Reduce network traffic.

You can check out and check in projects or objects to help make sure only one developer works on an object at one time. You can check out objects from the server repository and download them to your local repository for editing. If you check out an object, then Siebel Tools locks the object in the server Repository and no other developer can check it out. This configuration helps to avoid conflicts that result if more than one developer works on the same object at the same time. If you check in an object, then Siebel Tools removes the lock and other developers can check it out.

About the Project Object Type

A *project* is an object type that your development team can use to help make sure only one developer works on an object at one time. A set of objects can reference a project according to a functional area. Every object in a repository references a project. You can check out and check in only the objects that you need. You are not required to check out and check in entire projects.

About Project Names

The project name describes the functional area of the objects that reference this project. For example, the Account project includes objects that Siebel CRM uses to manage accounts. A suffix at the end of the project name includes more information:

- **Includes a suffix.** The project includes objects that are specific to the Siebel application that the suffix describes. For example, the SSE suffix of the Account (SSE) project indicates that this project includes account objects that Siebel CRM uses for the Siebel Sales application (**S**iebel **S**al**E**s). The Account (SSV) project includes objects that Siebel CRM uses for Siebel Service (**S**iebel **S**er**V**ice).
- **Does not include a suffix.** The project includes objects that multiple Siebel applications use. For example, the Account project.

Guidelines for Using Checkout and Checkin

If you check out or check in a project or an object, then it is recommended that you use the following guidelines:

- To check out a project, you must disable password encryption in the client or in the CFG file when you run Siebel Tools. Password encryption interferes with checkout.
- You check out projects and objects only in the language mode that Siebel Tools uses. For more information, see "Enabling Language Override" on page 20.
- The sample database cannot receive checked out projects or objects from the server repository. You cannot check in projects or objects from the sample database to the server repository. You can use the sample database only for instructional use.
- You can check out and check in objects only with the server repository that you use to extract the local repository.
- Before doing a checkin, you must make sure that the projects and objects that you are checking in are stable, that all scripting is complete, and that you successfully tested the configuration in your local repository.
- To make sure that the configuration in the server repository remains consistent, it is recommended that you check in all projects and objects that you modify at the same time. For example, if you create a new picklist that references the Picklist project, and if you configure Siebel CRM to reference this object in the Oppty project, then check these projects into the server repository at the same time.
- Consider how the timing of your checkin affects other developers.

  CAUTION: Depending on the project size, a checkin might require significant time to finish. Do not interrupt a checkin. If you interrupt a checkin, then the repository might become unstable. If for any reason a checkin is interrupted, you must do the checkin again until it finishes successfully.

Getting Projects from the Server Repository

This topic describes how to get projects from the server repository. It includes the following information:

- Getting All Projects from the Server Repository on page 87

- Getting Some Projects from the Server Repository on page 88
- Getting Locale Data from the Server Repository on page 89

A *Get* is the act of copying a project from the server repository to your local repository. The Get is different from checking out in the following ways:

- Getting a project does not lock this project in the server repository.
- Getting projects overrides all locked and unlocked projects on your local repository.

To start the executable that performs a Get, the following situations must exist:

- The user must be the same user who installed Siebel Tools on the local computer.
- You must set the ODBC driver that Siebel Tools uses to perform the Get to System DSN instead of User DSN. This configuration allows any user to do the Get.

Getting All Projects from the Server Repository

A *Full Get* is a type of Get that copies all projects from the server repository to your local repository. It is useful the first time you initialize the local repository, or if you must replace the projects that your local repository contains. You can use a Full Get to synchronize the local repository with modifications that reside in the server repository. You must do a Full Get before you compile the repository because the repository must use the full set of Siebel objects. For more information, see "Compiling Your Modifications" on page 179.

This task is a step in "Roadmap for Setting Up and Using Siebel Tools" on page 14.

To get all projects from the server repository

1. Log in to Siebel Tools, connected to the local repository.
2. Click the Tools menu, and then click Check Out.
3. In the Repository drop-down list, click the name of your development repository.

   You need to do this step only one time. The repository that you choose is not necessarily the same repository that Siebel Tools opens.
4. Click All Projects.
5. Click Options.
6. In the Development Tools Options dialog box, make sure the following items are set:
   - **Server Data Source.** Must reference the server development repository.
   - **Client Data Source.** Must reference the local repository that you already initialized, and that you connect to when you open Siebel Tools.

   For more information, see "Specifying the Data Source That Siebel Tools Uses" on page 15.
7. (Optional) Modify how Siebel Tools performs database commits.

   For more information, see "Configuring Siebel Tools to Do Database Commits at the End of a Full Get" on page 88.

8 In the Check Out dialog box, click Get.

Siebel Tools copies all objects from the server repository to your local repository.

Configuring Siebel Tools to Do Database Commits at the End of a Full Get

This topic describes how to configure Siebel Tools so that it performs database commits only at the end of a Full Get. A Full Get performs database commits in regular intervals during the Get operation rather than during a single commit at the end of the Get, by default. A Full Get can provide better performance, but if an error occurs, then you must do the Full Get again.

To configure Siebel Tools to do database commits at the end of the Get

1. In Siebel Tools, click the View menu, and then click Options.
2. In the Development Tools Options dialog box, click the Check In/Out tab.
3. Make sure the Enable Incremental Commit During Full Get check box does not contain a check mark.
4. Click OK.

Getting Some Projects from the Server Repository

You can use a Get to overwrite projects that reside in your local repository with projects from the server repository. Getting some projects from the server repository is useful in the following situations:

- You modified local copies of projects, and you must revert back to the versions that reside in the server repository.
- Other developers checked their modifications into the server repository, and you must copy these modifications to your local repository.

To get some projects from the server repository

- Do all the steps described in "Getting All Projects from the Server Repository" on page 87 with the following difference:
  - In Step 4 on page 87, in the Projects list, choose the projects that you must get.

  Siebel Tools copies only objects that reference the projects that you choose. It copies these objects from the server repository to your local repository.

Getting Locale Data from the Server Repository

After you do a Full Get, you can get locale data without also getting parent objects. This configuration is useful if you work in one language, and then switch to another language. For example, assume you populate your local repository with English (ENU) data, but you must switch to Japanese (JPN). After you switch the language mode to JPN, you can use the Get Locale-Specific Data option to copy only JPN records from the server repository to your local repository.

To get only locale data for projects

- Do all the steps described in "Getting All Projects from the Server Repository" on page 87 with the following difference:
  - In Step 4 on page 87, in the Projects list, choose only the projects that require locale data.
  - In Step 8 on page 88, make sure the Get Locale Specific Data Only check box includes a check mark.

  Siebel Tools copies only the data that it stores in the locale objects that reference the projects that you choose.

Using Checkout and Checkin

This topic describes how to use checkout and checkin. It includes the following information:

- Allowing Object Locking for a Project on page 89
- Checking Out Projects from the Server Repository on page 90
- Checking Out Objects from the Server Repository on page 91
- Checking In Projects or Objects to the Server Repository on page 92
- Locking a Project in the Local Repository on page 93
- Locking an Object in the Local Repository on page 93
- Unlocking a Project on page 94
- Undoing Project Checkout on page 94
- Preventing Object Checkin and Checkout on page 95
- Viewing Locked Objects in a Project on page 95
- Viewing Differences Between Local Repositories and Server Repositories on page 95

Allowing Object Locking for a Project

If you set the Allow Object Locking property of a project to TRUE, then the following situations apply:

- You cannot check out this project from the server repository. To check out an entire project from the server repository, you must set the Allow Object Locking property to FALSE.

- You can check out and check in an object that references this project.

If you set the Allow Object Locking property of a project to TRUE, and if another developer already checked out an object that references this project, then you cannot do the following work on this object:

- Delete the object.
- Rename the object.
- Assign the object to a different project.

You cannot set the Allow Object Locking property for a project that resides in your local repository.

This task is a step in "Roadmap for Setting Up and Using Siebel Tools" on page 14.

To allow object locking for a project

1. Log in to Siebel Tools with the SADMIN user Id, connected to the server repository.

   To modify the Allow Object Locking property, you must use the SADMIN user Id, and Siebel Tools must connect to the server repository. For more information, see "Specifying the Data Source That Siebel Tools Uses" on page 15.

2. In the Object Explorer, click Project.
3. In the Projects list, locate a project.

   For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

4. Right-click the project that you located in Step 3, and then click Toggle Allow Object Locking.

   If an object that references this project is locked in the server repository, then Siebel Tools sends an error message to the system administrator. Siebel Tools does not display the Toggle Allow Object Locking option in the following situations:

   - Someone else already locked this project in the server repository.
   - The EnableObjectCOCI parameter in the Siebel Tools configuration file is set to FALSE.
   - You did not log in to Siebel Tools with the SADMIN user Id.
   - Siebel Tools is not connected to the server repository.

Checking Out Projects from the Server Repository

This topic describes how to check out projects from the server repository.

This task is a step in "Roadmap for Setting Up and Using Siebel Tools" on page 14.

To check out projects from the server repository

1. To enable object checkout, make sure the following parameter in the Siebel section of the tools.cfg file is set to TRUE:

```
EnableObjectCOCI=TRUE
```

If this parameter does not exist in the Siebel section, then add it now. For more information, see "About the Configuration Files" on page 17.

2 Log in to Siebel Tools, connected to your local repository.

3 Make sure the project allows object locking.

For more information, see "Allowing Object Locking for a Project" on page 89.

4 Click the Tools menu, and then click Check Out.

5 In the Project Check Out dialog box, make sure you choose the correct repository.

For more information, see "Elements of the Project Check Out Dialog Box" on page 231.

6 Click the projects that you must check out, and then click Options.

7 In the Development Tools Options dialog box, make sure you specify the Server and Client data sources correctly, and then click OK.

For more information, see "Specifying the Data Source That Siebel Tools Uses" on page 15.

8 In the Check Out dialog box, click Check Out.

Siebel Tools does the following:

- Locks all objects in the server repository that reference the project. This lock prevents another developer from checking out these objects.
- Copies all the objects that it locked. It copies these objects from the server repository to your local repository. These objects remain locked on your local repository. You can edit them.

Checking Out Objects from the Server Repository

This topic describes how to check out an object from the server repository. You can check out only a top-level object. For information about top-level object types, see "Displaying Object Types in the Object Explorer" on page 26.

To check out objects from the server repository

1 Do Step 1 on page 90 through Step 3 on page 91.

2 In the Object List Editor, right-click the object that you must check out, and then click Check Out.

Siebel tools displays the Check Out Object dialog box. If another developer already checked out the object, or if the project that this object references does not allow object locking, then Siebel Tools disables the Check Out button that it shows in the Check Out Object dialog box. For more information, see "Allowing Object Locking for a Project" on page 89.

3 In the Check Out Object dialog box, choose the objects to check out.

For more information, see "Elements of the Object Check Out Dialog Box" on page 234.

4 Click Check Out.

Siebel Tools does the following work:

- Locks the object and all child objects of this object in the server repository. This lock prevents another developer from checking out the object.
- Copies the object that it locked from the server repository to your local repository. This object remains locked on your local repository. You can edit it.

Checking In Projects or Objects to the Server Repository

This topic describes how to check projects or objects into the server repository.

This task is a step in "Roadmap for Setting Up and Using Siebel Tools" on page 14.

To check in projects or objects to the server repository

1 Make sure the EnableObjectCOCI parameter is set to TRUE.

For more information, see Step 1 on page 90.

2 Click the Tools menu, and then click Check In.

3 In the Check In dialog box, make sure you choose the correct repository.

For more information, see "Elements of the Check In Dialog Box" on page 235.

4 Click Options.

5 In the Development Tools Options dialog box, make sure you set the Siebel Server and the client Data Sources correctly, and then click OK.

For more information, see "Specifying the Data Source That Siebel Tools Uses" on page 15.

6 Do one of the following:

- To check in some projects or objects, click Selected Objects, and then choose the projects and objects that you must check in.
- To check in all locked projects and objects, click Locked/New Objects.

7 Click Check In.

If an object references the project, then Siebel Tools does the following work:

- Unlocks this object in the local repository. Removing this lock allows another developer to check out the object. Siebel Tools removes the lock for each object that references the project.
- Copies this object from the local repository to the server repository. Siebel Tools copies each object that references the project, including any new objects that you added.

Locking a Project in the Local Repository

You can lock a project that resides in the local repository without checking it out from the server repository. This is useful in the following situations:

- You must test a configuration on your local computer, but you must not prevent other developers from checking out this project from the server repository.
- You discard your work when you are done. In this situation, it is not necessary for you to check in an object that you modify.

If you lock a project in the local repository, then consider the following:

- You cannot check in a project or object that is locked in the local repository.
- To check a project into the server repository, you must first check out this project only from the server repository.
- If you lock a project locally, and if you get or check out this project, then Siebel CRM overwrites this project and all objects that reference this project.

To lock a project in the local repository

1. Log in to Siebel Tools, connected to the local repository.
2. Do one of the following:
   - To lock a project from an object:
     - Locate an object. For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.
     - Right-click the object, and then click Lock Project.
   - To lock a project from the Projects list:
     - In the Object Explorer, click Project.
     - In the Projects list, locate the project that you must lock.
     - Make sure the Locked property contains a check mark.

   After you lock a project, you can edit all objects that reference this project.

Locking an Object in the Local Repository

You can lock an individual object that references a project that resides in your local repository. It is not necessary to check out this object from the server repository.

To lock an object in the local repository

1. Make sure the project allows object locking.

   For more information, see "Allowing Object Locking for a Project" on page 89.
2. In the Object List Editor, right-click the object, and then click Lock Object.

Unlocking a Project

This topic describes how to unlock a project.

To unlock a project

1. Log in to Siebel Tools, connected to the local repository or the server repository, depending on where the locked project resides.
2. In the Object Explorer, click Project.
3. In the Projects list, locate the project that you must unlock.

   For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.
4. Make sure the Locked property does not contain a check mark.

   Siebel Tools removes the lock from the project and all objects that reference this project. After you unlock a project, Siebel Tools makes all objects that reference the project available for editing. For more information, see "Using the W Property" on page 29.

Undoing Project Checkout

After you check out a project, you can undo the checkout. You can also use the Get option to do the following:

- Overwrite a project that you checked out from the server repository.
- To remove the lock for a project, check the project into the server repository.
- Enable projects for object checkin or checkout.

For more information, see "Getting Projects from the Server Repository" on page 86.

To undo project checkout

1. Click the Tools menu, and then click Check In.
2. In the Check In dialog box, choose the project or objects where you must undo checkout.
3. Click Undo Check Out.

   Siebel Tools does the following:

   - Unlocks the project or object that resides in the server repository.
   - Does not unlock the project or object that resides in the local repository.

   If one of the projects or objects that you choose is new, then Siebel Tools disables the Undo Check Out button. For more information, see "Elements of the Check In Dialog Box" on page 235.

Preventing Object Checkin and Checkout

You can lock the project that resides in the server repository to prevent a developer from checking it out or checking it in.

CAUTION: Modifying an object directly in the server repository for purposes other than preventing checkin and checkout is not recommended.

To prevent object checkin and checkout

1. Log in to Siebel Tools, connected to the server repository.
2. Do Step 2 on page 93.

   Siebel Tools locks the project and all objects that reference this project. Developers cannot check out these locked objects.

Viewing Locked Objects in a Project

You can view locked objects that reside in the server repository or the local repository.

To view locked objects in a project

1. Make sure the project allows object locking.

   For more information, see "Allowing Object Locking for a Project" on page 89.

2. In the Object Explorer, click Project.
3. In the Projects list, locate the project that includes the objects that you must view.

   For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

4. Right-click the project that you located in Step 3, and then click one of the following:

   - View Server Locked Objects
   - View Client Locked Objects

   Siebel Tools displays the View Locked Objects dialog box. This dialog box lists the objects that are locked that reference this project.

Viewing Differences Between Local Repositories and Server Repositories

Before you check in an object, you can compare the version of this object that the local repository contains to the version that the server repository contains. Siebel Tools compares the current state of this object to the state of the object when you checked it out.

To view differences between local repositories and server repositories

1. Open Siebel Tools.
2. Click the Tools menu, and then click Check In.
3. In the Check In dialog box, choose the project that you must compare.
4. Click Diff.

   Siebel Tools displays the Object Comparison dialog box. This dialog box lists the projects and any differences between the objects that the local repository contains and the objects that the server repository contains.

Setting Options for Checkout and Checkin

This topic describes how to set options for checkout and checkin. It includes the following information:

- Configuring Third-Party Code Control on page 96
- Configuring Siebel Tools to Restart Editors After Checkout on page 101

You can use the Development Tools Options dialog box to set options for checkin and checkout.

Configuring Third-Party Code Control

You can integrate the checkin feature in Siebel Tools with a code control system from a third party. You can use any third-party control system that Siebel CRM can start and control from a batch file. Some example third-party control systems include Borland Star Team, ClearCase, CVS Open Source, Microsoft Visual SourceSafe, Serena, and PVCS.

If you use third-party code control, then Siebel Tools creates an archive each time you check a project into the server repository. It also checks the project into the code control system. The archive includes all the objects that the project contains. The code control system can include successive versions of this project. You can use third-party code control only to check in items. You cannot use it to check out items. For more information, see Chapter 9, “Archiving Objects.”

To configure third-party code control

1. In Siebel Tools, click the View menu, and then click Options.
2. In the Development Tools Options dialog box, click the Check In/Out tab.
3. Set the options in the Source Control Integration section.

   For more information, see “Elements of the Check In Dialog Box” on page 235.
4. Click OK.

5. Modify the batch file.

   For more information, see "Configuring the Predefined Batch File" on page 98.

Using Checkin with Third-Party Code Control

After you configure Siebel Tools to use third-party code control, you can use Siebel Tools to check in projects just like you typically do. In this example, assume the following:

- Project A and Project B are currently checked out.
- You must check these projects into the server repository and into the code control system.
- Version 5 is the latest version of the ProjectA.sif file that resides in the code control system.
- Version 6 is the latest version of the ProjectB.sif file.

To use checkin with third-party code control

- In Siebel Tools, click the Tools Menu, and then click Check In.

  Siebel Tools does the following:

  - Checks Project A and Project B into the server repository.
  - Starts the srcctrl.bat file from the following folder:

    SIEBEL_TOOLS_ROOT`\BIN\`

  This batch file does the following work:

  - Checks out ProjectA.sif and ProjectB.sif. It locks these projects in the code control system.
  - Exports Project A to the ProjectA.sif file that resides in the *SIEBEL_TOOLS_ROOT*`\TEMP\projects\` folder.
  - Exports Project B to the ProjectB.sif file that resides in the *SIEBEL_TOOLS_ROOT*`\TEMP\projects\` folder.
  - Checks the ProjectA.sif file and the ProjectB.sif file into the code control system.
  - Sets the latest version of the ProjectA.sif file that resides in the code control system to version 6.
  - Sets the latest version of the ProjectAB.sif file that resides in the code control system to version 7.

Reverting to a Prior Version

In this example, you revert version 5 of Project A that you checked into the server repository. You revert this version to version 4.

To revert to a prior version

1. Check out version 4 of the ProjectA.sif file from the code control system into the following folder:

```
SIEBEL_TOOLS_ROOT\TEMP
```

2 Check out ProjectA from the server repository.

3 Import the ProjectA.sif file into the local repository. Use the Overwrite option to resolve object definition conflicts.

   This step replaces the existing version of ProjectA with the archived version.

4 Check ProjectA into the server repository.

   Siebel Tools checks the ProjectA.sif file into the code control system as version 6.

Configuring the Predefined Batch File

The srcctrl.bat batch file includes the commands that the code control software uses to check the archived projects into the code control system. You must modify this file, and then distribute it to your developers. The following command runs this batch file:

```
SRCCTRL action folder comment_file project_file
```

where:

- *action* is check in. You cannot use check out.
- *folder* is the folder that includes the items.
- *comment_file* is a file that includes comment text that Siebel Tools sends to the code control system with the project file.
- *project_file* is the name of the archive file for one project, enclosed in double quotes. For more information about archive files, see "Overview of Archiving Objects" on page 189.

You cannot modify this command. You cannot modify the parameters that it sends.

After checkin finishes, Siebel Tools runs the srcctrl.bat file one time for each project. It checks the archive file for the project into or out of the code control system. For information, see "Code of the Predefined Batch File" on page 98 and "About Predefined Objects" on page 24.

Support for Using the Predefined Batch File

Oracle can help you troubleshoot source integration, but modifying the srcctrl.bat batch file is your responsibility. To get support on third-party commands that you place in this batch file, you must contact the vendor who provides the third-party source control system.

Code of the Predefined Batch File

The following srcctrl.bat file comes predefined with Siebel CRM. You must modify this batch file so that it uses the path that your environment uses:

```
set PATH=C:\Program Files\DevStudio\Vss\win32\;%PATH%
set SOFTWARE=ss
set CHECKIN=%SOFTWARE% checkin
set CHECKOUT=%SOFTWARE% checkout
set ADD=%SOFTWARE% add
set SETPROJ=%SOFTWARE% cp
set PROJECT=$/PROJPOOL
set SRC_USR=
set SRC_PSWD=
set OPTIONS=-i-y -y%SRC_USR%,%SRC_PSWD%
set COMMENT=-c@
set NON_COMMENT=-c-
set FILE=
set LOGFILE=C:\Temp\xml.log
echo =======================srcctrl.bat========================== >> %LOGFILE%
set ACTION=%1
shift
set DIR=%1
shift
set COMMENT=%COMMENT%%1
shift
set FILE=%1
echo Change local folder to %DIR% >> %LOGFILE%
chdir %DIR% >> %LOGFILE% 2>&1
echo Set %PROJECT% as the working folder at Code Control System >> %LOGFILE%
%SETPROJ% %PROJECT% >> %LOGFILE% 2>&1
if errorlevel 100 goto END
if %ACTION%==checkout goto CHECK_OUT
if %ACTION%==checkin goto CHECK_IN
:CHECK_OUT
echo ============Check out file %FILE% from Code Control System============
if not exist %FILE% echo "New File" >> %FILE%
attrib +r %FILE%
echo Add %FILE% if it doesn't exist in Code Control System >> %LOGFILE%
%ADD% %FILE% %NON_COMMENT% %OPTIONS% >> %LOGFILE% 2>&1
echo Start checking out %FILE% from Code Control System >> %LOGFILE%
%CHECKOUT% %FILE% %NON_COMMENT% %OPTIONS% >> %LOGFILE% 2>&1
goto END
:CHECK_IN
echo ============Check in file %FILE% into Code Control System============
echo Check in %FILE% into Code Control System >> %LOGFILE%
%CHECKIN% %FILE% %COMMENT% %OPTIONS% >> %LOGFILE% 2>&1
attrib -r %FILE%
goto END
:END
echo ===================End Of srcctrl.bat====================== >> %LOGFILE%
```

Parameters That the Batch File Uses

Table 9 describes the parameters that the srcctrl.bat batch file uses.

Table 9. Parameters That the Batch File Uses for the Code Control Software

Parameter	Description
PATH	Identifies the folder where you install the code control software. You must modify this parameter so that it identifies this folder.
SOFTWARE	Name of the command-line utility that the code control system uses. For example, Microsoft Visual SourceSafe uses the ss command-line utility.
CHECKIN	Requests checkin into the code control system.
CHECKOUT	Requests checkout from the code control system.
ADD	Requests to add files in the code control system.
SETPROJ	Sets the working folder in the code control system.
PROJECT	Identifies the working folder in the code control system that includes the items that this system checks in and checks out.
COMMENT	Text of the Comments clause that runs in the command line for each file that the code control system checks in or checks out.
OPTIONS	Text of the Options clause that runs in the command line.
SRC_USR	User name that the Options clause includes in the command line. This user name is the logon name for the code control system. It is not the user for a Siebel application.
SRC_PSWD	User password that the Options clause includes in the command line. This password is the password for the code control system.
FILE	Name of the archive file. The parameter list of the batch file includes this file name. You must check in or check out this file.
LOGFILE	Path and file name of the log file that the batch file creates.

Running Simultaneous Siebel Tools Sessions After You Modify the ODBC Data Source

If you modify the ODBC data source, and then exit Siebel Tools, then Siebel Tools writes these modifications to the Siebel Preference Files (SPF) that resides in the following folder:

SIEBEL_TOOLS_ROOT\BIN

It caches these preferences. The next time you Log in to Siebel Tools, it does not read the ODBC settings in the tools.cfg file, and you cannot use a single user Id to run simultaneous Siebel Tools sessions. You can open only one local data source at a time. For more information, see "Specifying the Data Source That Siebel Tools Uses" on page 15 and "About the Configuration Files" on page 17.

Configuring Siebel Tools to Restart Editors After Checkout

If the Restart the Editors After Check Out check box includes a check mark, and if an editor is open when you do a checkout, then Siebel Tools restarts the editor after checkout finishes.

To configure Siebel Tools to restart editors after checkout

1 In Siebel Tools, click the View menu, and then click Options.

2 In the Development Tools Options dialog box, click the Check In/Out tab.

3 Make sure the Select the Restart the Editors After Check Out check box includes a check mark.

4 Click OK.

Customizing Projects

This topic describes how to customize projects. It includes the following information:

- Creating New Projects on page 101
- Renaming Projects on page 102
- Modifying the Project That an Object References on page 103
- Modifying the Location Where Siebel Tools Saves Objects and Projects on page 103

Creating New Projects

The projects that come predefined with Siebel CRM include a structure that you can use in a typical development effort. If you find that multiple developers must access the same set of objects simultaneously, then you can create a new project. You can create new projects that group related sets of new objects, or that separate existing objects into more manageable groups. For more information, see “About Predefined Objects” on page 24.

To create new projects

1 Create a plan for the new project:

   a Create an application development plan that includes a PERT (Program Evaluation and Review Technique) chart that documents dependencies and parallel activities.

   b Analyze the plan to determine whether or not developers must access objects that reside in the same project at the same time. If they must do this simultaneous access, then create a project plan that groups objects into separate projects.

   You can also enable projects for object checkin or checkout. For more information, see “Using Checkout and Checkin” on page 89

2 Create a new project:

a. In the Object Explorer, click Project.
b. In the Projects list, right-click, and then click New Record.
c. Enter a Name for the project, and then step off the record.
d. If necessary, create more projects to meet the requirements that you identified in Step 1.

You cannot use Siebel Tools to delete a project. If you must delete a project, then you must use SQL.

Creating Projects in Server Repositories

This topic describes how to create a project in the server repository.

To create projects in server repositories

1. Create a new project on the server repository.
2. Do a Get of the project to the local repository.

   For more information, see "Getting Projects from the Server Repository" on page 86.
3. Check out the project.
4. Modify the new project in the local repository.
5. To update the server repository, check in the project.

Renaming Projects

You can rename only a project that resides in the server repository. You cannot rename a project that resides in the local repository. You cannot modify the name of a top-level object that is checked out. For information about top-level object types, see "Displaying Object Types in the Object Explorer" on page 26.

To rename projects

1. Make sure developers check in all projects.
2. Log in to Siebel Tools, connected to the server repository.
3. Click the File menu, Open Repository, and then choose the repository that you must modify.
4. In the Object Explorer, click Project.
5. In the Projects list, locate the project that you must modify.

   For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

6 Lock the project, and then modify the Name property.

  CAUTION: Do not modify the value in the Name property of a predefined project. A predefined project is a project that comes with Siebel CRM. All projects that Siebel Tools shows in the Projects list immediately after you install Siebel CRM but before you do any customization are predefined projects. For more information, see "About Predefined Objects" on page 24.

7 Request each developer to do a Get of all projects that reside in the server repository.

8 Request each developer to do a full compile the next time each of these developers does a compile.

  For more information, see "Compiling Your Modifications" on page 179.

Modifying the Project That an Object References

You can modify the project that an object references. This configuration might be useful if you must separate a large project into smaller projects.

To modify the project that an object references

1 In the Object Explorer, click Project.

2 In the Projects list, locate the old project.

  For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

3 Check out the old project.

  For more information, see "Getting Projects from the Server Repository" on page 86.

4 In the Projects list, locate the new project.

5 Check out the new project.

6 Locate the object that you must modify, and then modify the Project property of this object to the name of the new project.

  For more information, see "Modifying Objects" on page 108.

7 Check in the projects that you checked out in Step 3 and Step 5.

Modifying the Location Where Siebel Tools Saves Objects and Projects

Siebel Tools temporarily stores an object or project that you check out or check in as an archive file in the following location:

- **Object.** Siebel Tools stores it in the *`client_root`*`\TEMP\Object` folder.
- **Project.** Siebel Tools stores it in the *`client_root`*`\TEMP\Project` folder.

You can modify this `TEMP` folder to another location. The exception is if you configure Siebel Tools to use third-party code control. If Siebel Tools uses this code control, then it must use the default folder. For more information, see "Where Siebel Tools Stores Archive Files" on page 190 and "Configuring Third-Party Code Control" on page 96.

5 Customizing Objects

This chapter describes how to customize objects. It includes the following topics:

- Overview of Customizing Objects on page 105
- Creating, Modifying, Copying, Validating, and Deleting Objects on page 107
- Examining Objects on page 111

Overview of Customizing Objects

The work that you can do with an object varies depending on whether or not the project that this object references does the following:

- **Allows object locking.** You can check out some of the objects in a project and leave other objects unlocked on the Siebel Server so that they are available to other developers to check out.
- **Does not allow object locking.** You must check out the entire project to edit any object that references this project.

Table 10 summarizes the work that you must do for various tasks, such as creating an object that references a project that allows object locking. For more information, see the following information:

- Allowing Object Locking for a Project on page 89
- Checking In Projects or Objects to the Server Repository on page 92
- Unlocking a Project on page 94

Table 10. Summary of Work You Must Perform Depending on Project Locking

Task	Project Allows Object Locking	Project Does Not Allow Object Locking
Create object	You must do the following: 1. Lock the project locally. 2. Create the new object. 3. Unlock the project. 4. Check in the object.	You must do the following: 1. Check out the project. 2. Create the new object. 3. Check in the project.
Modify object	You must do the following: 1. Check out the object. 2. Modify the object. 3. Check in the object.	You must do the following: 1. Check out the project. 2. Modify the object. 3. Check in the project.

Table 10. Summary of Work You Must Perform Depending on Project Locking

Task	Project Allows Object Locking	Project Does Not Allow Object Locking
Create new object by copying an existing one	You must do the following: 1. Check out the object that you must copy. 2. Lock the project that the object references locally. 3. Copy the object and assign the project. This step refers to the same project that you lock in step 2 or to a different project that does not allow object locking. 4. Unlock the project locally. 5. Lock the new object locally. 6. Check in the object to the server repository.	You must do the following: 1. Check out projects. 2. Copy the object and create a new one. 3. Check in projects.
Delete object	Cannot perform. The Allow Object Locking property must be set to FALSE.	You must do the following: 1. Check out the project. 2. Delete the object. 3. Check in the project.
Rename object		You must do the following: 1. Check out the project. 2. Rename the object. 3. Check in the project.
Assign object to different project		You must do the following: 1. Check out the source project and the destination project. 2. Modify the object so that it references the destination project. 3. Check in the source project. 4. Check in the destination project.

Creating, Modifying, Copying, Validating, and Deleting Objects

This topic describes how to create, modify, copy, or delete an object. It includes the following information:

- Creating Objects on page 107
- Modifying Objects on page 108
- Copying Objects on page 108
- Validating Objects on page 109
- Deleting Objects on page 110

This task is a step in "Roadmap for Setting Up and Using Siebel Tools" on page 14.

Creating Objects

You can use a wizard to create an object, or you can create it manually. It is recommended that you use a wizard. For example, to create a new business component, you use the Business Component Wizard. A wizard guides you while you configure a new object. It prompts you for the required property values and automatically configures any required child objects. If a wizard is not available for the object type that you must create, then you can create it manually in the Object List Editor. For more information about using wizards and creating objects, see *Configuring Siebel Business Applications*.

Using a Wizard to Create Objects

This topic describes how to use a wizard to create an object.

To use a wizard to create objects

1. Click the File menu, and then click New Object.
2. Choose the appropriate wizard to create the new object.
3. Follow the instructions that the wizard shows.

Using the Object List Editor to Create New Objects

This topic describes how to use the Object List Editor to create a new object. For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

To use the Object List Editor to create new objects

1. In the Object Explorer, click the object type that you must create.
2. Right-click in the Object List Editor, and then click New Record.

3. Enter property values in the new row in the Object List Editor.

   You must enter values in the Name and Project properties. Siebel Tools might require that you set other properties, depending on the object type.

4. To save your modifications, click anywhere outside of the new row.

Modifying Objects

You can use the Object List Editor or the Properties window to modify an object. For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24. For guidelines about when to modify an object or when to create a new object, see *Configuring Siebel Business Applications*.

Using the Properties Window to Modify Objects

This topic describes how to use the Properties window to modify an object.

To use the Properties window to modify an object

1. In the Object List Editor, locate the object that you must modify.

   For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

2. Click the View menu, Windows, and then click Properties.
3. In the Properties window, enter a new value for a property.
4. To save your modifications, choose another property or click anywhere outside the Properties window.

   Siebel Tools saves your modifications and places a check mark in the Changed property.

Renaming Objects

It is recommended that you copy an object, and then rename the copy instead of only renaming the original object. If you rename an object, then Siebel Tools might display an error message that is similar to the following:

```
Modifying the name of a checked out or locked object causes "unique constraint" error
during check-in. To avoid this error, modify the name of the object back to the
original name. Do you must continue?
```

For more information, see "Copying Objects" on page 108.

Copying Objects

This topic describes how to copy an object. For guidelines about copying an object, see *Configuring Siebel Business Applications*.

To copy an object

1. In the Object List Editor, locate the object that you must copy.

   For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

2. Click the Edit menu, and then click Copy Record.

   Siebel Tools inserts a new record above the record that you copied. This new record includes all of the same values as the record that you copied, except the Name property is empty and the Changed property contains a check mark. Siebel Tools also creates a copy of any child objects for the record you copied. For example, if you copy the Account business component, then it creates a new business component that includes several hundred business component fields.

3. Enter a new value for the Name property.

4. If necessary, modify other properties and child objects.

5. To save your modifications, click anywhere outside of the new row.

Validating Objects

The *Validate Tool* is an error correction tool that you can use to make sure each object that you customize does not contain a configuration error. For example, you can use it to make sure an error workflow process does not itself contain an error workflow process. When you validate a workflow process, Siebel Tools displays warnings that describe errors that the workflow process contains. The validation dialog box allows you to correct these errors. This dialog box is without a mode. You can keep it open to view the error messages while you correct the problems that the dialog box reports, or you can proceed with deployment without fixing these validation errors.

Validation uses a set of rules that help make sure your configuration modifications are logically consistent with other objects. If you validate an object, then Siebel Tools also validates the child objects of this object. For example, if you validate the Account business component, then Siebel Tools also validates the child objects of the Account business component, such as fields, joins, and so on.

Using the Object List Editor to Validate Objects

This topic describes how to use the Object List Editor to validate an object.

To use the Object List Editor to validate objects

1. In the Object List Editor, locate the object that you must delete.

   For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

2. Click the Tools menu, and then click Validate Object.

   Siebel Tools displays the Validate dialog box. For more information, see "Elements of the Validate Dialog Box" on page 237.

3. Click Options.
4. In the Validation Options dialog box, in the Rules section, choose the validation rules that Siebel Tools must enforce.

   For more information, see "Elements of the Validation Options Dialog Box" on page 238.
5. (Optional) In the Time Filter section, choose one of the following options:
   - **Last Validated.** Validates objects that you modified since the last time you ran validation.
   - **Custom.** Validates objects that you modified since the date and time that you enter. Enter a date and time to use this option.
6. (Optional) In the Action section, choose the following options:
   - **Do Not Report Warnings.** Siebel Tools reports only errors, not warnings. It sets the Enforce field for warnings to No.
   - **Abort Validation After.** You enter a number in the window that this section shows. If Siebel Tools identifies the number of errors that you specify in this window, then it stops validating the object, and then displays the Error dialog box. For example, if you enter 8, then Siebel Tools stops validating after it identifies 8 errors.
7. Click OK.
8. In the Validate dialog box, click Start.

   Siebel Tools validates the object, and then displays any errors that it finds in the Validate dialog box.

Using the Command Line to Validate Objects

You can use the command line to validate objects.

To use the command line to validate objects

1. Open a command line, and then navigate to the following folder:

   SIEBEL_TOOLS_ROOT\BIN
2. Enter the following command:

   ```
   siebdev.exe /bv
   ```

   The /bv switch runs all validation rules for the entire repository.

Deleting Objects

This topic describes how to delete an object.

To delete objects

1 In the Object List Editor, locate the object that you must delete.

For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

2 Make sure the project that the Project property references does not allow object locking.

For more information, see "Allowing Object Locking for a Project" on page 89.

3 Check out the project that you examined in Step 2.

For more information, see "Checking Out Projects from the Server Repository" on page 90.

4 Click the Edit menu, and then click Delete Record.

CAUTION: It is recommended that you do not delete a predefined object. Other objects might reference this predefined object. Instead, you can make this object inactive. For more information, see "Using the Inactive Property" on page 30.

Siebel Tools deletes the object, and it also deletes any child objects of this object. For example, assume you create a business component named My Account Business Component. If you delete this business component, then Siebel Tools deletes all child objects of this business component, such as fields. It does not delete objects that only reference the object that you delete. For example, if you delete a view, then it does not delete the applets that reference this view.

Examining Objects

This topic describes how to examine objects. It includes the following information:

- Searching the Repository on page 111
- Viewing Object Relationships on page 113
- Comparing and Synchronizing Objects Between Repositories and Archives on page 114
- Comparing Different Versions of a Workflow Process or Task UI on page 117
- Determining When Siebel CRM Created or Updated a Record on page 118

Searching the Repository

This topic describes how to locate object definitions for multiple object types. For information about how to locate object definitions for a single object type, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

If you know that one or more properties includes a value, then you can search the repository across all properties of multiple object types to locate the objects that include this value.

To search the repository

1 Click the Tools menu, and then click Search Repository.

Searching the repository can consume a significant amount of time. You can click Cancel in the Search Repository dialog box to cancel a search at any time while the search runs.

2 In the Search Repository dialog box, in the Parameters section, enter the search criteria in the Search value window.

3 To limit the search to values that are case-sensitive, make sure the Case Sensitive check box contains a check mark.

For example, to locate objects that include a value of Account and not account.

4 To limit the search to values that match the entire search string that you enter in Step 2, make sure the Exact Match check box contains a check mark.

5 In the Types to Search list, choose the object type that Siebel Tools must search.

Note the following:

- Siebel Tools searches all object types, by default.
- You can click the object type to choose a single object type.
- You can use multiselect to choose multiple object types.

  For more information, see "Choosing More Than One Record in the Object List Editor" on page 44.
- For improved performance, search only the minimum number of object types.

6 Click Search Now.

Siebel Tools runs the search and lists the results in the window that resides at the bottom of the Search Repository dialog box. This window lists all the objects that match the search criteria. It includes the following columns:

Column	Description
Type	Object type of the object that the search returns.
Name	Name of the object that the search returns.
Property	Property name of the object that the search returns.
Value	Property value of the object that the search returns.

7 To display the object definition of an item that the search returns, double-click this item in the results window.

Siebel Tools displays the object definition of the item in the Object List Editor.

8 To export the search results to a file, click Export.

Viewing Object Relationships

You can use a visualization view to examine how objects relate to each other.

To view object relationships

1. Open Siebel Tools.
2. Set options for the visualization views:
   a. Click the View menu, and then click Options.
   b. In the Development Tools Options dialog box, click the Visualization tab, and then set the options.

      For more information, see "Development Options for Visualization Views" on page 242.
3. To open a visualization view, do one of the following:
   - Click the View menu, click Visualize, and then click one of the following menu items:
     - View Details
     - View Relationships
     - View Descendents
     - View Web Hierarchy
   - Right-click an object in the Object List Editor, and then choose a Visualization view.

Table 11 describes the Visualization views.

Table 11. Description of Visualization Views

View	Description
Details	Displays a diagram that illustrates relationships depending on the following object type that you choose in the Object List Editor: ■ **Business Component.** Illustrates the relationship between this business component and tables that this business component references. ■ **Business Object.** Illustrates the relationship between this business object and the links and business components that this business object references.
Relationships	Displays a diagram that illustrates relationships depending on the following object type that you choose in the Object List Editor: ■ **Business Component.** Illustrates the relationship between this business component and links to other business components. ■ **Table**. Illustrates the relationship between this table and other tables.

Table 11. Description of Visualization Views

View	Description
Descendents	Displays a dialog box that lists all objects that reference the current object in their Upgrade Ancestor property. For example, if you right-click the Proposal Template Section business component, and then choose View Descendents, then Siebel Tools lists the Proposal Template Section Hierarchy business component. This is the only business component that includes Proposal Template Section in the Upgrade Ancestor property.
Web Hierarchy	Displays a diagram that illustrates a Web hierarchy depending on the following object type that you choose in the Object List Editor: ■ Applet ■ Application ■ Business Component ■ Screen ■ View The Web hierarchy displays the parent-child relationships between the object you choose and the parent and child objects of this object throughout the hierarchy.

Comparing and Synchronizing Objects Between Repositories and Archives

You can compare two objects that are of the same object type. Siebel Tools uses color-coded icons to highlight differences. You can choose and copy properties and individual child objects from one object to another object. You can use this feature to propagate a modification that you make to an ancestor object to the descendents of this object or to other objects that are of a similar type. You can assess and adjust differences between objects. You can also compare properties of checked out objects with their counterparts on the Siebel Server. For more information about ancestor objects, see *Configuring Siebel Business Applications*.

You can compare two objects that are of the same type. The Object Comparison dialog box displays a line-by-line comparison between the two objects. You can compare the top-level objects that are defined in the following items:

- Current repository
- Different repositories
- Archive files

For information about top-level object types, see “Displaying Object Types in the Object Explorer” on page 26.

Comparing Two Objects in the Same Repository

You can compare two objects that reside in the same repository.

To compare two objects in the same repository

1. In the Object Explorer, click an object type.
2. In the Object List Editor, choose two top-level object types.

   For more information, see "Choosing More Than One Record in the Object List Editor" on page 44.
3. Click the Tools menu, Compare Objects, and then click Selected.
4. Examine the results in the Compare Objects dialog box.

   For more information, see "Elements of the Compare Objects Dialog Box" on page 239.

Comparing Two Objects in Different Repositories

You can compare an object that resides in the current repository to an object that resides in a different repository.

To compare two objects in different repositories

1. In the Object Explorer, click an object type.
2. In the Object List Editor, choose one top-level object type.
3. Click the Tools menu, Compare Objects, and then click Selected vs. Repository.
4. In the Open Repository dialog box, choose the repository that includes the object you must compare, and then click Open.
5. Examine the results in the Compare Objects dialog box.

   This dialog box displays the following:

   - The object in the current working repository in the left window
   - The corresponding object in the other repository in the right window

   You can update the current working repository or the other repository from the Object Comparison dialog box. To do an update, you must make sure the project that the object references is locked. For more information, see "Elements of the Compare Objects Dialog Box" on page 239.

Comparing an Object in the Current Repository to an Object in an Archive

You can compare an object that resides in the current repository to an object that resides in an archive.

To compare an object in the current repository to an object in an archive

1. In the Object Explorer, click an object type.
2. In the Object List Editor, choose one top-level object type.
3. Click the Tools menu, Compare Objects, and then click Selected vs. Archive.

4 In the Select Archive File to Compare Against dialog box, choose the archive file that includes the object you must compare.

  Siebel Tools saves an archive file as a SIF file. For more information, see "Overview of Archiving Objects" on page 189.

5 Click Open.

  Siebel Tools starts the comparison at the project level. It does the following:

  - If it finds a corresponding object in the archive, then it displays the Compare Objects dialog box.
  - If it does not find a corresponding object in the archive, then it does nothing.

  For more information, see "Elements of the Compare Objects Dialog Box" on page 239.

Comparing Objects in Two Different Archives

You can compare an object that resides in an archive to an object that resides in another archive.

To compare objects in two different archives

1 In the Object Explorer, click an object type.

2 In the Object List Editor, choose one top-level object type.

3 Click the Tools menu, Compare Objects, and then click Archive vs. Archive.

4 In the Select Archive File for Left Side of Comparison dialog box, choose an archive file, and then click Open.

5 In the Select Archive File for Right Side of Comparison dialog box, choose an archive file, and then click Open.

  Siebel Tools displays the Object Comparison dialog box. It populates the left side and right side of this dialog box with the contents of the archives you choose in Step 4 and Step 5. During the comparison, these archives are read-only. For more information, see "Elements of the Compare Objects Dialog Box" on page 239.

Synchronizing Objects Between Repositories

You can use the Compare Objects dialog box to synchronize objects.

To synchronize objects between repositories

1 Lock the projects that the objects that you must synchronize reference.

  For more information about projects, see Chapter 4, "Checking Out and Checking In Projects and Objects."

2 In the Object List Editor, choose two top-level object types of the same object type.

  For more information, see "Choosing More Than One Record in the Object List Editor" on page 44.

3 Click the Tools menu, click Compare Objects, and then click Selected.

4 In the Compare Objects dialog box, choose an object in the First Selection box.

For more information, see “Elements of the Compare Objects Dialog Box” on page 239.

5 Click the right arrow button.

Siebel Tools does one of the following, depending on whether or not a corresponding object exists in the repository that the Second Selection section represents:

- **Corresponding object exists.** Siebel Tools modifies the properties of the object that resides in the repository that the Second Selection section represents. It modifies these properties so that they are equal to the object properties of the object that the First Selection section shows.
- **Corresponding object does not exist.** Siebel Tools creates a corresponding object in the repository that the Second Selection section represents.

If you synchronize an object from one repository to another repository, then Siebel Tools synchronizes this object and any child objects that this object includes.

Comparing Different Versions of a Workflow Process or Task UI

Siebel Tools allows you to compare two different versions of the same workflow process or task UI. It uses two separate windows to display these differences.

To compare different versions of a workflow process or task UI

1 Log in to Siebel Tools.

2 In the Object Explorer, click Workflow Process.

Alternatively, you can compare two versions of a task UI.

3 In the Workflow Processes list, choose two versions of the same workflow process.

Make sure you choose two versions of the same workflow process and that these versions include different values in the Version property. Siebel Tools allows you to choose two entirely different workflow processes, but the comparison is not meaningful. You must click Revise to create a new version of the same workflow process.

For more information about Revise, see “WF/Task Editor Toolbar” on page 228. For more information about choosing two workflow versions, see “Choosing More Than One Record in the Object List Editor” on page 44.

4 Click the Tools menu, Compare Objects, and then click Selected.

Siebel Tools displays each workflow process version in a separate window. It displays the first record that you choose in Step 3 in the top window and the second record that you choose in Step 3 in the bottom window. It compares the first record to the second record. It considers the first record as the newer record. It uses the following colors to indicate differences between these records:

- **Yellow.** You modified the object. For example, you added a process property to a business service step.
- **Red.** You deleted an object from the old version.
- **Green.** You added an object to the new version.

Note the following behavior:

- If you click an object that is not red or green in one window, then Siebel Tools chooses the corresponding object in the other window. This functionality works only if a corresponding object exists. For example, if you click a red or green object, then Siebel Tools does not choose any object in the other window because no corresponding object exists.
- You can double-click the object to view details about your modifications.
- You can repeat Step 3 and Step 4 to compare multiple sets of workflows. Siebel Tools displays each set in a separate tab. You can click each tab to navigate between sets.
- To compare the workflow process properties of the two versions, you can double-click the background of either version, and then Siebel Tools displays an object comparison dialog box that allows you to view the differences for process properties instead of the differences between process steps. If a difference exists between the process properties of the two versions, then Siebel Tools sets the background color of one of the versions to yellow.

Determining When Siebel CRM Created or Updated a Record

You can determine the last time Siebel CRM created or updated a record and who made the modification or update.

To determine when Siebel CRM created or updated a record

1. In the Object List Editor, locate the object you must modify.

   For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

2. Click the Help menu, and then click About Record.
3. In The Siebel Tools dialog box, click Details to display more information about the record.

6 Using Siebel Script Editors

This chapter describes how to use the Siebel Script Editors. It includes the following topics:

- Using the Siebel Script Editor on page 119
- Using the ST eScript Engine on page 123
- Using the Siebel Debugger on page 132
- Using the Script Profiler on page 138

Using the Siebel Script Editor

This topic describes how to use the Siebel Script Editor. It includes the following information:

- Overview of Using Siebel Script Editor on page 119
- Setting Options for the Siebel Script Editor on page 121
- Opening the Siebel Script Editor on page 122

Overview of Using Siebel Script Editor

The *Siebel Script Editor* is an editor that allows you to create and maintain scripts that use Siebel VB, Siebel eScript, or Browser Script. If you cannot use a declarative configuration to implement the functionality you require, then you can use the following editors to add a script to a Siebel object:

- **Server Script Editor.** Allows you to create and modify Siebel eScript or Siebel VB.
- **Browser Script Editor.** Allows you to create and modify Browser Script that runs in the Siebel client.

For more information, including a description of scriptable events and callable methods on browser objects, see *Siebel Object Interfaces Reference*, *Siebel eScript Language Reference* and *Siebel VB Language Reference*.

About the Siebel Script Editor

You can use the Siebel Script Editor to edit Siebel VB script or Siebel eScript script. It allows you to do the following:

- Cut, copy, and paste the text from one location to another location in the Siebel Script Editor. You can also cut and paste text into the Siebel Script Editor.
- Import or export a Siebel script.
- Associate a given Siebel script with a predefined object event, such as a PreSetFieldValue event for a business component. For more information, see "About Predefined Objects" on page 24.

- Debug custom script. For more information, see "Using the Siebel Debugger" on page 132.
- Compile custom script. For more information, see "Compiling Your Modifications" on page 179.

The Siebel Script Editor uses a window that is similar to the Windows Notepad editor. It includes the following elements:

- Title bar
- Drop-down list that you can use to specify an object
- Drop-down list that you can use to specify an event
- Text entry window
- Vertical and horizontal scroll bars that you can use to navigate.

Object Types You Can Script

You can use the Siebel Script Editor to add a script to the following object types:

- Application
- Applet
- Business Component

An object definition for each of these object types includes a Scripted property. If this property includes a check mark, then this object includes a script.

Menus in the Siebel Script Editor

The following items are available from the File menu:

- **Import.** Imports a Siebel script.
- **Export.** Exports a Siebel script.
- **Save.** Saves a Siebel script. Make sure you save your script before you exit the editor.
- **Exit.** Closes the Siebel Script Editor window.

The following items are available from the Edit menu:

- **Cut.** Deletes the selection and saves it to the clipboard.
- **Copy.** Copies the selection to the clipboard.
- **Paste.** Copies the contents of the clipboard to the chosen area.
- **Delete.** Deletes the selection.
- **Select All.** Selects the entire script.
- **Find.** Displays the Find in Script dialog box. You can search for text or white space.
- **Replace.** Displays the Replace in Script dialog box. You can search and replace text or white space.

Guidelines for Using Siebel Script Editor

If you use the Siebel script editor, then it is recommended that you use the following guidelines:

- Check out or lock the project that the object definition that you modify references. If the project is not locked, then you cannot add text in the Editor window. For more information about projects, see Chapter 4, "Checking Out and Checking In Projects and Objects."
- To verify the format of your Siebel VB or Siebel eScript script, click the Debug menu, and then click Check Syntax. The Siebel Compiler displays any format errors it finds and indicates the lines where these errors occur.
- Save your work frequently and before you close the Siebel Script Editor. Click the File menu, and then click Save to save your work. If you close the Siebel Script Editor without saving your work, then Siebel Tools discards any modifications you make since the last save.
- You must compile the projects that you modify before you run a Siebel application. For more information, see "Compiling Your Modifications" on page 179.
- When pasting text into the Siebel Script Editor, avoid using two code blocks that use the same name. To do this, do the following depending on the type of script you use:
  - **Siebel eScript.** Place the code between `function` *`Name`* `{` and `}.` You must enclose all function code with curly braces.
  - **Siebel VB.** Place the code between `Sub` *`Name`* and `End Sub`.

Guidelines for Testing a Script

If you test a script, then it is recommended that you use the following guidelines:

- To test a script, you can click the Debug menu, and then click Start. Siebel CRM runs with the new modifications incorporated. You can also click the Start button in the Debug toolbar.
- To display an error when you test your script, you can start Siebel CRM in debug mode. To use debug mode, use the `/H` option on the start-up command line. If it encounters an error in your script, then it displays a dialog box that describes the error. To troubleshoot the error, you can open the Script Editor, click the Debug menu, and then click Check Syntax. For more information, see "Validating Script Syntax" on page 134.

  If a run-time error occurs in the Siebel client, or if Siebel CRM does not run in Debug mode, then it displays an error message that includes an error code. It returns control to the location in the predefined Siebel code just before the error occurred.

Setting Options for the Siebel Script Editor

You can set options for working in the Siebel Script Editor, including setting a default scripting language, specifying a location for compiling browser scripts, and defining options for debugging. Script Assist options are available only if the ST eScript Engine is enabled. For more information, see "Overview of Using the ST eScript Engine" on page 123.

To set options for the Siebel Script Editor

1. In Siebel Tools, click the View menu, and then click Options.
2. In the Development Tools Options dialog box, click the Scripting tab.
3. Set the options.

   For more information, see "Development Options for Scripting" on page 242.

Suppressing Error Messages

This topic describes how to configure Siebel Tools to not display the SBL-EXL-00151 error code and error text in the pop-up error message that the RaiseErrorText method creates.

To suppress error messages

1. Open Siebel Tools.
2. Click the Screens menu, System Administration, and then click System Preferences.
3. In the System Preferences list, set the Suppress Scripting Error Code system preference using values in the following table.

System Preference Name	System Preference Value
Suppress Scripting Error Code	TRUE

Opening the Siebel Script Editor

This topic describes how to open the Siebel Script Editor.

To open the Siebel Script Editor

1. In the Object Explorer, click one of the following object types:
   - Application
   - Applet
   - Business Component
2. In the Object List Editor, locate the object you must modify.

   For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.
3. Do one of the following:
   - Right-click the object you located in Step 2, and then click Edit Server Script or Edit Browser script.
   - Click the View menu, Editors, and then click Server Script Editor or Browser Script Editor.

Using the ST eScript Engine

This topic describes how to use the ST eScript Engine. It includes the following information:

- Overview of Using the ST eScript Engine on page 123
- Enabling the ST eScript Engine on page 127
- Setting Options for the ST eScript Engine on page 127
- Opening the Script Assist Window on page 128
- Using Fix and Go on page 128
- Using Running Tool Tip on page 129
- Using Script Libraries with the ST eScript Engine on page 130

Overview of Using the ST eScript Engine

The following versions of the Siebel eScript Engine are available:

- **ST eScript Engine.** Available starting with Siebel CRM version 7.8 and higher. The ST eScript Engine is the default Siebel eScript Engine in version 8.0.
- **T eScript Engine.** Available before Siebel CRM version 7.8.

It is recommended that you use the ST eScript Engine. Starting with Siebel CRM version 8.0, you can use only the ST eScript Engine.

Except for a few differences, the ST eScript Engine is compatible with script that you create with the T eScript Engine. This document refers to each engine by name only if it must describe a difference in functionality between the ST eScript Engine and the T eScript Engine.

The ST eScript Engine is compliant with the ECMAScript Edition 4 standard. ECMAScript is the implementation of JavaScript as defined by the ECMA -262 standard. The ST eScript Engine includes the following capabilities:

- **Performance.** Provides higher throughput with a lower CPU and memory footprint in situations where you use a significant amount of script. The result is improved performance and lower maintenance on heavily scripted events.
- **Scalability.** If many users concurrently run scripts, then the performance of the ST eScript Engine is superior to the performance of the T eScript Engine.
- **Strong typing.** Supports strong typing that is compliant with the ECMAScript Edition 4 standard. Strongly typed objects result in scripts that are more functional and improved performance. The T eScript Engine does not support strong typing.
- **Functionality.** Adds new functionality, such as Script Assist, script libraries, favorites, and Fix and Go. For more information, see "About Script Assist" on page 125 and "Using Fix and Go" on page 128.

For more information, see *Siebel eScript Language Reference*.

About ST eScript Engine Warnings

The ST eScript Engine can display warnings that notify you of the following potential problems that might occur if you compile your custom script:

- Referencing a method or property that is not predefined
- Referencing an undeclared identifier
- Using a variable before it initializes this variable
- Using a redundant declaration of an untyped variable
- Calling a function that includes an insufficient number of arguments

These errors might cause a run-time failure. You can use these warnings to help fix errors before you compile your script. To enable or disable the Enable Warnings option, see "Setting Options for the ST eScript Engine" on page 127.

Example of a Warning Message

The following code is an example of a compile warning message that Siebel Tools creates after a run-time failure:

```
function foo(a)
  {
  var oApp: Application;
     oApp.myMethod ();
  return;
  }
foo ();
Semantic Warning around line 5:Variable oApp might not be initialized.
Semantic Warning around line 5:No such method myMethod
Semantic Warning around line 10:Calling function foo with insufficient number of
arguments.
Unhandled Exception: Function expected
```

How the ST eScript Engine Determines the Type of Variables That a Script Uses

Type deduction is a feature of the ST eScript Engine that determines the type of local variables that a script uses. It scans the assignments that the script makes to each local variable. If the ST eScript Engine can successfully determine the type for all local variables, then the compiler performs strict type checks and creates statically bound code that runs faster and uses less memory. This configuration might introduce more compile warnings as a result of performing these type checks. It cannot determine the type in all situations. It is recommended that you type your script. To enable or disable type deduction, see "Setting Options for the ST eScript Engine" on page 127.

Example of a Script That Deduces the Local Variable Type

The following script deduces the type of the oDate local variable to the Date. It then creates a warning about the MyMethod method. This method is not defined. This script fails at run time:

```
function goo()
  {
  var oDate;
     oDate = new Date ()
     oDate.myMethod ();
  return;
  }
goo ()
Semantic Warning around line 19:No such method myMethod
Unhandled Exception: 'myMethod' is not defined
```

About Script Assist

Script Assist is part of the ST eScript Engine. To help you develop a script, it inspects object definitions, and then makes information about these object definitions available to you. It includes the following functionality:

- **Syntax highlight.** Uses color to highlight reserved words, data types, operators, and other syntax in Siebel VB and Siebel eScript script. You cannot modify these colors. The following table lists the colors that it uses:

Item In the Code	Color
Reserved word or Siebel VB statement.	Blue
Data type.	OrangeRed
Operator.	Navy
String literal.	SteelBlue
Delimiter. For Siebel eScript only.	Brown
Function. For Siebel VB only.	Magenta

- **Method list.** Displays a list of methods and properties that are available for an object. For more information, see "Icons That the Script Assist Window Contains" on page 126.
- **Repository inspection.** Inspects objects and object types in the repository without requiring you to type a string literal. This functionality results in fewer mistakes in your script. It also understands predefined constants for a business component method.
- **Favorites.** Uses italics to indicate the most frequently used object, method, or property name in the Script Assist window. Favorites exist for only a single Siebel Tools session. When you log out of Siebel Tools, it clears these favorites. If you create a new function, then you must add it to the declarations, and then save the script modifications. If you do not do this, then Siebel Tools does not display the function as a favorite.
- **Script libraries.** Allows you to call a business service function after you declare the business service. You do not declare property sets or make an InvokeMethod call. A script library helps you develop code that is reusable and modular. For more information, see "Using Script Libraries with the ST eScript Engine" on page 130.

- **Auto complete.** Automatically completes an entry after you enter a minimum number of unique characters in the Script Assist window. For example, if you enter Bus, then Siebel Tools automatically enters the word BusComp.
- **Auto indent.** Maintains a running indent. If you press the Return key or the Enter key, then it inserts spaces and tabs that left-justifies the code.
- **Tool tips.** Allows you to view descriptions of the method arguments that you use.
- **Includes child scripts.** Script Assist can parse a script that you write on a business component, applet, or business service. You can use the Application drop-down list to choose the Siebel application that includes the child script. Script Assist displays the scripts that are written on this application object in the Script Assist window.
- **Includes scripts you write in the general section.** A script that you write in the general section of the script explorer window is available in the Script Assist window. For example, if you write a helper function named Helper in the general section of the current script, and if you start Script Assist, then it includes Helper in a pop-up window.

Using Script Assist with a Custom Siebel eScript Method

If a script references a custom Siebel eScript method that resides on a business component, and if this script does not reside on the same business component that includes this custom method, then Script Assist does not recognize the custom method and cannot display data from it. For example, assume you create a business component method named MyMethod on the Account business component. You then create a script on the Contact business component that references the MyMethod method. In this situation, Script Assist does not include any information about your custom MyMethod method.

Icons That the Script Assist Window Contains

Table 12 describes the methods and properties that the Script Assist window shows when you choose an object. For more information, see "Opening the Script Assist Window" on page 128.

Table 12. Icons That the Script Assist Window Contains

Icon	Description
▤	Read-only property
✎	Modifiable property
▣	Method
⊶	Class object
▭	Primitive

Enabling the ST eScript Engine

If you must use the T eScript Engine, then you can disable the ST eScript Engine in Siebel Tools. If you disable the ST eScript Engine, you can enable it later.

To enable the ST eScript Engine

1. Open Siebel Tools.
2. Click the Screens menu, System Administration, and then click System Preferences.
3. In the System Preferences list, set the Enable ST Script Engine system preference to one of the following values:
   - **TRUE.** Enables the ST eScript Engine.
   - **FALSE.** Disables the ST eScript Engine.

   CAUTION: Disabling the ST eScript Engine is not recommended. For help with disabling this engine, 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.

   If you must revert to the T eScript Engine after you use the ST eScript Engine, and if you use strongly typed code while you use the ST eScript Engine, then you must undo this strongly typed code before you revert to the T eScript Engine. For more information about setting system preferences, see *Siebel Applications Administration Guide*.
4. Recompile your scripted objects.

   For more information, see "Compiling Your Modifications" on page 179.
5. To use the Siebel eScript Engine, exit Siebel Tools, and then reopen it.

Enabling the ST eScript Engine in the Siebel Application

If you enable the ST eScript Engine in the development environment, then make sure you also enable it in the Siebel application. You must use the same Siebel eScript Engine setting when you compile code in these environments, and you must use the same engine setting for these environments that you use to compile the code.

Setting Options for the ST eScript Engine

This topic describes how to set options for the ST eScript Engine.

To set options for the ST eScript Engine

1. Make sure the ST eScript Engine is enabled.

   For more information, see "Enabling the ST eScript Engine" on page 127.
2. In Siebel Tools, click the View menu, and then click Options.
3. In the Development Tools Options dialog box, click the Scripting tab.

4 To define the options in the Engine Settings section, use the information in the following table. It is recommended that you enable all of these options.

Option	Description
Enable Warnings	If this check box includes a check mark, then Siebel Tools displays script compile warning messages. For information, see "About ST eScript Engine Warnings" on page 124.
Deduce Types	If this check box includes a check mark, then Siebel Tools deduces the type of local variables. For more information, see "How the ST eScript Engine Determines the Type of Variables That a Script Uses" on page 124.
Fix and Go	If this check box includes a check mark, then Siebel Tools allows you to test and debug your script without compiling. For more information, see "Using Fix and Go" on page 128.

Opening the Script Assist Window

You can open the Script Assist window.

To open the Script Assist window

1 In the Script Editor Explorer window, choose the object you must modify.

2 Press CTRL+SPACE.

Siebel Tools displays the Script Assist window. This window lists all methods and properties that are available for the object you choose.

Using Fix and Go

If you enable Fix and Go, then you can edit a script in a Siebel Tools session, and then test your modifications in the Siebel client without closing this client or recompiling the script. This feature allows you to save time in script development, testing, and debugging. You can use Fix and Go only with a server script and the ST eScript Engine.

To use Fix and Go

1 Enable Fix and Go in the Development Tools Options window.

For more information, see "Setting Options for the ST eScript Engine" on page 127.

2 Create a server script in the Siebel Script Editor, save it, and then compile the repository.

If you attempt to save a script that includes a format error, then Siebel Tools displays a script error message that prompts you to fix the line or the lines that cause the error. For more information, see "Compiling Your Modifications" on page 179.

3 Use the Siebel Debugger to run the script.

4 If an error occurs, then stop the script.

5 Make modifications, save your script, and then run it again.

You must save and compile all script modifications before you exit Siebel Tools. If you do not save your modifications, then they are lost.

Using Running Tool Tip

Running Tool Tip is a help feature in the Siebel Script Editor that you can use to write a script. If you enter a method name in the Running Tool Tip window, then it displays the method arguments that you can use with this method. If you enable Script Assist, then Siebel Tools enables Running Tool Tip, by default. For more information, see "About Script Assist" on page 125.

To use Running Tool Tip

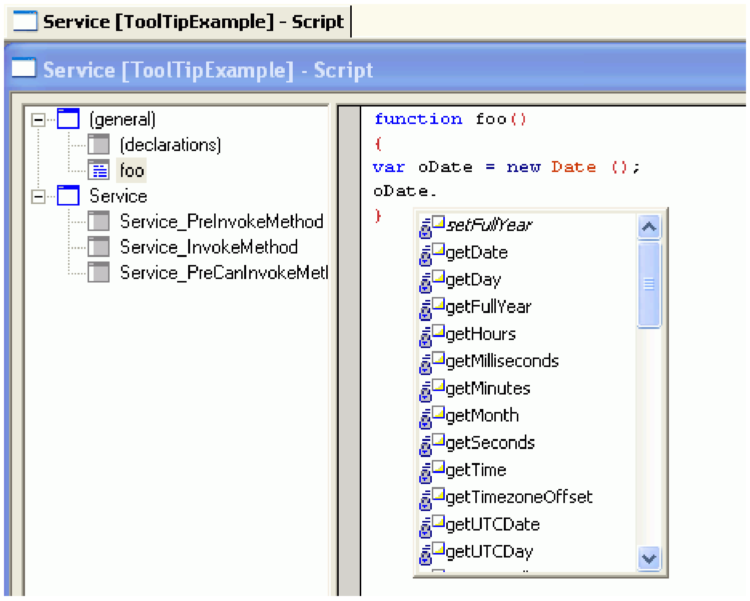
1 Open the Siebel Script Editor.

For more information, see "Opening the Siebel Script Editor" on page 122.

2 In the Script Editor window, enter a method name followed by an open parenthesis.

Siebel Tools displays the Running Tool Tip window. This window displays the method name and the method arguments. It uses italicized text to suggest an argument. Siebel Tools hides this window after you enter all the required method arguments.

Figure 4 includes the Running Tool Tip window.



```
function foo()
{
var oDate = new Date ();
oDate.
}
```

Figure 4. Running Tool Tip Window in the Siebel Script Editor

How Running Tool Tip Differs from Tool Tips in Script Assist

If you enter a function name followed by a left parenthesis in the Script Editor, then Script Assist or Running Tool Tip shows depending on the type of function you enter.

Running Tool Tip Window

If you enter a simple call expression, then Siebel Tools displays the Running Tool Tip window. A call expression allows you to send anything to the function according to the type of argument. For example, if you use the following code to enter a data object function, then Siebel CRM can send a value that you choose to this function:

```
Date.SetFullYear
```

For example, assume you enter the following code:

```
var oDate = new Date();
oDate.SetFullYear(
```

In this situation, Siebel Tools displays the following code in the Running Tool Tip window:

```
oDate.SetFullYear(year, month, date)
```

Script Assist Window

If you enter a collection function, then Siebel Tools displays the Script Assist window. This window includes a list of methods and properties that you can choose for an object. A *collection function* is a type of function that includes a finite set of values. You can send values to a collection function. For example, if you enter the following code, then Siebel Tools displays fields that are defined only for this business component:

```
BusComp.GetFieldValue
```

The following code is an example:

```
var bo = TheApplication().GetBusObject(
```

For more information about using functions and expressions in Siebel eScript, see *Siebel eScript Language Reference*.

Using Script Libraries with the ST eScript Engine

A *script library* is a part of the ST eScript Engine that allows you to call a business service method from a script. This topic describes how to make a custom business service method available to a business service script library and how to call this method in the script library. Using a script library is optional. Siebel CRM supports code you write before Siebel CRM version 8.0. For more information about script libraries, see *Siebel eScript Language Reference*.

Making a Custom Business Service Method Available in a Script Library

This topic describes how to make a custom business service method available in a script library.

To make a custom business service method available in a script library

1. Create a script for a business service method.
2. Validate the script.

   For more information, see "Validating Script Syntax" on page 134.
3. Save the script.
4. Make sure the External Use property of the business service contains a check mark.

   Siebel Tools adds the custom business service method to the script library.

Using a Script Library to Call a Custom Business Service Method

After you make a business service available for use in Siebel Tools, the following items apply:

- You can call a business service method of this business service from another script.
- The Script Assist window displays these business service methods. For more information, see "About Script Assist" on page 125.

To use a script library to call a custom business service method

1. Make sure the following options are enabled:
   - Enable Method Listing
   - Enable Auto Complete

   For more information about setting these options, see "Setting Options for the Siebel Script Editor" on page 121.
2. In the Siebel Script Editor, type the name of a business service object followed by a period (.).

   Siebel Tools displays the business service methods that are available for the business service.
3. Choose the business service method that you must add to the script.
4. Validate the script.

   For more information, see "Validating Script Syntax" on page 134.

Example of Using a Script Library

In this example, the mathService business service is enabled for use in Siebel Tools. It uses a business service method named square:

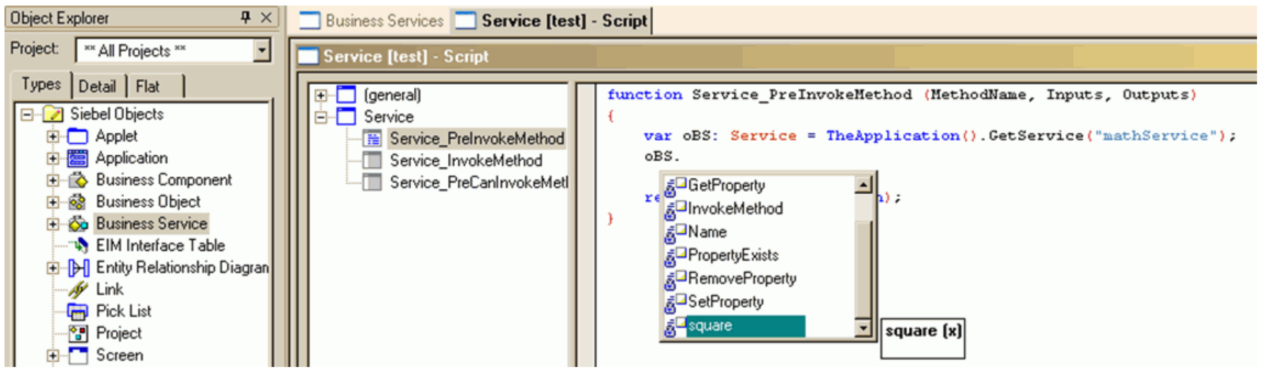
```
function square (x)
{
  return (x * x);
}
```

If you use a script library to call an argument, then the compiler examines the argument types to make sure the argument that your script calls is valid and is compatible. The following code calls a business service method named square:

```
var oBS: Service = TheApplication().GetService ("mathService");
var value = 10;
var square_value = oBS.square (value);
```

Figure 5 displays a list of the business service methods that are available for the mathService script library. This list displays the following code:

```
var square_value = oBS
```



```
function Service_PreInvokeMethod (MethodName, Inputs, Outputs)
{
    var oBS: Service = TheApplication().GetService("mathService");
    oBS.
```

Figure 5. Example of a Script Assist Window That Displays the Business Service Methods That Are Available for the MathService Script Library

Using the Siebel Debugger

This topic describes how to use the Siebel Debugger. It includes the following information:

- Overview of Using the Siebel Debugger on page 132
- Setting Debug Options on page 133
- Accessing the Siebel Debugger on page 134
- Validating Script Syntax on page 134
- Using Breakpoints on page 135
- Using the Calls Window on page 135
- Using the Watch Window While it Monitors a Script on page 136
- Tracing a Script on page 136

Overview of Using the Siebel Debugger

The Siebel Debugger can help you edit a script and remove an error from a script that you write in Siebel VB or Siebel eScript. It displays variables and their values in the Siebel Script Editor window and a diagnostic window. You can use it to slow down or suspend a script from running so that you can monitor the logical flow of your script and to examine the contents of variables. You can use the Siebel Debugger to do the following:

- **Set a breakpoint.** Allows you to use the Debugger so that you can examine the state of the program at this line. For more information, see "Using Breakpoints" on page 135.
- **Step over a line of code.** For example, if the current code line calls a subroutine, then the Debugger skips this line and stops at the next line. It does not call the subroutine.
- **Step into a subroutine.** Runs one line of code according to the following:
  - **The current line calls a subroutine or function.** The Debugger stops at the first line of the subroutine or function that the current line calls.
  - **The current line does not call a subroutine or function.** The Debugger stops at the next code line.
- **Examine the value of a variable.** The Siebel Debugger includes a Watch window that displays variables and their values. You can examine these values while a script runs.

Setting Debug Options

This topic describes how to set debug options.

To set debug options

1. In Siebel Tools, click the View menu, and then click Options.
2. In the Development Tools Options dialog box, click the Debug tab, and then set the options.

   The following table includes example settings for the Debug tab. For more information, see "Development Options for Debugging" on page 244.

Option	Value
Executable	siebel.exe
CFG File	c:\siebel\bin\w32u\enu\uagent.cfg For more information, see "Specifying the Data Source That Siebel Tools Uses" on page 15.
Working Directory	c:\siebel\bin\w32u
Arguments	You can use the following arguments: ■ **/h**. ■ **/l.** Sets the language that Siebel Tools uses for the debug session. You can include these arguments on the same line and in any order. For example, you can use one of the following for ENU: ■ `/h /l ENU` ■ `/l ENU /h`
User Name	sadmin

Option	Value
Password	******
Data Source	Sample
Enable Profiler	Contains a check mark.

3 Click the Scripting tab.

4 In the Debugging section, set the options, and then click OK.

   For more information, see "Development Options for Visualization Views" on page 242.

Accessing the Siebel Debugger

You can access the Siebel Debugger.

To access the Siebel Debugger

- Do one of the following:
  - Open the Siebel Script Editor, set a breakpoint in your script, and then click the Start button.

    For more information, see "Using Breakpoints" on page 135.
  - Run a script that includes a run-time error.

    If a script includes a run-time error, then Siebel Tools suspends the script, starts the Debugger, and highlights the line that includes the error. An unhandled Siebel VB or eScript error is an example of a run-time error.

Validating Script Syntax

The debugger includes a syntax checker that you can use to make sure your script compiles properly.

CAUTION: The Check Syntax feature identifies only syntax errors and errors that occur if an object or variable is not initialized. It does not identify other types of errors and it cannot trap an error that might cause a run-time error. It examines only script that is attached to the current active object. If an error exists in another script, then you cannot compile the repository.

To validate script syntax

1 Open the Siebel Script Editor, click the Debug menu, and then click Check Syntax.

   Siebel Tools does a test compile, and then does one of the following:
   - **It finds no error.** It displays nothing.
   - **It finds an error.** It displays a dialog box that describes the error.

2. If the script includes an error, then click Go to Line in the dialog box.

   Siebel Tools positions the cursor on the script line that causes the error and highlights this line.

3. Correct the code and repeat Step 1.

   If the syntax of the line that you modified is correct, then the dialog box displays the next error it finds, if any.

4. Repeat Step 1 and Step 2 until you correct all errors.
5. Click the File menu, and then click Save.
6. Compile the repository.

   For more information, see "Compiling Your Modifications" on page 179.

7. When the compile finishes, click Start to restart Siebel CRM.

Using Breakpoints

A *breakpoint* is a marker in a line of code that stops code from running at this line. It allows you to use the Debugger so that you can examine the state of the program at this line.

To use breakpoints

- Do one of the following:
  - **You are editing a script.** Place the cursor on the line of code where you must set a breakpoint, and then press F9.
  - **You are debugging script.** Click a line of code.

  If a breakpoint does not already exist on this line, then Siebel Tools adds a breakpoint. If a breakpoint already exists on this line, then Siebel Tools removes it.

Using the Calls Window

The Calls window includes a list of subroutine and function calls that Siebel Tools runs prior to the current code line.

To use the Calls window

1. Open the Siebel Script Editor.
2. Click the Calls button in the Debugger toolbar while you run the Debugger.

   Each line in the Calls window represents a subroutine that Siebel Tools entered but did not complete. If you choose an entry in this list, then the following occurs:

   - The Debugger window displays the line of code that makes this call.

- The Variable window displays the variables that are associated with the procedure that makes the call.

Using the Watch Window While it Monitors a Script

The Watch window displays variables and variable values. You can use it to monitor these values while a script or workflow process runs. In Siebel CRM version 8.0, the Watch window supports the following variable types:

- Local variables
- Global variables
- Shared global variables
- Application variables
- Persistent profile properties
- Dynamic profile properties

For a detailed example that uses the Watch window, see the information about defining a workflow process that closes obsolete service requests in *Siebel Business Process Framework: Workflow Guide*.

To use the Watch window while it monitors a script

1. Attach a script to an object, and then compile this object.

   For more information, see “Compiling Your Modifications” on page 179.

2. Start the Siebel client from the Siebel Debugger.

   You can press F5 or click the Start icon on the Debug menu. Make sure the /h argument is set in the Debug options. For more information, see “Setting Debug Options” on page 133.

3. Display the Watch window.

   You can press SHIFT+F9 or click the Watch button (glasses icon) on the Debug toolbar.

Tracing a Script

You can run a trace on an allocation, event, or SQL command. You can start a trace for a user account, such as your development team. The Siebel Server saves the trace information to a log file. Tracing a script is not the same as tracing a file. For more information about tracing a file, see the information that describe the Trace, TraceOn, and TraceOff methods in *Siebel Object Interfaces Reference*.

To trace a script

1. Log in to the Siebel client, navigate to the Administration - Server Management screen, and then the Components view.

2 Choose the component that you must trace.

3 Click the Component Event Configuration tab.

4 Locate the Object Manager Extension Language Log event.

   If this event does not exist, then the component you chose in Step 2 does not support logging. Most components support logging, but some do not. If necessary, repeat Step 2 but choose another component.

5 Set the Log Level to 1.

   To disable logging when you are done, you can set the Log Level to 0 (zero).

6 Click the Component Parameters tab.

7 (Optional) To display only the script tracing parameters, enter a query. Use values from the following table.

Field	Value
Parameter Alias	Trace*
Subsystem	Object Manager

8 Set one or more tracing parameters. Use values from the following table.

Object to Trace	Alias	Description
Allocations	TraceAlloc	Enter `0` (zero) to disable logging. Enter `1` to enable logging.
Events	TraceEvents	
SQL Commands	TraceSql	
Users	TraceUser	Enter a comma-separated list of user names. Do not use spaces. For example, you can enter the following: `sadmin,mmasters,hkim,cconnors`

To set tracing parameters, enter a value depending on when the modification must take affect:

- **Immediately.** You modify values in the Current Value column.
- **After a restart.** You modify values in the Value on Restart column.

Example of a Trace

The following code is an example of a trace that Siebel Tools creates after the Business Service Simulator runs:

```
ObjMgrExtLangLogObjMgrExtLangLog000000080475f1578:02007-12-12 07:33:45[User: ] SQLBIND, 90, 1, Y.
ObjMgrExtLangLogObjMgrExtLangLog000000080475f1578:02007-12-12 07:33:45[User: ] COMPILE, END, <unknown>.
ObjMgrExtLangLogObjMgrExtLangLog000000080475f1578:02007-12-12 07:33:45[User: ] EVENT, BEGIN, Service
[sam], Service_PreInvokeMethod. ObjMgrExtLangLogObjMgrExtLangLog000000080475f1578:02007-12-12 07:33:45
[User: ] EVENT, END, Service [sam], Service_PreInvokeMethod. ObjMgrExtLangLogObjMgrExtLangLog0
00000080475f1578:02007-12-12 07:34:39[User: ] SQLSTMT, 91,
```

Example of a Detailed Trace

The following code is an example of a trace that Siebel Tools creates if the log level is set to high. For brevity, this book includes only part of this log file:

```
2021 2007-12-12 07:46:29 2007-12-12 07:56:46 -0700 000003fd 001 003f 0001 09 SCCObjMgr_enu 11534365 5452
780 d:\21022\ses\siebsrvr\log\SCCObjMgr_enu_0011_11534365.log 8.1 [21022] ENU

ObjMgrLogInfo3000000b9475f1578:02007-12-12 07:46:29(modpref.cpp (949)) SBL-DAT-50803: SharedFileReader:
D:\fs\userpref\SADMIN&Siebel Universal Agent.spf_SDCHS21N625_5452_780: File succesfully opened.

ObjMgrLogInfo3000000b9475f1578:02007-12-12 07:46:29(modpref.cpp (949)) SBL-DAT-50804: LoadPreferences:
D:\fs\userpref\SADMIN&Siebel Universal Agent.spf: Preferences succesfully loaded.

ObjMgrExtLangLogObjMgrExtLangLog0000000b9475f1578:02007-12-12 07:46:29[User: ] SQLSTMT, 1, SELECT, SELECT

T1.CONFLICT_ID,
T1.DB_LAST_UPD_SRC,
CONVERT (VARCHAR (10),T1.DB_LAST_UPD, 101) + ' ' + CONVERT (VARCHAR (10),T1.DB_LAST_UPD, 8),
CONVERT (VARCHAR (10),T1.LAST_UPD, 101) + ' ' + CONVERT (VARCHAR (10),T1.LAST_UPD, 8),
CONVERT (VARCHAR (10),T1.CREATED, 101) + ' ' + CONVERT (VARCHAR (10),T1.CREATED, 8),
T1.LAST_UPD_BY,
T1.CREATED_BY,
T1.MODIFICATION_NUM,
T1.ROW_ID,
T1.BUILD_NUMBER,
T1.CURR_WEBFILE_VER,
T1.ENTERPRISE_NAME,
T1.PREV_WEBFILE_VER,
T1.RECORD_INFO_TEXT,
T1.REC_IDENTIFIER
FROM
   dbo.S_UIF_WEB_FILE T1
WHERE
   (T1.REC_IDENTIFIER = ?).

ObjMgrExtLangLogObjMgrExtLangLog0000000b9475f1578:02007-12-12 07:46:29[User: ] SQLBIND, 1, 1,
16:sdchs21n625:4330siebel.
```

Declaring Functions and Procedures in Siebel VB

The Siebel Compiler compiles Siebel VB functions and procedures in alphabetical order for each object definition. If a function or procedure calls another function or procedure that is not defined, then the compiler creates an error message that is similar to the following:

```
function_name Is An Unknown Function
```

To avoid this error, you must use the Declare statement to declare the function or procedure in the general declarations section. Siebel eScript does not require you to declare these functions. For more information, see *Siebel VB Language Reference*.

Using the Script Profiler

This topic describes how to use the Script Profiler. It includes the following information:

- Overview of the Script Profiler on page 139
- Enabling the Script Profiler and Line Profiler on page 141
- Running the Script Profiler on page 142

Overview of the Script Profiler

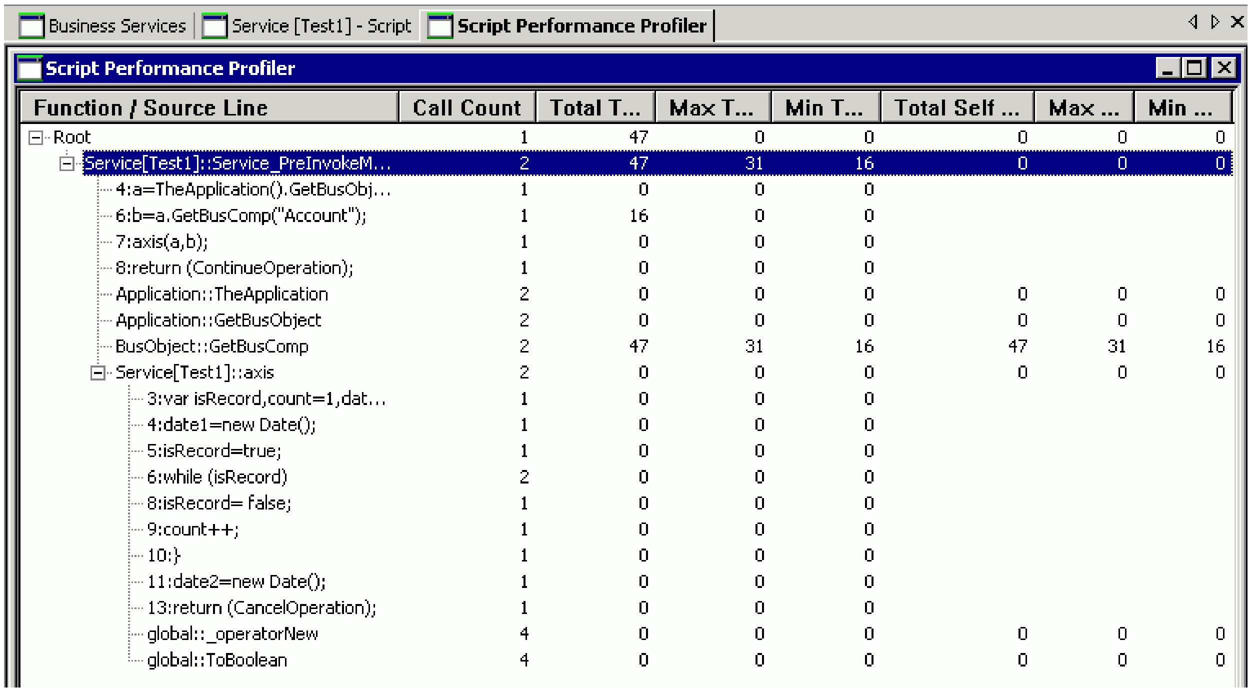
The *Siebel Script Performance Profiler* is a part of the ST eScript Engine that allows you to observe and monitor the performance of a script. For brevity, this book refers to the Siebel Script Performance Profiler as the Script Profiler.

If you start Siebel CRM in debug mode from Siebel Tools, then the Script Profiler gathers and displays data for all executable scripts. It displays data in the Script Performance Profiler window in Siebel Tools and updates this data if a script runs. You can use this data to monitor script performance, identify performance bottlenecks, and compare performance with previous script runs. The Script Profiler allows you to do the following:

- **Use the Call Tree view.** Displays profile data as a tree of function calls. For more information, see "Example of the Script Performance Profiler Window" on page 139.
- **View function profile and line profile.** Line profile data includes the call count and total time spent for each line that runs in a function. Line profile information is available only for a compiled script and is not available for a line that does not run.
- **Save profile data to file.** You can save profile data that the Script Performance Profiler window shows to a text file.
- **Use the Siebel Script Debugger and the Siebel Script Profiler.** You can use the Script Profiler and the Script Debugger at the same time. Profile data is consistent even if the function uses Debugger functionality, such as a breakpoint.
- **View the script source.** Siebel Tools opens the objects that a script references in the Script Editor window. You can double-click a function name or line number to view the script from the Script Performance Profiler window. The View Source option is available only for a compiled script. It is not available for a runtime script.

Example of the Script Performance Profiler Window

Figure 6 displays an example of the Script Performance Profiler window. It allows you to examine script performance. It displays a line number for each code line, total time spent to run that line, the number of times Siebel CRM runs the line, and so on.



Function / Source Line	Call Count	Total T...	Max T...	Min T...	Total Self ...	Max ...	Min ...
Root	1	47	0	0	0	0	0
Service[Test1]::Service_PreInvokeM...	2	47	31	16	0	0	0
4:a=TheApplication().GetBusObj...	1	0	0	0			
6:b=a.GetBusComp("Account");	1	16	0	0			
7:axis(a,b);	1	0	0	0			
8:return (ContinueOperation);	1	0	0	0			
Application::TheApplication	2	0	0	0	0	0	0
Application::GetBusObject	2	0	0	0	0	0	0
BusObject::GetBusComp	2	47	31	16	47	31	16
Service[Test1]::axis	2	0	0	0	0	0	0
3:var isRecord,count=1,dat...	1	0	0	0			
4:date1=new Date();	1	0	0	0			
5:isRecord=true;	1	0	0	0			
6:while (isRecord)	2	0	0	0			
8:isRecord= false;	1	0	0	0			
9:count++;	1	0	0	0			
10:}	1	0	0	0			
11:date2=new Date();	1	0	0	0			
13:return (CancelOperation);	1	0	0	0			
global::_operatorNew	4	0	0	0	0	0	0
global::ToBoolean	4	0	0	0	0	0	0

Figure 6. Example of the Script Performance Profiler Window

Description of the Columns in the Script Performance Profiler Window

Table 13 describes the columns that the Performance Profiler window includes. For the Source Line, Call Count, and Total Time columns, if line profiling is enabled for a function, then the line profile displays as child nodes of a function in the Call Tree view and in the Flat Profile view.

Table 13. Description of Columns in the Script Performance Profiler Window

Column	Description
Function/Source Line	Function name and the name of the object that contains the function. For example: `Service [ATP Check]::service_preinvokeMethod` and `BusComp[Account]::foo`
Call Count	The number of times Siebel CRM calls a function. You can use one of the following views: ■ **Flat Profile view.** Displays the total number of times Siebel CRM calls the function. ■ **Call Tree view.** Displays the number of times Siebel CRM calls the function in the current position in the call tree. It displays profile data as a tree of function calls. A node represents each function in a call sequence. You can drill down into each node to examine a subtree. You can use the Expand All option in the Profiler toolbar to expand all nodes.
Total Time	Total milliseconds spent in this function and in nested functions.
Max Time	Maximum milliseconds spent in this function and in nested functions.
Min Time	Minimum milliseconds spent in this function and in nested functions.
Total Self Time	The total milliseconds spent in the current function, not including time spent in the subtree of this function.
Max Self Time	The maximum milliseconds spent in the current function, not including time spent in the subtree of this function.
Min Self Time	The minimum milliseconds spent in the current function, not including time spent in the subtree of this function.

Enabling the Script Profiler and Line Profiler

The Script Profiler and the line profiler are not enabled after you install Siebel Tools. You must enable them.

To enable the Script Profiler

1 Make sure your Siebel Tools environment is currently connected to the database that a Siebel application uses, and that this application is opened in debug mode.

You can use the Script Profiler only if your Siebel Tools environment is currently connected to the database that a Siebel application uses, and if this application is opened in debug mode.

2 Make sure the ST eScript Engine is enabled.

   You can use the Script Profiler only if the ST eScript Engine is enabled. For more information, see "Enabling the ST eScript Engine" on page 127.

3 Set the debug options on the Debug tab of the Development Tools Options dialog box.

   Make sure you complete options in the Run-time Start Up Information section and the Login Information section. For more information, see "Setting Debug Options" on page 133.

4 Make sure the Enable Profiler check box on the Debug tab of the Development Tools Options dialog box contains a check mark.

   The Enable Profiler option is available only if the ST eScript Engine is enabled. Siebel Tools enables the Script Profiler the next time you use the debug mode in Siebel Tools to open a Siebel application.

5 Enable line profiling:

   a Click Set Line Profile Rules.

      The Set Line Profile Rules button resides in the Profiler Start Up Options section on the Debug tab of the Development Tools Options dialog box.

   b In the Line Profile Rules dialog box, add a rule using information from the following table, and then Click Add.

Item	Description
Object Type	Choose the object type. Only scriptable objects are available. For more information, see "Object Types You Can Script" on page 120.
Object Name	Enter the name of an object. You can include the entire object name or use wildcard operators.
Function	Enter the name of a function. You can enter the entire function name or use wildcard operators. For example, you can enter `Service_PreInvokeMethod` or `Service*`.

   c Repeat Step b for each rule you must add.

   d Click Ok to exit the Line Profile Rules dialog box.

6 Click OK to exit the Development Tools Options dialog box.

   The settings for the line profile rule remain active only for the current Siebel Tools session.

   To disable line profiling, make sure the Enable Profiler check box on the Debug tab of the Development Tools Options dialog box does not contain a check mark.

Running the Script Profiler

This topic describes how to run the Script Profiler. For more information, see "Overview of the Script Profiler" on page 139.

To run the Script Profiler

1. Log in to Siebel Tools.

2. Make sure the Script Profiler is enabled.

   For more information, see "Enabling the Script Profiler and Line Profiler" on page 141.

3. Open the Siebel Script Editor for a scriptable object, and then compile the script to a new repository.

   Make sure you compile the scripted objects or the projects that these objects reference to the repository that the Siebel client uses. If you do not compile these objects, then the Script Profiler does not display them. For more information, see "Object Types You Can Script" on page 120 and "Compiling Your Modifications" on page 179.

4. Open the configuration file that your Siebel application uses and replace the default siebel.srf file with the file that you compiled in Step 3.

   The Script Profiler is now enabled for runtime sessions and you can open the Siebel application from Siebel Tools in debug mode.

   For more information, see "Specifying the Data Source That Siebel Tools Uses" on page 15.

5. In Siebel Tools, click the Debug menu, and then click Start.

   This step opens the Siebel application in debug mode. If Siebel CRM prompts you for a login, then enter the user name and password, and then click Run. For example, to run a script for the Business Service Simulator, you query for the Service Name and Method Name, and then click Run.

6. In Siebel Tools, click the View menu, Profiler, and then click Call Tree.

   Siebel Tools displays the functions that Siebel CRM calls when it runs a script as a hierarchy in the Script Performance Profiler window. It displays the functions that it calls from other functions as child objects. For an example, see "Example of the Script Performance Profiler Window" on page 139.

7. In the Script Performance Profiler window, you can do the following:

   - Navigate between nodes to view the different parameters for a function.

   - Right-click a node, and then choose a menu item. For more information, see "Navigating in the Script Performance Profiler Window" on page 144.

8. (Optional) To set or reset line profile rules for a function, do one of the following:

   - Right-click a function node, and then click Enable Line Profiling or Disable Line Profiling. If you modify a line profile rule, then this modification applies to future calls to the function that this rule references. For more information, see "Enabling the Script Profiler and Line Profiler" on page 141.

   - Use the Debug tab in the Development Tools Options dialog box. For more information, see "Setting Debug Options" on page 133.

9 Continue monitoring your script.

   For example, assume you must test custom script on a business service. In the Siebel client, you can navigate to the Business Service Simulator to run a business service, and then navigate back to the Profiler window to examine the profile data that Siebel CRM logs while the script runs.

Navigating in the Script Performance Profiler Window

You can right-click in the Script Performance Profiler window, and then click one of the following menu items. The menu items that Siebel Tools enables depends on the script. Some of these items might not be available for all scripts:

- **Expand Node**. Expands the function node.
- **Expand All**. Expands the entire tree of a function node.
- **Export**. Exports profile data to a text file that you specify.

  You can use Expand All and Export in the Profiler toolbar. For more information about the Profiler toolbar, see "Example of the Script Performance Profiler Window" on page 139.
- **View Source**. Navigates to the current function in the script. If a Script Editor window is not open for this function, then Siebel Tools opens a new editor window and places the cursor at the beginning of this function.
- **Enable Line Profiler.** Enables a function for line profiling. If a function is not in the list of functions chosen for line profiling, then you can click Enable Line Profiler to enable it.
- **Disable Line Profiler.** Disables a function for line profiling.

7 Localizing Strings and Locale Data

This chapter describes how to localize strings and locale data. It includes the following topics:

- Configuring Symbolic Strings on page 145
- Converting Symbolic Strings on page 153
- Configuring Locale Data on page 161
- Using the Locale Management Utility on page 165
- Using Advanced Compile to Localize a Repository on page 175

Configuring Symbolic Strings

This topic describes how to configure symbolic strings. It includes the following information:

- Overview of Configuring Symbolic Strings on page 145
- Modifying the Configuration File to Support Symbolic Strings on page 147
- Creating a Project for Your Symbolic Strings on page 147
- Modifying a Predefined Symbolic String on page 148
- Creating a New Symbolic String on page 148
- Modifying a Symbolic String to Globally Update Display Values on page 149
- Setting a Symbolic String Reference on page 150
- Entering a String Override on page 152

Overview of Configuring Symbolic Strings

A *symbolic string* is an object that you can use to store the value of a string. It allows you to define a string one time, and then reference it from multiple objects. An object can reference this symbolic string to get a literal value, and then display it as text in the Siebel client. For example, an applet can reference a symbolic string to get the text it shows in a control caption, a list column display name, or an applet title.

A *translatable string* is a type of string that Siebel CRM can translate to another language. For example, it can translate the text string that defines a control label from English to French.

The symbolic string centralizes all the strings in the repository. It includes strings in English and all other languages. It includes the following advantages:

- Reduces redundancy because many objects can reference one symbolic string
- Results in a more consistent user interface

- Simplifies maintenance because you are required to maintain only one string for a word or phrase
- Eliminates duplicate translations of the same word
- Reduces translation costs

Siebel CRM cannot translate a nontranslatable property to another language. For example, the HTML Sequence property, HTML Height property, and HTML Width property are nontranslatable properties.

If you compile to the repository, then Siebel Tools uses the current language mode that you set for Siebel Tools. It uses this language mode to choose from one of several translations for a set of symbolic string locale records. For more information, see "Compiling Your Modifications" on page 179.

Symbolic Strings in the Object Hierarchy

The Symbolic String object type is a top-level object type. it includes the child Symbolic String Locale object type. Each symbolic string represents a word or phrase that is language independent. For example:

Account

Siebel CRM stores all translations of this word or phrase, including English, as a symbolic string locale. For information about top-level object types, see "Displaying Object Types in the Object Explorer" on page 26.

Siebel Tools stores data for a symbolic string in the S_SYM_STR (symbolic strings) table. It stores data for a symbolic string locale in the S_SYM_STR_INTL (Repository Symbolic String Locale) table. Objects that reference symbolic strings store foreign key references to these strings in the S_SYM_STR table.

Symbolic strings do not include other types of strings that Siebel CRM typically includes as seed data, such as LOVs, error messages, or predefined queries. For more information about localizing these types of strings, see "Fixing Orphaned String References After an Upgrade" on page 164.

How Siebel Tools Determines a Translatable String

Siebel Tools does the following to compile an object property that displays a translatable string, such as the Title property of an applet:

- If a value exists in the string language that the compile uses, then it compiles this string override value.
- If the string override field for the current language mode that you set in Siebel Tools does not include a string value, then Siebel Tools uses the current language mode and the String Value property of the associated symbolic string locale to determine the value.

In most situations, a string override does not exist. For more information, see "Entering a String Override" on page 152.

How Some Early Releases Store Translatable Strings

Some early releases of Siebel Tools store translatable strings in the locale objects of a parent object type. For example, each applet includes a set of child locale records that define the text for the applet title that Siebel CRM shows in the Siebel client.

Modifying the Configuration File to Support Symbolic Strings

This topic describes how to modify the Siebel Tools configuration file to support symbolic strings.

To modify the configuration file to support symbolic strings

1. Open the tools.cfg file.

   For more information, see "About the Configuration Files" on page 17.

2. In the Siebel section, set the EnableToolsConstrain parameter to one of the following values:

   - **TRUE.** Constrained mode. Siebel Tools requires you to choose a translatable string from the list of available string references. You cannot override the string reference. For example, you cannot enter a value for a string override field. You cannot create a new symbolic string reference.
   - **FALSE.** Unconstrained mode. Siebel Tools does not require you to choose a translatable string from the list of string references. You can override the string reference. For example, you can enter a value in a string override field. You can create a new symbolic string reference.

3. (Optional) Set the SymStrPrefix parameter to a custom value.

   To distinguish between a predefined symbolic string and a custom symbolic string that you create, Siebel Tools sets the prefix for any new symbolic string you create to X_, by default. You can modify this value. For example, if you set this parameter to `SymStrPrefix=X_NewString`, then any new symbolic string you create includes the `X_NewString` prefix. For more information, see "Modifying a Predefined Symbolic String" on page 148 and "About Predefined Objects" on page 24.

Creating a Project for Your Symbolic Strings

The Symbolic String project is very large. Checking in or checking out the entire project might consume a significant amount of time. It is recommended that you do not check out the entire Symbolic String project. Instead, you can create a new project and set the Project property for any new symbolic string you create to reference this project. For more information about projects, see Chapter 4, "Checking Out and Checking In Projects and Objects."

To create a project for your symbolic strings

1. Log in to Siebel Tools.
2. In the Object Explorer, click Project.
3. In the Projects list, create a new project.

   For example, create a new project named CompanyXYZ New Symbolic String.

4. Make sure the Locked property contains a check mark.

Modifying a Predefined Symbolic String

You can modify a *predefined symbolic string*, which is a string that comes predefined with Siebel CRM. It includes a SBL_ prefix in the Name property. For more information, see "About Predefined Objects" on page 24.

CAUTION: It is recommended that you do not modify a predefined symbolic string. Instead, it is recommended that you create a new symbolic string that includes the text you require. It is recommended that you do not modify the display value of a predefined symbolic string. Siebel CRM might use this display value throughout the Siebel client. For a monolingual deployment, you risk modifying text in the Siebel client that you do not intend to modify. For a multilingual deployment, you risk breaking the relationships between display values for different languages. For more information, see "Creating a New Symbolic String" on page 148.

To modify a predefined symbolic string

1. Log in to Siebel Tools and display the Symbolic String object type.

   For more information, see "Displaying Object Types in the Object Explorer" on page 26.

2. In the Object Explorer, click Symbolic String.

3. In the Object List Editor, locate the object you must modify.

   For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

4. Set the Project property of the string you locate in Step 3 to the project you create in "Creating a Project for Your Symbolic Strings" on page 147.

   To modify multiple records, see "Using the Properties Window to Modify Objects" on page 108.

Creating a New Symbolic String

You create a new symbolic string.

To create a new symbolic string

1. Make sure the EnableToolsConstrain parameter allows you to create a new symbolic string.

   For more information, see "Modifying the Configuration File to Support Symbolic Strings" on page 147.

2. Log in to Siebel Tools and display the Symbolic String object type.

   For more information, see "Displaying Object Types in the Object Explorer" on page 26.

3. In the Object Explorer, click Symbolic String.

4 In the Symbolic Strings list, create a new record. Use values from the following table.

Property	Description
Name	Enter a unique name for this symbolic string. For more information about the prefix that Siebel Tools adds to the name of any custom symbolic string you create, see "Modifying the Configuration File to Support Symbolic Strings" on page 147.
String Value	Enter the text that this symbolic string shows. In some situations, this value might be a value that Siebel Tools determines according to the current language mode and the String Value property of the child symbolic string locale object. For more information, see "Setting the Language Mode" on page 19. Siebel Tools truncates any trailing spaces you enter, including full width spaces in Japanese.
Definition	Enter text that describes the purpose of this symbolic string.
Project	Set this property to the project you create in "Creating a Project for Your Symbolic Strings" on page 147.

Modifying a Symbolic String to Globally Update Display Values

You can configure Siebel CRM to make global modifications to the values it shows in the Siebel client. For example, this feature can be useful in the following situations:

- You must modify all instances of the word Account that Siebel CRM shows in the Siebel client to Customer.
- You must deploy Siebel CRM specific to an industry in a locale other than English. The text strings that it shows in the Siebel client might not be appropriate for this industry.

To modify a symbolic string to globally update display values

1 Set the language mode.

   For more information, see "Enabling Language Override" on page 20.

2 In the Object Explorer, click Symbolic String.

3 In the Object List Editor, locate the object you must modify.

   For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

4 Modify the value of the String Value property.

5 Compile the projects that include the objects that reference this symbolic string.

   For more information, see "Compiling Your Modifications" on page 179.

Setting a Symbolic String Reference

A *symbolic string reference* is a reference to a symbolic string. It allows you to set the text that Siebel CRM shows in the Siebel client for an object property, such as an applet title or application display name. This topic describes two different ways that you can set this text.

Using the String Reference Property to Set a Symbolic String Reference

This topic describes how to use the String Reference property to set a symbolic string reference.

To use the String Reference property to set a symbolic string reference

1. Make sure the EnableToolsConstrain parameter is set to FALSE.

   For more information, see "Modifying the Configuration File to Support Symbolic Strings" on page 147.

2. In the Object List Editor, locate the object you must modify.

   For example, locate the Account List Applet. For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

3. In the Title - String Reference property, click the drop-down arrow.

   This property varies according to object type. For more information, see "Properties That Store a Symbolic String" on page 150.

4. In the Title - String Reference dialog box, locate the string reference you require, and then click Pick.

   Siebel Tools enters a value in the Title property according to the current language mode and the value in the String Value property of the child symbolic string locale.

5. If no existing symbolic string meets your requirements, then you can enter a string override.

   For more information, see "Entering a String Override" on page 152.

Properties That Store a Symbolic String

Table 14 describes the properties that store a symbolic string. The property varies according to object type. Siebel Tools displays these properties only in the Object List Editor. It does not display them in the Properties window.

Table 14. Properties That Store a Symbolic String

Object Type	Label for the String Reference Property
Web Page	Title - String Reference
Applet	Title - String Reference
Screen	Viewbar Text - String Reference

Table 14. Properties That Store a Symbolic String

Object Type	Label for the String Reference Property
View	A view includes the following properties that reference a symbolic string: ■ Title - String Reference ■ Thread Title - String Reference
Application	Display Name - String Reference

Entering a Value to Set a Symbolic String Reference

This topic describes how to enter a value to set a symbolic string reference.

To enter a value to set a symbolic string reference

1. Make sure the EnableToolsConstrain parameter is set to FALSE.

   For more information, see "Modifying the Configuration File to Support Symbolic Strings" on page 147.

2. In the Object List Editor, locate the object you must modify.

   For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

   For example, locate the Account List Applet.

3. In the Title property, enter the text that Siebel CRM must show in the Siebel client.

   This property varies according to object type. For more information, see "Properties That Store the Display Text" on page 151.

4. Tab out of the field.

   Siebel Tools searches for a string reference that includes a value in the String Value property that matches the value you enter, and then does one of the following depending on the result of the search:

   - **A unique match exists.** Siebel Tools enters the symbolic string reference it finds into the String Reference property. It enters this value according to the current language mode and the value that the String Value property of the child symbolic string locale contains.

   - **Multiple matches exist or no match exists.** Siebel Tools displays an error dialog box. You can click OK to close the dialog box, and then use the String Reference Property. For more information, see "Using the String Reference Property to Set a Symbolic String Reference" on page 150.

Properties That Store the Display Text

Table 15 describes the properties that store the display text. The property varies according to object type. Siebel Tools displays these properties in the Object List Editor and in the Properties window.

Table 15. Properties That Store the Display Text

Object Type	Property That Stores the Display Text
Web Page	Title
Applet	Title
Screen	Viewbar Text
View	A view includes the following properties that include display text: ■ Title ■ Thread Title
Application	Display Name

Entering a String Override

If you cannot find a symbolic string that meets your requirements, then you can override the symbolic string that Siebel CRM uses. Table 15 on page 152 lists the properties that store a translatable string. Siebel CRM includes a corresponding String Override property for each of these properties that store a translatable string. For example, the Title – String Override property is the corresponding property for the Title property of an applet. You can use this override property to enter a custom string.

To enter a string override

1. Make sure the EnableToolsConstrain parameter is set to FALSE.

   For more information, see “Modifying the Configuration File to Support Symbolic Strings” on page 147.

2. In the Object List Editor, locate the object you must modify.

   For example, locate the Account List Applet. For more information, see “Locating and Modifying Object Definitions in the Object List Editor” on page 24.

3. In the Title-String Override property, enter the string you require, and then tab out of this field.

   This property varies according to object type. For more information, see “Properties That Store the String Override” on page 153.

   For more information, see “How Siebel Tools Stores the Override Value You Enter” on page 153.

How Siebel Tools Stores the Override Value You Enter

If you modify the value in the Title-String Override property, then Siebel Tools also modifies the value in the Title property to the custom value you enter. It stores this value in a locale object that is a child of the parent you modify. For example, if you use the Title-String Override property to modify the display name for the Account List Applet to My Big Accounts, then Siebel Tools adds a child locale object to the Account List Applet. The Title property of this locale object includes the custom value you enter, such as My Big Accounts. To view this locale object, you must display it in the Object Explorer. For more information, see "Displaying Object Types in the Object Explorer" on page 26.

To set this value, Siebel Tools uses the current language mode and stores it as a language-dependent value. This value does not affect other references to symbolic strings.

Properties That Store the String Override

Table 16 describes the properties that store the string override. This property varies according to object type. Siebel Tools displays these properties in the Object List Editor and in the Properties window.

Table 16. Properties That Store the String Override

Object Type	Property That Stores the String Override
Web Page	Title - String Override
Applet	Title - String Override
Screen	Viewbar Text - String Override
View	A view includes the following properties that store a string override: ■ Title - String Override ■ Thread Title - String Override
Application	Display Name - String Override

Converting Symbolic Strings

This topic describes how to convert symbolic strings. It includes the following information:

- Consolidating Symbolic Strings on page 158
- Using a Batch File to Convert Strings on page 160

You can use Siebel Tools to convert a translatable string to a symbolic string. Siebel CRM stores this translatable string as a child locale record of a top-level object type. It stores symbolic strings in a single table. Converting a translatable string might be useful in the following situations:

- You upgrade to a new Siebel CRM version and you must convert custom translatable strings to symbolic strings.

- You use string overrides to store text strings and periodically must convert them to symbolic strings.

Converting to symbolic strings reduces the size of the repository, simplifies translations, and helps to make sure that the text that Siebel CRM shows in the Siebel client is consistent. It also requires that development work is finished.

To convert symbolic strings

1. Review the caution information described in "Caution About Converting and Consolidating Symbolic Strings" on page 156.
2. Prepare to do the conversion:
   a. Back up the Siebel database and repository.
   b. Verify the Siebel Tools configuration file:
      - Make sure the DataSource parameter references the correct Siebel database. The conversion utility uses this Siebel database.
      - Make sure the EnableToolsConstrain parameter is set to FALSE and that the SymStrPrefix parameter is set to the appropriate prefix.

      For more information, see "Modifying the Configuration File to Support Symbolic Strings" on page 147.
   c. Make sure all projects are unlocked.

      For more information about projects, see Chapter 4, "Checking Out and Checking In Projects and Objects."
   d. Inform other developers not to log on to the development environment while the conversion runs.
3. Log in to Siebel Tools and display the Attribute object type.

   For more information, see "Displaying Object Types in the Object Explorer" on page 26.
4. Identify the object types you must convert:
   a. In the Object Explorer, click the Flat Tab, and then click Attribute.
   b. In the Attributes list, query the Name property for the following string:

      ```
      *String Reference*
      ```

      The Parent Type property displays the complete set of object types that you must convert. If an object type includes more than one property that references a symbolic string, then you can run the conversion for this object type only one time.
5. Convert locale strings to symbolic strings. You must convert the locale strings for the object types you identify in Step 4. You can use the conversion utility to do this conversion:
   a. Open a command line.
   b. Navigate to the following folder:

      SIEBEL_TOOLS_ROOT\BIN

c Enter the following command:

```
consoleapp config_file app_lang user_Id password "business_service"
"business_service_method: parameters"
```

Use values from the following table. For more information, see "Running the Console Application Executable" on page 156.

Parameter	Description
config_file	Specify the Siebel Tools configuration file, such as tools.cfg. Use the default data source. For more information see "About the Configuration Files" on page 17 and "Modifying the Configuration File to Support Symbolic Strings" on page 147.
app_lang	Specify the application language, such as ENU.
user_Id	Specify the user Id.
password	Specify the password.
business_service	Specify the String Conversion business service.
business_service_method	Specify the name of the business service method.
parameters	Specify the input parameters to the business service method. For more information, see "Separating Conversion Files into Smaller Files" on page 157.

6 Export candidates for one of the object types you identified in Step 4. Enter the following command:

```
consoleapp "ConversionExport: Filename=name.txt,Repository=repository_name,
Object=object_type,LogFile=object_typeExport.log,Language=language_code,MatchMi
n=an_ineger"
```

For example, the following command exports the control object type:

```
consoleapp "ConversionExport: Filename=Control.txt,Repository=Siebel Repository,
Object=Control,LogFile=ControlExport.log,Language=ENU,MatchMin=1"
```

For information about setting parameters in this command, see "Parameters You Use with the Conversion Export Utility" on page 246.

For more information, see "How the Conversion Export Utility Converts Strings" on page 157.

7 Repeat Step 6 for each object type that you identified in Step 4.

8 Import the symbolic strings that you converted in Step 6. Enter the following command:

```
consoleapp "ConversionImport: Filename=name.txt,Repository=repository_name,
LogFile=name.log,UnlockProjects=false,SkipParentUpdates=true,Project=Symbolic
Strings"
```

For example:

```
consoleapp "ConversionImport: Filename=Control.txt,Repository=Siebel
Repository,LogFile=ConversionImport.log,UnlockProjects=false,SkipParentUpdates=
true,Project=Symbolic Strings"
```

For information about setting parameters in this command, see "Parameters That You Can Use with the Conversion Import Utility" on page 247.

For more information, see "How the Conversion Import Utility Converts Strings" on page 157.

9 Consolidate strings.

For more information, see "Consolidating Symbolic Strings" on page 158.

Caution About Converting and Consolidating Symbolic Strings

Converting or consolidating symbolic strings might consume computer processing resources.

CAUTION: Converting or consolidating symbolic strings might consume a significant amount of computer processing resources. For information about the requirements for computer processing speed, see *Siebel System Requirements and Supported Platforms* on Oracle Technology Network.

NOTE: For Siebel CRM product releases 8.1.1.9 and later and for 8.2.2.2 and later, the system requirements and supported platform certifications are available from the Certification tab on My Oracle Support. For information about the Certification application, see article 1492194.1 (Article ID) on My Oracle Support.

File and Object command-line parameters are case sensitive.

CAUTION: File and Object command-line parameters that you use when you convert or consolidate a symbolic string are case sensitive. Other command-line parameters are not case sensitive.

Running the Console Application Executable

You can use consoleapp.exe (console application executable) to run various utilities that you use to manage symbolic strings.

To run the console application executable

1 On the computer where you installed Siebel Tools, navigate to the folder:

SIEBEL_TOOLS_ROOT\BIN

2 Enter the following command:

consoleapp *parameter_1 parameter_n*

This command runs the console application executable.

Separating Conversion Files into Smaller Files

The command that this topic describes separates an export file that the conversion utility creates into smaller, more manageable files according to object type. For example, an SRF might include up to 130,000 controls. To improve performance, you can simultaneously import multiple consolidation files of the same object type or of different object types. An average desktop PC can typically run only 10 simultaneous conversion import processes.

To separate conversion files into smaller files

- Add the following parameter to the command you use in Step 5 on page 154:

  ```
  "SplitFile: Filename=Control.txt,Lines=2000"
  ```

  Use values from the following table:

Parameter	Description
Filename	Name of the export file.
Lines	Approximate number of lines in each file. The conversion utility does not separate a set of symbolic strings, so the number of lines might not equal the value you set in this parameter.

How the Conversion Export Utility Converts Strings

The Conversion Export utility can convert any object that can include a translatable string. For example, a control, list column, or applet. The export utility does the following work:

1. Creates a sorted list of English (ENU) child records for each translatable string in an object.
2. Sequentially processes each object that includes multiple translatable strings. For example, a list column that includes a display name and prompt text includes multiple translatable strings.
3. Uses the list it creates in Step 1 to create information about any new symbolic strings.
4. For records that include identical ENU translations, it compares the non-ENU records and reuses the same symbolic string for subsequent records, if possible.
5. Repeats Step 1 through Step 4 for the next object type.
6. Produces an output file that includes information about the new symbolic strings, including information about each language translation and replacement strings. This file is for information purposes. It is not a log file. It is not necessary to review the contents of this file.

How the Conversion Import Utility Converts Strings

To convert the records that use the new symbolic strings, you can use a utility that performs inserts, updates, and deletes on the Siebel database. For input, this utility uses the output file that the conversion export creates. This utility does the following:

1. Creates a new symbolic string record and child symbolic string locale records for each string according to the string values in the target objects. The export file includes information about each string, including a unique name and information about each child locale object.

2 Adds a reference to the new symbolic string in the relevant property of each original object. For example, assume 10 applets exist and that the title for each of these applets is My Big Service Requests. Assume that the non-ENU values for these titles all use the same value. In this situation, the export file includes information about one new symbolic string and instructions for each of the 10 applets to use this new symbolic string as the title. The conversion import does the following:

   - Creates a symbolic string with an ENU value of My Big Service Requests.
   - Sets the Title - String Reference property for each of the 10 applets to the name of this new symbolic string.
   - Clears redundant values for child locale objects. Each of the 10 applets now include a value in the String Reference property and a value in the String Override property. The value in the String Override property is redundant and the conversion import clears this value for each child locale object.

3 Deletes any records that the object locale records no longer reference.

For more information, see "Converting Symbolic Strings" on page 153.

How Siebel CRM References Symbolic Strings at Run Time

Siebel CRM uses the same repository before and after a conversion. For example, assume an applet gets the value for the Title property from a child applet locale record. During conversion, Siebel Tools does the following:

1 Creates a symbolic string.

2 Places the reference for this symbolic string in the Title - String Reference property of the applet.

3 Removes the applet locale record.

After Siebel Tools finishes conversion it gets the applet title from the symbolic string. The display value for the title that it compiles is the same value that this title contained before the conversion. Siebel Tools compiles strings into object definitions in the repository. Siebel CRM reads symbolic strings from these object definitions. It does not read them from the S_SYM_STR table at run time.

Consolidating Symbolic Strings

Consolidating symbolic strings eliminates duplicate symbolic strings that the conversion utility might create when it converts these strings. The conversion utility converts all the symbolic strings for one object type, and then converts all the symbolic strings for the next object type. It does this until it finishes converting all the symbolic strings that you specify. It creates multiple symbolic string records for the same display value that occurs in multiple object types. Duplicate symbolic strings can include identical sets of locale records or one symbolic string might include more child locale records than another string. The strings that these objects use in common are identical. Consolidation can remove these duplicates.

To consolidate symbolic strings

1. Review the important caution information described in "Caution About Converting and Consolidating Symbolic Strings" on page 156.
2. Convert symbolic strings.

   For more information, see "Converting Symbolic Strings" on page 153.
3. Merge duplicate symbolic strings. Run the following command:

   ```
   consoleapp "ConsolidationExport:Filename=ConsExp.txt,Repository=Siebel
   Repository,LogFile=ConsolidationLog.txt,Language=ENU,MatchMin=2"
   ```

   Use values from the following table. For more information, see "Running the Console Application Executable" on page 156.

Parameter	Description
Filename	The name of the export file.
Repository	The name of the repository.
LogFile	The name of the log file.
Language	Same as the Language parameter that you use to export candidates for an object type. For more information, see "Separating Conversion Files into Smaller Files" on page 157.
MatchMin	The minimum number of matches that the utility must find in a set of matching symbolic strings before it writes them to the file. The default value is 2.
SkipSBLStrings	You can use one of the following values: ■ **TRUE.** Ignore all strings that include the SBL_ prefix in the Name property. ■ **FALSE.** Consolidate all strings that include the SBL_ prefix and all custom strings that you create. ■ **MasterOnly.** Do not consolidate strings that include the SBL_ prefix, but use them as master strings. The default value is TRUE.

   After you convert locale strings to symbolic strings, you can use the consolidation utility to find duplicate symbolic strings and merge them and their references into a single symbolic string. As input, this utility uses the file that the consolidation export creates. It deletes redundant symbolic strings. It replaces all references that objects make to these strings with a reference to the master string. This consolidation might consume a significant amount of time because the object types in the repository include approximately 80 translatable string properties.
4. Import consolidated strings. Use the ConsolidationImport business service method.

   For example:

```
consoleapp "ConsolidationImport:Filename=ConsExp.txt,Repository=Siebel
Repository,
LogFile=ConsolidationLog.txt,UnlockProjects=false,SkipParentUpdates=true"
```

These parameters are the same parameters that you use to import the symbolic strings in Step 8 on page 155. For more information, see "Running the Console Application Executable" on page 156.

5 (Optional) Separate a consolidation export file into smaller files.

To separate a consolidation export file into smaller files, you use the same parameters that you use to separate a conversion export file. You can then do Step 4 to import each of these smaller files. For more information, see "Separating Conversion Files into Smaller Files" on page 157.

Using a Batch File to Consolidate Strings

You can use a batch file to consolidate strings.

To use a batch file to consolidate strings

1 Navigate to the *SIEBEL_TOOLS_ROOT*\BIN folder.

2 Enter the following command:

```
strcons Action User_Id Password
```

You use the same parameters that you use when you use a batch file to convert strings. The only difference are:

- If you set the action parameter to Import, then the utility imports the files from the working folder that the TEST_LOCATION parameter in the batch file identifies.
- You do not specify an object type.

For more information, see "Using a Batch File to Convert Strings" on page 160.

Using a Batch File to Convert Strings

You can run the conversion utility from a batch file.

CAUTION: If the Siebel Tools installation path includes a space, then you must enclose this path in quotes.

To use a batch file to convert strings

1 Navigate to the *SIEBEL_TOOLS_ROOT*\BIN folder.

2 Enter the following command:

```
strconv "object_type" action user_Id password
```

For example:

```
strconv "Applet" export user_Id password
```

The following table describes the parameters you can use to run the strconv.bat (string conversion) batch file. You set other parameters in the batch file. For information about the batch file, see comments that the batch file contains.

Parameter	Description
object_type	Object type to convert. For example: ■ `Applet` ■ `Control` ■ `List Column`
action	You can use one of the following values: ■ **Export.** The utility exports all convertible locale records. ■ **Import.** The utility imports the file that the object_type parameter identifies.
user_Id	The user name that you use to log in to the Siebel application.
password	The user password that you use to log in to the Siebel application.

Configuring Locale Data

This topic describes how to configure locale data. It includes the following information:

- Configuring Nontranslatable Locale Object Properties on page 161
- Displaying Controls and List Columns According to Locale on page 162
- Fixing Orphaned String References After an Upgrade on page 164

Configuring Nontranslatable Locale Object Properties

A locale can be one of the following:

- **Translatable.** For example, a text string that defines a control label.
- **Nontranslatable.** For example, the HTML Sequence property, HTML Height property, or HTML Width property of a control.

User interface conventions can vary by locale. For example, one locale might require a different sequence of fields from another locale. To configure a nontranslatable object property for a locale, you can enable language override mode. This mode allows you to store a nontranslatable, locale property as a child locale record of the parent object.

The following example assumes a Siebel enterprise requires Japanese (JPN) and four Western European languages. Japanese does not use middle names but Western European languages do use middle names. Japanese uses the family name first. Western European languages use the family name last.

To configure nontranslatable locale object properties

1. Set the language mode to JPN.

   For more information, see "Setting the Language Mode" on page 19.

2. Enable language override.

   If you do not enable language override, and if you compile any of the Western European languages, then Siebel CRM uses the Japanese configuration of no middle name and family name first. For more information, see "Enabling Language Override" on page 20.

3. In the Object List Editor, locate the object you must modify.

   For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

4. To define locale values, modify the object properties:

   a. To hide the middle name, remove the value from the Title-String Override property.

   b. Reverse the order of the first name and last name.

      You modify these locale values to meet the needs that this example requires. The locale values you must modify depend on your business requirements. You can use the Object List Editor or the layout editor.

5. Compile these modifications to the JPN.srf file.

   For more information, see "Compiling Your Modifications" on page 179.

Displaying Controls and List Columns According to Locale

You can control how Siebel CRM displays a control or list column according to the locale requirements.

CAUTION: To hide or display a control or list column, it is recommended that you modify a property. If you delete a control or list column, then Siebel Tools deletes it from all languages even if you enable language override.

The following example displays locale objects in an applet.

To display controls and list columns according to locale

1. Set the language mode.

   For more information, see "Setting the Language Mode" on page 19.

2 In the Object Explorer, expand the Applet tree.

3 Do one of the following:

- Click Control.
- Expand the List tree, and then click List Column.

4 In the Object List Editor, locate the object you must modify.

For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

5 Modify a property. Use values from the following table.

Object Type	Property	Description
Control	Visible	If TRUE, then Siebel CRM displays this control in the Siebel client in the parent language and in all other languages that it supports.
	Visible-Language Override	If you enable language override, then you can set this property to one of the following values: ■ **FALSE.** Hides the control in the Siebel client. ■ **TRUE.** Displays the control in the Siebel client. For more information, see "Enabling Language Override" on page 20.
List Column	Show in List	If TRUE, then Siebel CRM displays this column in the Siebel client in the parent language and in all other languages that it supports.
	Show in List-Language Override	If you enable language override, then you can set this property to one of the following values: ■ **FALSE.** Hides the column in the Siebel client. ■ **TRUE.** Displays the column in the Siebel client. For more information, see "Enabling Language Override" on page 20.

Fixing Orphaned String References After an Upgrade

An upgrade from one release of Siebel CRM to another release can result in some string references disappearing. The Fix Strings Utility allows you to locate these orphaned strings and update them with new references. This utility uses the Siebel Tools Fix String References business service and the FixStringReferences business service method.

To fix orphaned string references after an upgrade

1. Navigate to the following folder:

   `*SIEBEL_TOOLS_ROOT*\BIN`

2. Open a Windows command line.

3. Enter the following command:

   ```
   consoleapp config_file app_lang user_Id password "Siebel Tools Fix String
   References""FixStringReferences:Repository=repository_name,PriorRepository=prior_r
   epository_name,LogFile=log_file_name.log,FixReferences=true_or_false,VerboseOutput
   =true_or_false,Object=object_type"
   ```

   Use values from the following table.

Parameter	Description
config_file	Specify the path to and name of the Siebel Tools configuration file.
app_lang	Specify the current language, such as ENU.
user_Id	Specify the user Id for the repository that this utility searches.
password	Specify the password for the repository that this utility searches.
repository_name	Required. Specify the name of the repository that includes the string references to fix.
PriorRepository	Specify the name of the repository that your deployment used before the upgrade. If you set the FixReferences parameter to TRUE, then you must include the PriorRepository parameter.
log_file_name	Required. Specify the name of the log file. This utility writes this log file to the current working folder. You can enter an explicit path for this log file.
FixReferences	You can set this parameter to one of the following values: ■ **TRUE.** Fix invalid references. ■ **FALSE.** Save invalid references to the log file. This is the default value.

Parameter	Description
VerboseOutput	You can set this parameter to one of the following values: ■ **TRUE.** Write progress information to the command line. ■ **FALSE.** Do not write progress information to the command line. This is the default value.
object_type	Specify the object type, such as Applet. The utility finds invalid string references that this object type references. If you do not include this parameter, then the utility finds invalid string references for all object types.

Example Commands That Fix Orphaned String References After an Upgrade

The following example runs the utility for the Business Service object type, writes progress output to the command line, and writes information to the fixstrings.log file:

```
consoleapp SIEBEL_TOOLS_ROOT\bin\enu\tools.cfg ENU jgolding db2 "Siebel Tools Fix
String References" "FixStringReferences:Repository=Siebel
Repository,LogFile=fixstrings.log,FixReferences=false,VerboseOutput=true,Object=Bu
siness Service"
```

The following example runs the utility for all object types and writes the results to the fixstrings.log file:

```
consoleapp SIEBEL_TOOLS_ROOT\bin\enu\tools.cfg ENU jgolding db2 "Siebel Tools Fix
String References" "FixStringReferences:Repository=Siebel
Repository,LogFile=d:\temp\fixstrings.log,FixReferences=false"
```

For more information, see "About the Configuration Files" on page 17.

Using the Locale Management Utility

This topic describes how to use the Locale Management Utility. It includes the following information:

- Finding Untranslated Text Strings on page 166
- Finding Existing Translations on page 167
- Finding Objects That the Locale Management Utility Modifies on page 168
- Finding Objects Modified Since the Last Export on page 169
- Exporting Text Strings and Locale Properties to a File on page 170
- Importing Text Strings and Locale Properties on page 171
- Modifying the Value in All Strings for a Language on page 172
- Using the Command Line to Run the Locale Management Utility on page 172

The *Locale Management Utility* (LMU) is a utility in Siebel Tools that you can use to manage how you configure Siebel CRM to localize text strings, such as field labels, and other locale properties, such as the height and width of controls. You can use it to export text strings to a file. After Siebel Tools modifies or translates these strings in the file, you can use the Locale Management Utility to import them back into the repository. The Locale Management Utility also includes search and comparison tools.

Finding Untranslated Text Strings

You can use the Locale Management Utility to find text strings in the repository that Siebel CRM has not translated or to find text strings that Siebel Tools must retranslate because Siebel CRM modified the source string since the last translation. The Locale Management Utility searches or compares objects. It does not search or compare properties. If a locale object includes multiple string properties, then the search returns all strings that the locale object contains, even if only one of them is translated.

To find untranslated strings

1. Log in to Siebel Tools.
2. Click the Tools menu, Utilities, and then click Locale Management.
3. In the Locale Management dialog box, in the Options tab, in the Languages section, choose the source language and the target language.
4. In the Objects section, choose the applications or projects that you must localize.

   For more information about projects, see Chapter 4, “Checking Out and Checking In Projects and Objects.”
5. Click the Untranslated Strings tab.

   The Untranslated Strings tab includes an Attribute column. Siebel CRM uses the terms attribute and property. These terms have the same meaning.
6. (Optional) To display strings that are marked as Redo, make sure the following check box contains a check mark:

   Report String Attributes of Objects Marked With 'Redo' Fag

   For more information, see “How the Locale Management Utility Indicates That Siebel CRM Modified a Record Since the Last Export” on page 170.
7. Click Find Strings.

   The Locale Management Utility searches the properties of the string objects. It searches the objects only in the application or project you choose. It displays object definitions that include strings that are not translated. If you choose to display Redo objects in Step 6, then it also displays the strings that you must retranslate.
8. (Optional) To do more, you can click the following buttons:

   - **Find Views.** Find the views that reference the untranslated strings.

- **Go To.** You choose an object in the Results list, and then click Go To. The Object Explorer displays the object definition that includes the string.
- **Export.** Export all untranslated strings to a file.

9 (Optional) Determine if an existing translation exists that you can use for the untranslated strings that the Locale Management Utility shows in Step 7.

   For more information, see "Finding Existing Translations" on page 167.

Finding Existing Translations

To find an existing translation that you can use for an untranslated string, you can use the Locale Management Utility to search the objects in the repository. You can reuse an existing translation for an object that you create or modify.

To find existing translations

1. Click the Tools menu, Utilities, and then click Locale Management.
2. In the Locale Management dialog box, in the Options tab, in the Languages section, choose the source language and the target language.
3. In the Objects section, choose the application or project that you must localize.
4. Click the Untranslated Strings tab.
5. Click the Find Translations button.

   After a few moments, the Locale Management Utility displays the results. For more information, see "How the Locale Management Utility Finds Existing Translations" on page 167.

How the Locale Management Utility Finds Existing Translations

The Locale Management Utility compares an untranslated string to the properties that might contain this string in other objects in the repository. If it finds an object that includes the same string, then it searches for a translation in the language that you choose as the target language. If it finds a translation, then it displays the best candidates for translation in the Results list and allows you to export it to a file.

For example, assume you choose English-American as the source language and Spanish as the target language. You create an applet with a title of Customer that is not translated. If you click Find Translation in the Locale Management dialog box, then the Locale Management Utility searches the repository for other objects that include a property that includes the Customer text string. If it finds a match, then it searches for a Spanish translation of this string. If a translation already exists, then it displays this translation and you can export it to a file.

How the Locale Management Utility Chooses Among Multiple Translations

If the Locale Management Utility finds more than one translation for a source string, then it does the following:

- If the source string resides in the property of an object that is related to a business component, such as Control Caption or List Column Display Name, then the Locale Management Utility examines translations from the same business component first, and then does the following:
  - If multiple translations exist in the same business component, then it chooses the string that occurs most frequently.
  - If no translations exist in the same business component, then it chooses the translation that occurs most frequently in all business components.

  For example, assume Applet A references the Account business component. Applet A includes a control caption with the value of Account and this value is translated to Account_FRA for French. Assume you create a new applet, Applet B, that also references the Account business component and that Applet B also includes a control caption with the value of Account. If you run Find Translations, then the Locale Management Utility finds Account_FRA as an existing translation and chooses it as the best candidate for this string.

- If the source string is not a property related to a business component, such as Menu Item Caption, then the Locale Management Utility chooses the translation that occurs most frequently.

Finding Objects That the Locale Management Utility Modifies

This topic describes how to find objects that the Locale Management Utility modifies.

To find objects that the Locale Management Utility modifies

1. Click the Tools menu, Utilities, and then click Locale Management.
2. In the Locale Management dialog box, in the Options tab, in the Languages section, choose the source language and the target language.
3. In the Objects section, choose the application or project that you must localize.

   For more information about projects, see Chapter 4, “Checking Out and Checking In Projects and Objects.”
4. Click the Modified Objects tab.
5. In the Search Criteria section, make sure the Changed Since check box includes a check mark.
6. Click the Changed Since drop-down arrow, and then choose the date that the Locale Management Utility must use as the starting date for the search.
7. Click Start.

Finding Objects Modified Since the Last Export

You can use the Locale Management Utility to find objects in the repository that were modified since the last time you exported strings. This search might be useful if your development and localization efforts occur simultaneously. It helps you keep strings in the repository synchronized with the strings that you export to a file for localization.

To find objects modified since the last export

1. Click the Tools menu, Utilities, and then click Locale Management.
2. In the Locale Management dialog box, in the Options tab, in the Languages section, choose the source language and the target language.
3. Click the Modified Objects tab.
4. (Optional) In the Search Criteria section, set the Locale Management Utility to search from a date:
   - Make sure the Changed Since check box includes a check mark.
   - Click the Changed Since drop-down arrow, and then choose the date that the Locale Management Utility must use as the starting date for the search.

     If you use the Changed Since option, and if you click Start, then the Locale Management Utility marks all records it returns for a modified project as Redo regardless of whether Siebel CRM modified a locale property. It does this because it searches for modifications that reside at the object level, not at the property level. For more information, see "How the Locale Management Utility Indicates That Siebel CRM Modified a Record Since the Last Export" on page 170.
5. (Optional) In the Search Criteria section, set the Locale Management Utility to compare objects in the repository to objects in a source file:
   - Make sure the Different From File check box includes a check mark.
   - Click Browse, and then choose the source file.
6. Click one of the following buttons:
   - **Start.** Finds records that match the search criteria, flags the records that it returns as Redo, and then displays the results.
   - **Preview.** Finds records that match the search criteria and displays the results. It does not flag records as Redo.
7. (Optional) To do more, you can click the following buttons:
   - **Save.** Saves a result set in a log file.
   - **Go To.** The Object Explorer displays the parent object of the string or property.
   - **Load.** Imports a result set from a previously saved file. After the Locale Management Utility displays this result set in the Results list, you can use Go To to examine each record.

How the Locale Management Utility Indicates That Siebel CRM Modified a Record Since the Last Export

If Siebel CRM modified a record in the repository since the last export, then it might require another translation. The Locale Management Utility uses the Redo flag to mark this record.

If you use the Locale Management Utility to import records, then it compares the source language records in the repository with the source language records in the import file. If Siebel CRM modified the records in the repository since the export occurred, then this utility uses the Redo flag to mark the target language records. This configuration helps you identify records that might require another translation. For more information, see "Importing Text Strings and Locale Properties" on page 171.

Exporting Text Strings and Locale Properties to a File

You can use the Locale Management Utility to export strings and other locale properties to a file. This file can use any one of the following file types:

- .slf
- .txt
- .xlf

You cannot use the Locale Management Utility to export to a Microsoft Excel.xls file.

To export text strings and locale properties to a file

1. In Siebel Tools, click the Tools menu, Utilities, and then click Locale Management.
2. In the Locale Management dialog box, in the Options tab, in the Languages section, choose the source language and the target language.

   To export text strings and locale properties, the language mode and the Locale Management Utility source language must use the same language. For more information, see "Setting the Language Mode" on page 19.

3. In the Objects section, choose the application or project that the Locale Management Utility must export.
4. Click the Export Tab.
5. Click String Attributes Only or All Localizable Attributes.

   A localizable property can include translatable strings and other locale properties, such as the width and height of controls. Each locale property might contain a different value that meets the need for a specific locale. Siebel CRM uses the terms attribute and property. These terms have the same meaning.

6. Click Export.
7. In the Save As dialog box, choose the folder where the Locale Management Utility must export the file.

   For example, *SIEBEL_TOOLS_ROOT*\OBJECTS\language_code, where *language_code* is the target language you choose in Step 2.

8. Enter a file name, choose a file type, and then click Save.

   - If you click String Attributes Only in Step 5, then the available file types are .txt or .xlf.
   - If you click All Localizable Attributes in Step 5, then the available file type is .slf.

Importing Text Strings and Locale Properties

You can use the Locale Management Utility to import translated strings and other locale properties into the repository. You can preview the import before the utility actually does the import.

To import text strings and locale properties

1. Click the Tools menu, Utilities, and then click Locale Management.
2. In the Locale Management dialog box, in the Options tab, in the Languages section, choose the source language and the target language.
3. Click the Import tab.
4. Enter the name of the file that the utility must import.

   You can use the Browse button to choose the file. The default file name is one of the following:

   - `Results.txt` if the file includes strings only
   - `Results.slf` if the file includes all locale properties

   If you import an XML Localization Interchange Field file (.xlf), then make sure a working Internet connection exists during the import.
5. (Optional) To preview the import, do the following:

   a. Enter the path and name of the file where the utility must store the results for previewing.

      The default file name is `preview.log`.

   b. Click Preview.

      The Locale Management Utility writes the results of the import to the log file. It does not write the results to the repository. It does not mark modified records with a Redo flag during preview.
6. (Optional) To mark records that Siebel CRM modified since the export occurred, make sure the following check box contains a check mark:

   Mark Changed Records with Redo Flag

   For more information, see "How the Locale Management Utility Indicates That Siebel CRM Modified a Record Since the Last Export" on page 170.
7. Click Import.

   The utility imports the locale properties into the repository. It creates the following log file in the *SIEBEL_TOOLS_ROOT*\OBJECTS folder:

```
LMUImportTruncation.log
```

This file includes detailed information and error messages about records that the utility did not import.

Modifying the Value in All Strings for a Language

You can use the Locale Management Utility to replace strings in bulk. For example, assume you must configure Siebel CRM to modify all text strings that include Accounts to Companies for the English locale. You can use the Locale Management Utility to export the strings to a file, modify the file so that it includes only Companies instead of Accounts, and then import these strings into the repository.

Using the Locale Management Utility to replace strings is most useful for strings that Siebel CRM stores in string-override fields. For information about modifying symbolic strings, see "Modifying a Symbolic String to Globally Update Display Values" on page 149.

To modify the value in all strings for a language

1. Export the strings you must modify to a file.

   The source language and the target language cannot be the same language. For more information, see "Exporting Text Strings and Locale Properties to a File" on page 170.

2. Use a text editor to modify the strings in the file that you created in Step 1:

   - Remove strings that you do not want to replace from the file.
   - In the Target String column of the file, enter the new string that replaces the old value.

3. Import the file.

   For more information, see "Importing Text Strings and Locale Properties" on page 171.

Using the Command Line to Run the Locale Management Utility

You can use the command line to run the Locale Management Utility. The commands that this topic describes use the following format:

- *xxx*. A placeholder for a required parameter.
- [*xxx*]. A placeholder for an optional parameter.
- *xxx*|*yyy*. An OR condition. For example, xxx or yyy.

You must include the absolute path with any file name that you specify. For example, if you specify the export file as `results.txt`, then the utility creates this file in the current folder. In this example, if the installation folder is `C:\Program Files\Siebel\8.0\Tools`, then the utility creates this file in the following folder:

```
C:\Program Files\Siebel\8.0\Tools\BIN
```

It does not create it in the following folder:

```
C:\Program Files\Siebel\8.0\Tools\OBJECTS
```

Using the Command Line to Export Strings and Locale Properties

The command that this topic describes allows you to export localizable properties for all projects or for all applications. You can specify one of the following values:

- **all.** Export all translatable properties and language override properties to a file that uses an .slf extension.
- **string.** Export only string properties to a file that uses a .txt or .xlf extension. If you do not specify a file name, then the utility displays an error.

Format

This command uses the following format:

```
/lmu source_language target_language export proj|app all|string file [project_file]
```

Project Parameter

Starting with Siebel CRM version 8.0, the Locale Management Utility includes the following parameter:

```
project_file
```

You can use it to identify the projects to export. This parameter identifies the name of an ASCII text file that includes a list of projects that are delimited by a line feed. If you do not include the project_file parameter, then the utility exports all projects.

You can use the proj|app parameter to identify the projects or applications that include the strings to export. If you use the project_file parameter, then you must choose proj. If you choose app, and if you include the project_file parameter, then the utility ignores this file. For more information about projects, see Chapter 4, “Checking Out and Checking In Projects and Objects.”

Example

The following command exports properties:

```
siebdev /u sadmin /p db2 /d server /lmu ENU FRA export proj all
C:\temp\my_proj_results.slf C:\temp\proj_to_exp.txt
```

This example exports all string properties and language override properties for the projects that the following file lists:

```
C:temp\proj_to_exp.txt
```

It exports these properties to the following file:

```
C:\temp named my_proj_results.txt
```

The source language of the file is English-American (ENU) and the target language is French (FRA).

Using the Command Line to Import a File

The command that this topic describes allows you to import a file. If the source string in the import file is different than the source string in the repository, then it marks the target locale object as Redo.

Format

This command uses the following format:

```
/lmu source_language target_language import file
```

Example

The following command imports a file:

```
siebdev /u sadmin /p db2 /d server /lmu ENU FRA import "C:\Program
Files\Siebel\8.0\Tools\objects\results.slf"
```

This example imports the `results.slf` file from the following folder. This folder is the installation folder for an earlier version of Siebel Tools:

```
C:\Program Files\Siebel\8.0\Tools\objects
```

The source language of the file is English-American (ENU) and the target language is French (FRA).

The file includes all localizable string properties and language override properties.

Using the Command Line to Export Strings That Require Translation

The command that this topic describes allows you to export all untranslated strings and strings marked with the Redo flag to a file. It can export all projects or all applications. The exported file includes the related view names.

Format

This command uses the following format:

```
/lmu source_language target_language todo proj|app [file]
```

Example

The following command exports strings that require translation:

```
siebdev /u sadmin /p db2 /d server /lmu ENU FRA todo app "C:\Program
Files\Siebel\8.0\Tools\objects\results.txt"
```

This example finds all untranslated strings and redo strings for all applications. It exports the results to the following file:

```
C:\Program Files\Siebel\8.0\Tools\objects\results.txt
```

The source language is English-American (ENU) and the target language is French (FRA).

Using Advanced Compile to Localize a Repository

This topic describes how to use Advanced Compile to localize a repository. For more information, see "Compiling Your Modifications" on page 179.

Overview of Using Advanced Compile

Advanced Compile is a feature in Siebel Tools that you can use to assist with localization. If you localize a repository, then you can use Advanced Compile to avoid the following problems:

- **Localization is not complete when testing begins.** Localized strings might become available only later in the development process. This situation requires you to delay testing until the localized strings are available or to test without these strings.
- **Missing translations might be difficult to find.** After you import localized strings, you might start testing using a language repository, but some strings might be missing because one of the following occurs:
  - All language translations are not imported.
  - Development continues beyond the localization export date.
  - Some projects are mistakenly not exported.

  These missing localized values can cause Siebel CRM to not display screens, leave the labels for tabs empty, or not display field labels. These problems might be difficult to fix.

To avoid these problems, Advanced Compile does the following:

- Inserts a temporary string for each missing translation so that Siebel CRM remains functional.
- Adds a *pseudolocalization prefix* to each string, which is a prefix that Siebel Tools uses to test the appearance of string in a language. It can include identifying characters, such as an Asian character or an accented European letter.

Advanced Compile is available with Siebel Tools version 8.0 and later.

Benefits of Adding a Prefix

Adding a pseudolocalization prefix provides the following benefits:

- **Detects a hard-coded string.** Adding a prefix makes the string different from the original English string. The code that references a hard-coded string fails. For example, assume Advanced Compile adds the ÐØÉ_ prefix to a customer status string named ACTIVE. Assume you hard-code a reference to this ACTIVE string. This hard-coded reference cannot locate the ACTIVE string because it does not match ÐØÉ_ACTIVE, and the code fails.
- **Detects code that does not accept a non-ASCII character.** If a script or add-in product that is not correctly internationalized encounters a string that includes a pseudolocalization prefix, such as ÐØÉ_ACTIVE, then an error can result. If this string does not include the pseudolocalization prefix, then Siebel CRM might not detect the error until localization is performed. Fixing this problem late in the development effort might be problematic.

- **Increases string length up to three characters.** If the translation is longer than the original text, then this increased length allows you to test field size and column size to make sure these sizes can accept a localized string. This situation might occur with a Western European language.

Data That Advanced Compile Modifies in the Repository

Advanced Compile does not modify the underlying data in the repository. It modifies only the strings in the compiled repository. It works on only strings that the S_SYM_STR table contains. It does not work on error messages that separate applications contain, such as Siebel Handheld or a third-party application.

Using Advanced Compile

This topic describes how to use Advanced Compile.

To use Advanced Compile

1 Make a backup copy of the repository.

   CAUTION: Before you use Advanced Compile, make a backup copy of the repository.

2 Enable language override.

   You must enable language override before you can use Advanced Compile. For more information, see "Enabling Language Override" on page 20.

3 Click the Tools menu.

4 While holding down the SHIFT key, click the Compile Projects menu item.

   Siebel Tools displays the Advanced button in the Object Compiler dialog box only if you hold down the SHIFT key.

5 Click Advanced.

6 In the Advanced Options dialog box, choose the following items. Each of these items is optional:

   - **Enable Pseudo-Localization.** Adds a prefix to each string.
   - **Pseudo-Localization Prefix.** Enter the characters that Siebel Tools uses to add a prefix to each string. Siebel Tools adds this prefix only if you add a check mark to the Enable Pseudo-Localization check box. To test a language that includes characters that are specific to the language, you must include one or more of these characters in the prefix.
   - **For Missing Translations, Display Lang_code + Object Name.** Inserts a temporary string for each missing translation.

7 Click OK.

8 In the Object Compiler dialog box, choose the projects that Siebel Tools must compile.

9 Choose a repository for the language that you must test.

10 Click Compile.

11 Test your work:

  a Click the Debug menu, and then click Start.

    Siebel Tools starts the Siebel client in a new browser window. If you enabled all options in Step 6, then Siebel CRM prefixes strings with the characters you specify in Step 6, and temporary strings include the prefix, language code, and object name. For more information, see "Setting Debug Options to Open the Siebel Client" on page 187.

  b To locate missing string translations, navigate the screens and views in the Siebel client.

8 Compiling, Testing, and Troubleshooting Your Customizations

This chapter describes how to use Siebel Tools to compile, test, and troubleshoot your customizations. It includes the following topics:

- Compiling Your Modifications on page 179
- Testing and Troubleshooting Your Modifications on page 187

Compiling Your Modifications

This topic describes how to compile your customizations. It includes the following information:

- Overview of Compiling on page 179
- Compiling Projects on page 180
- Compiling Objects on page 182
- Using an Incremental Repository Upgrade Kit on page 183
- Using the Command Line to Import, Export, or Compile Objects on page 183

This task is a step in "Roadmap for Setting Up and Using Siebel Tools" on page 14.

Overview of Compiling

After you modify objects you must compile these modifications to the repository. The compile adds any new objects you create to the repository. These objects are then available to the Siebel client. Siebel Tools can compile projects or an individual top-level object:

- **Compile projects.** More efficient if you modify many objects in one or more projects. For more information about projects, see Chapter 4, "Checking Out and Checking In Projects and Objects."
- **Compile objects.** More efficient if you modify only a few objects. For information about top-level object types, see "Displaying Object Types in the Object Explorer" on page 26.

Differences in SQL Code That Might Affect Your Database Customizations

If you compile a new repository, and then recompile this same repository, then the two versions of this repository are logically and functionally the same but they might contain slight differences in the SQL code. For example, a new compile might include the following SQL code:

```
select col1, col2
from
table
where col1 = 'ABC'
```

A recompile of this repository might include the following SQL code:

```
select col2, col1
from
table
where col1 = 'ABC'
```

Functionality for these repositories is identical but the SQL code is slightly different. These differences do not impact Siebel CRM but they might affect customizations you make to an Oracle database that expects the SQL code to remain identical for different compilations. It is important that your customizations accommodate this variability in the SQL code.

Compiling Projects

You use the Object Compiler to compile projects. For more information about projects, see Chapter 4, "Checking Out and Checking In Projects and Objects."

To compile projects

1. Make sure Siebel Tools is connected to a database that has the sort order set to binary.
2. Click the Tools menu, and then click Compile Projects.
3. In the Object Compiler dialog box, choose the projects that Siebel Tools must compile.

   If you compile only one or a subset of projects, then some restrictions apply. For more information, see "Compiling Individual Projects" on page 181.
4. In the Siebel Repository File window, click Browse, and then choose a repository.

   For more information, see "Identifying the Repository That You Must Compile" on page 181.
5. (Optional) To automatically start a local instance of the Siebel Web Client when the compile finishes, do the following:

   a. Make sure the Auto-start Web Client check box includes a check mark.
   b. Specify the location of the Siebel executable, the application configuration file, and other relevant settings in the Development Tools Options dialog box. For more information, see "Setting Debug Options to Open the Siebel Client" on page 187.

   If you enable Auto-start Web Client, and if the Siebel Web Client is already open, then this client restarts and displays the same view it displayed before the compile. It does this when the compile finishes.
6. Click Compile.

   Siebel Tools compiles the projects you choose in Step 3 to the repository that you specify in Step 4. Siebel CRM makes any modifications that you make immediately available in any instances of the Siebel Web Client that read this repository. For more information, see "Testing and Troubleshooting Your Modifications" on page 187.

Compiling Individual Projects

To compile individual projects, you must first compile all projects at least one time.

CAUTION: It is recommended that you compile a subset of projects only to a predefined repository or to a repository that you create from a full compile. This configuration makes sure that the repository you compile includes all objects that Siebel CRM requires. For more information, see "About Predefined Objects" on page 24.

If you compile a single project, then the Object Compiler does not remove inactive top-level objects from the repository. It does remove inactive child objects. For example:

- If you deactivate the Name list column in the Account List Applet, and then compile the Account SSE project, then the compiler removes the Name list column from the repository.
- If you deactivate the Account List Applet, and then compile the Account SSE project, then the compiler does not remove the Account List Applet.

For information about top-level object types, see "Displaying Object Types in the Object Explorer" on page 26.

Identifying the Repository That You Must Compile

You typically compile to the repository that the local instance of the Siebel client reads. The RepositoryFile parameter in the application configuration file identifies this repository.

Do not compile or modify the default repository that Siebel Tools uses. If you click Browse in the Object Compile dialog box, then the Save As dialog box displays the name of this repository. This repository is typically siebel.srf and resides in the following folder:

SIEBEL_TOOLS_ROOT\OBJECTS\ *language*

This file is locked because the Siebel Tools client is currently reading it. If you attempt to compile this repository, then Siebel Tools displays an error message.

To identify the repository that you must compile

1. Use a text editor to open the application configuration file.

   For more information, see "About the Configuration Files" on page 17.

2. In the Siebel section, examine the RepositoryFile parameter to determine the repository that it references.

   For example:

   ```
   RepositoryFile=siebel_sia.srf
   ```

Compiling Objects

You can compile a single object or a group of top-level objects of the same type. For example, if you modify several applets, then you can compile only the applets you modify rather than compiling entire projects. For information about top-level object types, see "Displaying Object Types in the Object Explorer" on page 26.

To compile objects

1 Do Step 1 on page 180 and Step 2 on page 180.

2 In the Object List Editor, right-click an object, and then click Compile Selected Objects.

   To choose multiple objects, see "Choosing More Than One Record in the Object List Editor" on page 44.

   For alternative ways to start the compiler, you can do the following:

   - Click the Compile button on the Debugger toolbar.
   - Press F7.

3 In the Siebel Repository File window, click Browse, and then choose a repository.

   For more information, see "Identifying the Repository That You Must Compile" on page 181.

4 Click Compile.

   Siebel Tools compiles the projects that you choose in Step 2 to the repository that you specify in Step 3. Siebel CRM makes any modifications that you make immediately available in any instances of the Siebel Web Client that read this repository. For more information, see "Testing and Troubleshooting Your Modifications" on page 187.

Objects That Must Exist in the Production Database

To work correctly, the following repository objects must exist in the production database. Siebel CRM sets the No Compile property for these objects to TRUE, by default. Siebel Tools does not compile them to the repository. You can configure these objects:

- Assignment objects and their child objects
- Dock objects and their child objects
- EIM interface table objects and their child objects
- Workflow policy objects and their child objects

You cannot configure the following objects but they must exist in the production database. These objects allow you to use various administrative and batch features:

- Schema maintenance objects
- Server component objects
- User key attribute objects

Using an Incremental Repository Upgrade Kit

If you compile a repository to create an Incremental Repository Upgrade Kit, then you can specify a reference repository when you compile the new repository. This reference repository can minimize the size of the kit and the time required to do an upgrade. A *reference repository* is a prior version of the repository. The incremental repository upgrade includes only the differences between the reference repository and the new repository. For more information, see *Siebel Anywhere Administration Guide*.

To use an incremental repository upgrade kit

1. Click the Tools menu, and then click Compile Projects.

   For more information about projects, see Chapter 4, “Checking Out and Checking In Projects and Objects.”

2. In the Object Compiler dialog box, click Reference SRF, and then specify the path and file name of the prior repository version.
3. Click the projects you must compile.
4. Click Compile.

Using the Command Line to Import, Export, or Compile Objects

This topic describes how to use the command line to import, export, or compile objects.

To use the command line to import, export, or compile objects

1. On the computer where you installed Siebel Tools, in Microsoft Windows, click the Start menu, enter cmd, and then press the Enter key.
2. In the command-line window, navigate to the following folder:

   SIEBEL_TOOLS_ROOT\BIN

3. Enter the following command:

   siebdev/*switch switch_parameters*

   where:

   - *switch* is batchimport, batchexport, or bc
   - *switch_parameters* are the parameters that you enter for the switch

     For more information about how to set the switch and parameters, see the following information:

     - Using the Command Line to Import Objects on page 184
     - Using the Command Line to Import Only Modified Objects on page 185

- Using the Command Line to Compile Objects on page 185
- Using the Command Line to Export Objects to an Archive on page 190

Using the Command Line to Import Objects

This topic describes how to use the command line to import objects.

To use the command line to import objects

1 Open a command line, and then navigate to the following folder:

   `*SIEBEL_TOOLS_ROOT*\BIN`

2 Enter the following command:

```
siebdev.exe /c config_file /d database /u user_name /p password /batchimport
"Siebel Repository" import_mode \ sif_files log_file
```

   where:

   - *import_mode* determines how to handle a conflict. You can use one of the following values:
     - overwrite.
     - merge.
     - skip. If you use skip, and if a corresponding object does not exist in the repository, then there is no conflict and the utility imports the record.

     For information, see "Options You Can Choose to Resolve a Conflict" on page 197.
   - *sif_files* specifies the name of one or more sif files that contains the objects that batchimport imports, such as *sif_file1*, *sif_file2*, *sif_file_n,* or the path to a folder that includes .sif files.
   - *log_file* specifies the name of the file that siebdev uses to log information.

You can use the full path or a relative path to the current folder to specify the archive file or the log file.

The following example imports the import1.sif file that resides in the parent folder and the import2.sif file that resides in the Siebel Tools installation folder. It uses the overwrite mode. It logs the results to the import.log file:

```
siebdev.exe /c tools.cfg /d sample /u sadmin /p sadmin /batchimport "Siebel
Repository" overwrite \import1.sif "C:\Program Files\Siebel\8.0\Tools\import2.sif"
import.log
```

The following example imports all sif files that the `C:\Program Files\Siebel\8.0\Tools\MyFiles` folder contains. It uses the merge mode. It logs the results to the import.log file:

```
siebdev.exe /c tools.cfg /d sample /u sadmin /p sadmin /batchimport "Siebel
Repository" merge "C:\Program Files\Siebel\8.0\Tools\MyFiles" import.log
```

Using the Command Line to Import Only Modified Objects

This topic describes how to use the command line to import only modified objects according to the objects that an SDF file defines.

To use the command line to import only modified objects

1 Open a command line, and then navigate to the following folder:

    SIEBEL_TOOLS_ROOT\BIN

2 Enter the following command:

```
siebdev.exe d database /u user_name /p password /t tag_name /batchimport
"repository_name" merge file_type input_folder log_file
```

where:

- *database* must specify one of the following values:
    - local
    - server
    - sample
- t *tag_name* specifies the name of a tag. Batch import imports only the repository modifications that Siebel Tools associates with this tag name. You can specify only one tag name. You cannot specify multiple tag names. For more information about tags, see "Using Object Tags to Manage Modifications That Multiple Developers Make" on page 56.
- *repository_name* is the name of the repository where you compile your modifications. It is typically Siebel Repository, as shown in the Repository field of the About Repository File dialog box. For more information, see Viewing Information About the Current Repository on page 49.
- *file_type* is an optional parameter. You can specify SDF or SIF.
- *input_folder* identifies the folder where the SDF or SIF files reside.
- *log_file* is the name of the file that siebdev uses to log information.

For example, the following command specifies to import all objects that are tagged with the ABC team - developer 1 tag that the SDF file references, and to import these objects from each of the SDF files that reside in the D:\SDF folder. It imports these files into the Siebel Repository. It saves the log file in the Import.log file that resides in the D:\SDF\ folder:

```
siebdev.exe /u sadmin /p sadmin /d Local /t "ABC team - developer 1" /batchimport
"Siebel Repository" "Merge" "SDF" "D:\SDF" "D:\SDF\Import.log"
```

Using the Command Line to Compile Objects

This topic describes how to use the command line to compile objects.

To use the command line to compile objects

1. Open a command line, and then navigate to the following folder:

   SIEBEL_TOOLS_ROOT`\BIN`

2. Enter the following command:

   `siebdev.exe /c` *`config_file`* `/d` *`database`* `/u` *`user_name`* `/p` *`password`* `/bc` *`repository_name SRF_name`*

The following example compiles a repository named Siebel Repository into the siebel.srf file:

```
siebdev.exe /c tools.cfg /d sample /u sadmin /p sadmin /bc "Siebel Repository"
Siebel.srf
```

If you do not specify the path to the repository file, then siebdev compiles to the following folder:

SIEBEL_TOOLS_ROOT`\OBJECTS`

Using the Command Line to Compile Objects in a Multilingual Deployment

You can use the following switch to support a multilingual deployment. This switch sets the Siebel Tools language for the compile:

`/tl` *`language_code`*

The following example compiles the repository for a Japanese deployment:

```
siebdev.exe /c tools.cfg /d sample /u sadmin /p sadmin /tl JPN /bc "Siebel
Repository" Siebel.srf
```

If you use the `tl` switch to specify a language code, and if the language pack for this language code does not exist in the repository, then the following occurs:

- The compile proceeds.
- The compile does not include strings for the compiled objects.
- Siebel CRM does not display list column headings and menu item text in the Siebel client.

Applying Patches in Batch

This topic describes how to apply patches in batch.

To apply patches in batch

1. Open a command line, and then navigate to the following folder:

   SIEBEL_TOOLS_ROOT`\BIN`

2. Enter the following command:

   `siebdev.exe /u sadmin /p sadmin /d local-dest /applybatchpatch` *`repository_name folder_that_contains_the_patch_files log_file_name`*

Testing and Troubleshooting Your Modifications

This topic includes the following information:

- Setting Debug Options to Open the Siebel Client on page 187
- Recovering from a Failed Development Server on page 188

To test your modifications, you can use a local instance of the Siebel Web Client that runs on your computer. If you compile objects and test the results locally, then consider the following:

- If you compile modifications to a repository, then any local instances of the Siebel Web Client that are open and that read this repository automatically close, reopen, and then display the modified configuration.
- You can configure Siebel Tools to automatically open the Siebel Web Client and read the most current repository. For more information, see Step 5 on page 180.
- To display your modifications in a local instance of the Siebel Web Client, this client must read the repository that you compile. For more information, see "Identifying the Repository That You Must Compile" on page 181.

This task is a step in "Roadmap for Setting Up and Using Siebel Tools" on page 14.

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

For information about using debug windows, see the following information:

- Using the Calls Window on page 135
- Using the Watch Window While it Monitors a Script on page 136

Setting Debug Options to Open the Siebel Client

The debug options allow you to modify the run time settings that open an instance of the Siebel Web Client in the following situations:

- You enable Auto-start Web Client. For more information, see Step 5 on page 180.
- You click the Debug menu, and then click Start to start an instance of the Siebel Web Client. You typically do this if you must debug Siebel eScript or Siebel VB. For more information, see *Siebel eScript Language Reference* or *Siebel VB Language Reference*.

To set debug options to open the Siebel client

1. In Siebel Tools, click the View menu, and then click Options.
2. In the Development Tools Options dialog box, click the Debug tab.
3. Set the debug options.

   Make sure you set the following options:

- Executable
- CFG File
- Browser
- Working Directory

For more information, see "Development Options for Debugging" on page 244.

Recovering from a Failed Development Server

This topic describes how to recover from a failed development server or failed repository. It assumes that you cannot access a backup copy of this server or repository. It requires that a recent image of the repository exist on the computer that a remote user uses, and that this user performed a full Get to create this repository. For more information, see "Getting All Projects from the Server Repository" on page 87.

To recover from a failed development server

1. Log in to Siebel Tools on the remote computer that includes the intact repository.
2. Archive the projects that this repository contains.

   For more information, see "Exporting Objects to an Archive" on page 190.
3. Create a new repository:
   - To create a repository record, you can add a new repository object in Siebel Tools.
   - To create the schema in the database for the repository, you can run imprep.ksh for Linux or UNIX.

   For more information, see "Exporting and Importing Repositories" on page 66.
4. Import the archive you created in Step 2 into the repository you created in Step 3.

   For more information, see "Importing Objects from an Archive" on page 195.
5. Test the repository you created in Step 3 and make sure it includes the expected functionality.
6. Check projects into the Siebel Server.

   For more information about projects, see Chapter 4, "Checking Out and Checking In Projects and Objects."

9 Archiving Objects

This chapter describes how to archive objects. It includes the following topics:

- Overview of Archiving Objects on page 189
- Exporting Objects to an Archive on page 190
- Importing Objects from an Archive on page 195
- Using the Application Deployment Manager on page 204

Overview of Archiving Objects

Archiving is a process where you export objects from the repository to an archive file. You can then import objects from the archive file back into the repository. You can use an archive to back up sets of objects or move them to another environment that shares the same physical database schema as the source environment. Note the following:

- An archive does not depend on a database because it includes only repository information. You can use it to exchange repository data between environments with different database platforms, including local and server databases, as long as these databases use the same schema.
- If you import objects from an archive, then you can specify conflict resolution rules for each object. You can configure Siebel Tools to ignore an imported object, replace an existing object with an imported object, or merge two objects according to object properties.
- You can archive individual objects or entire projects to an archive. For more information about projects, see Chapter 4, “Checking Out and Checking In Projects and Objects.”
- You can use code control software to control an archive. For more information, see “Configuring Third-Party Code Control” on page 96.
- If you must back up or move the entire repository to another environment, then see “Exporting and Importing Repositories” on page 66.

For more information about the files that you use when archiving objects, see “Files That You Use to Manage Repositories” on page 48.

Using Export and Import for Different Versions of Siebel CRM

If you export objects from one version of Siebel CRM to another version, then do not import these objects through .sif files into a different version of Siebel CRM because Siebel CRM might modify object definitions between these versions. If you import an object that is not valid, then an invalid configuration might result. Oracle does not support an invalid configuration.

Where Siebel Tools Stores Archive Files

The Temp parameter in the Siebel section of the tools.cfg file identifies the folder where Siebel Tools stores the following items:

- Archive files
- Projects and objects that you check out or check in

Siebel Tools sets this Temp parameter to the `*client_root*\TEMP` folder of your Siebel Tools installation folder, by default. For more information, see "About the Configuration Files" on page 17.

Exporting Objects to an Archive

You can export top-level objects to an archive, such as business components, applets, views, and projects. Siebel Tools exports and imports child objects with their parents. For information about top-level object types, see "Displaying Object Types in the Object Explorer" on page 26.

Do not export the repository object to export an entire repository. If you do this, then the export file is too large and it degrades performance. Instead, you can export a repository. For more information, see "Exporting Repositories" on page 70.

This task is a step in "Roadmap for Setting Up and Using Siebel Tools" on page 14.

To export objects to an archive

1. In the Object List Editor, locate the object you must export.

   For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

2. Click the Tools menu, and then click Add To Archive.

   Siebel Tools displays the Export to Archive File dialog box. The status bar in this dialog box indicates the child objects that Siebel Tools includes in the export. When the process finishes, this dialog box displays the top-level objects in the Objects to Archive list.

3. If you must add more objects, then repeat Step 1 through Step 2 for each object. Do not close the Export to Archive File dialog box while you add these objects.

   You can add objects of the same object type or you can add objects of another object type.

4. In the Export to Archive file dialog box, in the Archive File window, enter the path and file name of the archive file.

   For example, enter `C:\Siebel\8.2\Tools_1\objects\objects.sif`.

5. Click Save.

   Siebel Tools creates a SIF file in the location that you specify in Step 4.

Using the Command Line to Export Objects to an Archive

You can use the command line to export objects to an archive.

To use the command line to export an archive

1. Open a command line, and then navigate to the following folder:

    ```
    SIEBEL_TOOLS_ROOT\BIN
    ```

2. Run the siebdev.exe file. Use the following command:

    ```
    siebdev /c config_file /d database /u user_name /p password /batchexport
    repository_name input_file_name log_file
    ```

    The following example exports the objects that the `obj.txt` input file specifies. It logs results to the `export.log` file:

    ```
    siebdev /c tools.cfg /d sample /u sadmin /p sadmin /batchexport "siebel
    repository" obj.txt export.log
    ```

    For more information about the input file, see "Input File That Batch Export Uses" on page 191.

Input File That Batch Export Uses

The batch export switch uses an input file that specifies the objects to export. This input file uses the following format. You cannot use a space before or after a comma in this file:

```
object_type,object_name,query_expression,file_name.sif
```

where:

- *object_type* identifies the object type to export, such as Business Component.
- *object_name* identifies the name of the object to export, such as Account.
- *query_expression* can include any query that Siebel Tools can use. For more information, see "Using the Query Menu to Run a Query" on page 32.
- *file_name*.sif can use an absolute file path or a relative file path to the current folder.

You can include multiple lines in the input file and each of these lines can specify to export multiple objects to a different SIF file. If you specify the same SIF export file in multiple lines, then batch export uses only the last export that you specify.

For example, consider the following line from an input file. In this example, the batch export switch exports all business components where the Name property is like *Account* to a repository file named export.sif:

```
Business Component,*Account*,export.sif
```

Exporting Only Modified Objects to an Archive

This topic describes how to use an SDF file to export only the modifications that you make to an object instead of the entire definition of this object. For more information about:

- Using an SDF file to import only the modifications that you make, see "Importing Only Modified Objects from an Archive" on page 200,

- SDF files, see "How Siebel Tools Minimizes the Data That it Processes during Exports and Imports" on page 49.

To export only modified objects to an archive

1. Set up Siebel Tools to use iPackMode.

   For more information, see "Enabling Object Tagging in Siebel Tools" on page 57.

2. Synchronize the computer clocks:

   a. Log in to Siebel Tools.

   b. Click the View menu, and then click Options.

   c. In the General tab of the Development Tools Options dialog box, make sure the values in the Date and Time fields of the Changed Date section match the date and time values of the database server that contains the Siebel repository.

      Siebel Tools uses the values in these fields as the date reference. It extracts any objects that you have modified since this date. For more information about using the Changed Date section, see "Exporting Objects to a Mid-Level Release" on page 207.

      These values can vary only to allow for time zone differences. For example, if the database server resides in California and is set at 2:03 PM Pacific Standard Time, and if the computer that contains Siebel Tools resides in Colorado and uses Mountain Standard Time, then you must make sure the clock for the Siebel Tools computer is set to 3:03 PM. In the context of Greenwich Mean Time (GMT), a one hour difference exists Pacific Standard Time and Mountain Standard Time.

3. In Siebel Tools, click the Tools menu, and then click Generate Mid-Level Release.

4. In the Generate Mid-Level Release dialog box, in the Mid-Level Release Label window, enter text that describes the purpose of the export.

   Siebel Tools uses the text you enter in the folder names and file names that it creates during the export. This text must conform to Windows standards for folder names and file names.

5. In the Export Options section, accept the default Single File for All Objects value.

   This option configures Siebel Tools to create one SDF file for all objects.

6. In the Export File Type section, click SDF.

7. In the Export Criteria section, set one of the following export criteria:

   - **Tag-Time.** Siebel Tools does the SDF extraction according to the following items:
     - The tag that you choose in the Development Tag list
     - The Start Date when you modified the object
   - **Tag.** Siebel Tools does the SDF extraction according to the tag that you choose in the Development Tag list. It does not consider the Start Date. You can use the CTRL key to choose more than one tag.
   - **Time.** Siebel Tools does the SDF extraction according to only the Start Date. It does not consider the tag that you choose in the Development Tag list.

8 (Optional) In the Development Tag section, choose a development tag.

Siebel Tools displays tags in a list in the Development Tag section. These tags allow you to filter the objects that Siebel Tools exports. If you:

- **Do not choose a tag.** Siebel Tools exports the list of objects according to the value that you set in the Export Criteria section.
- **Choose a tag.** Siebel Tools filters the list to include only objects that are associated with the development tag you choose, and according to the value that you set in the Export Criteria section. You can hold down the CTRL key to choose multiple tags. To remove a tag that is currently chosen, you can hold down the CTRL key, and then click this chosen tag.

For more information, see "Managing Modifications That Developers Make to Repositories" on page 54.

9 Click Generate List.

The Generate Mid-Level Release dialog box lists the objects that it exports if you click Export.

10 Click Export.

Siebel Tools creates a single SDF file in the following folder:

```
TOOLS_ROOT\ADM\Mid-Level Release Label\
```

where:

- *Mid-Level Release Label* is the text you enter in Step 4.

For example:

```
Tools\ADM\GenerateSDF
```

Siebel Tools exports modifications that occurred since the date that the Start Date section contains in the Generate Mid-Level Release dialog box. For information about modifying this date, see "Exporting Objects to a Mid-Level Release" on page 207.

Using a Command Line to Export Only Modified Objects

This topic describes how to use a command-line interface to export modified objects.

To use a command line to export only modified objects

1 Open a command-line interface in the following folder:

```
SIEBEL_TOOLS_ROOT\BIN
```

2 Enter the following command:

```
siebdev /c tools.cfg /l enu /d datasource /u username /p password /t tag name /
MidlevelRelease "file type" "midlevel release label" "tag input file" "date and
time"
```

where:

- *datasource* identifies the same data source that you specify in the dialog box when you log in to Siebel Tools.
- *username* and *password* identifies the user name and password you use to log in to Siebel Tools.
- *tag name* identifies the tag for the current user. If object tagging is enabled, then you must specify a developer tag when you log in to Siebel Tools or if you use the command line. For example, if you are associated with multiple tags, and if one of these tags is COM, then you can use the following value:

  ```
  /t COM
  ```

  For more information about object tags, see "Managing Modifications That Developers Make to Repositories" on page 54.
- *file type* is SDF or SIF.
- *midlevel release label* identifies the name of the output folder. It is recommended that you specify an object type. For example, TABLE. For more information, see Step 4 on page 192.
- *tag input file* is an optional parameter that identifies a text file that includes a list of tag names that Siebel Tools uses to identify the objects it must extract. You must include only a single tag name on each line in this text file.
- *date and time* is an optional parameter that describes the dates and times of the computer clocks. For more information, see Step 2 on page 192.

Commands You Can Use to Export Only Modified Objects

Table 17 lists example commands that you can use to create SDF files from a command-line interface. The tags.txt file that a command references includes a list of tag names. A new line separates each tag name in this file.

Table 17. Example Commands You Can Use to Export Only Modified Objects

Task You Must Perform	Command You Use
Create an SDF file for a single tag that includes a timestamp.	`siebdev /c tools.cfg /l enu /d local /u SADMIN /p SADMIN /t "COM" /MidlevelRelease  "SDF" "Test" "D:/tags.txt" "04/01/2012 05:15:16"`
Create an SDF file for a single tag that does not include a timestamp.	`siebdev /c tools.cfg /l enu /d local /u SADMIN /p SADMIN /t "COM" /MidlevelRelease  "SDF" "Test" "D:/tags.txt"`
Create an SDF file for multiple tags that include a timestamp.	`siebdev /c tools.cfg /l enu /d local /u SADMIN /p SADMIN /t "COM" /MidlevelRelease  "SDF" "Test" "D:/tags.txt" "04/01/2012 05:15:16"`

Table 17. Example Commands You Can Use to Export Only Modified Objects

Task You Must Perform	Command You Use
Create an SIF file where one file includes all objects.	`siebdev /c tools.cfg /l enu /d local /u SADMIN /p SADMIN /t "COM" /MidlevelRelease  "SIF" "Test" "SINGLE" "04/01/2012 05:15:16"`
Create an SIF file for multiple files, where each file includes one object.	`siebdev /c tools.cfg /l enu /d local /u SADMIN /p SADMIN /t "COM" /MidlevelRelease  "SIF" "Test" "MULTI" "04/01/2012 05:15:16"`

Importing Objects from an Archive

This topic describes how to import objects from an archive. It includes the following information:

- Preparing the Siebel Tools Environment to Import Objects from an Archive on page 195
- Importing an Archive on page 196
- Importing Multiple Archives on page 198
- Using the Command Line to Import an Archive on page 199
- Importing Only Modified Objects from an Archive on page 200
- Using the Review Conflicts and Actions Dialog Box of the Import Wizard on page 202

Preparing the Siebel Tools Environment to Import Objects from an Archive

You must prepare the Siebel Tools environment before you import an archive.

To prepare the Siebel Tools environment to import objects from an archive

1. Make sure the local computer can access the import file through a network or local drive.
2. Make sure the repository you use to create the archive and the repository to which Siebel Tools imports the archive use the same schema version.

   You can export or import an archive only among repositories that use the same schema version. To determine the schema version that a repository uses, see “Viewing Information About the Current Repository” on page 49.
3. Make sure the repository is open in Siebel Tools and is the active repository.

   For more information, see “Viewing Information About the Current Repository” on page 49.
4. Make sure the projects that the import affects are checked out.

   For more information, see “Checking Out and Locking Projects Before You Import an Archive” on page 196.

Checking Out and Locking Projects Before You Import an Archive

Each object in an archive references a project. You must make sure that you check out and lock each of these projects. You can import an archive only to a project that is checked out in the local database of a client computer. Do not import to the Server database. The only exception includes projects or objects that reside in the archive but that do not exist in the repository. These projects are not checked out because they do not exist in the repository.

In some situations it might be difficult to identify the projects you must check out. The Import wizard informs you of any unlocked projects that you must lock. It does this after it analyzes the objects in the archive and compares them to the objects that reside in the repository. For more information, see "Using Checkout and Checkin" on page 89.

Importing an Archive

This topic describes how to use Siebel Tools to import an archive.

To import an archive

1. Log in to Siebel Tools.
2. Prepare the Siebel Tools environment to import an archive.

   For more information, see "Preparing the Siebel Tools Environment to Import Objects from an Archive" on page 195.
3. Click the Tools menu, and then click Import From Archive.
4. In the Select Archive To Import dialog box, choose the archive file, and then click Open.

   Siebel Tools displays the Import Wizard - Preview dialog box. It lists the projects and other top-level objects that the archive file you opened contains.
5. Specify the work that Siebel Tools does if it determines that the archive and the repository both contain the same top-level object. In the Conflict Resolution section, choose an option.

   For more information, see "Options You Can Choose to Resolve a Conflict" on page 197.
6. Click Next.

   Siebel Tools does one of the following:

   - If the objects in the archive already exist in the repository, and if Siebel Tools finds no conflicts, then it makes no modifications. It displays a dialog box that describes that it found no conflicts and made no modifications to the repository. In this situation, click OK and exit this procedure.
   - If any object involved in this import is not locked, or if any project that this object references is not locked, then Siebel Tools displays a warning message. You must cancel the import, lock the objects and projects, and then restart the Import wizard.

- If the objects in the archive already exist in the repository, and if Siebel Tools finds a conflict, or if the objects do not exist in the repository, then it displays the Import Wizard - Review Conflicts and Actions dialog box. This dialog box includes information about the differences. In this situation, go to Step 7.

  For more information about projects, see Chapter 4, “Checking Out and Checking In Projects and Objects.”

7. In the Import Wizard - Review Conflicts and Actions dialog box, in the Conflicting Objects section, choose an object.

   If you choose an object, then Siebel Tools displays differences in the Object Differences and Attribute Differences sections. Siebel CRM uses the terms attribute and property. These terms have the same meaning. For more information, see “Using the Review Conflicts and Actions Dialog Box of the Import Wizard” on page 202.

8. To make an adjustment, right-click a difference, and then choose an action.
9. Click Next.

   Siebel Tools displays the Summary dialog box and starts the import.

10. When Siebel Tools finishes the import, you can click Finish.

    Siebel Tools creates a log file named importlog.txt in the following folder:

    SIEBEL_TOOLS_ROOT\TEMP

    It includes the list of messages that it displayed in the Summary dialog box. To maintain a record of this import, you can modify the importlog.txt file name to the current date.

Options You Can Choose to Resolve a Conflict

A *conflict* is a situation that occurs if an object that Siebel Tools attempts to import from an archive does not match the corresponding object in the repository. Siebel Tools examines these objects to determine if they include the same properties, property values, and child objects. If these items are not the same, then Siebel Tools recognizes this situation as a conflict.

Table 18 describes options you can choose to resolve a conflict that occurs during an import. It describes the work that Siebel Tools does if it determines that the archive and the repository both contain the same top-level object.

Table 18. Options You Can Choose to Resolve a Conflict

Option	Description
Overwrite the Object in the Repository	Siebel Tools deletes the object in the repository and the child objects of this object, and then copy the object and child objects from the archive to the repository.
Merge the Object Definitions From the Archive With the Definition in the Repository	Siebel Tools does the following: ■ Replaces the properties that are different in the repository with the properties that the archive contains. ■ If the archive includes child objects that the repository does not contain, then it copies these objects from the archive to the repository. ■ Does not modify child objects in the repository that are not also in the archive. When Siebel Tools finishes this merge, the top-level objects in the repository include the same properties and child objects that the object in the archive includes. It also includes any child objects that already exist in the repository. Siebel Tools merges objects, by default. This option is typically the safest option.
Do Not Import the Object Definition From the Archive	Siebel Tools does not modify the objects in the repository.

Importing Multiple Archives

This topic describes how to import multiple archives.

To import multiple archives

1 Log in to Siebel Tools.

2 Prepare the Siebel Tools environment to import an archive.

For more information, see "Preparing the Siebel Tools Environment to Import Objects from an Archive" on page 195.

3 Click the Tools menu, and then click Import from Archives.

4 In the Select Archives to Import dialog box, choose one or more SIF files to import, and then click OK.

5. In the Conflict Resolution section of the Import from Archives dialog box, choose an option.

   You specify the work that Siebel Tools must do if it determines that the archive and the repository both contain the same top-level object. For more information, see "Options You Can Choose to Resolve a Conflict" on page 197.

   You can also do the following, as necessary:

   - To control the order that Siebel Tools uses to import the archive, choose a row, and then click the up arrow or down arrow.
   - To add more archive files, click Add, choose the files you must add, and then click OK.

     Siebel Tools does not add duplicate files to the import list.
   - To remove an archive, choose the archive you must remove, and then click Remove.
   - To view the objects that a SIF file includes, choose this SIF file in the SIF Files list, and then click Preview. After you finish previewing, click OK.

6. In the Import from Archives dialog box, Click Import.

   Siebel Tools begins the import in the order that you specify. You can click Cancel to cancel the import.

   Siebel Tools displays Done in the status bar at the bottom of the Import from Archive(s) dialog box when the import finishes. It displays the following status for a SIF file in the SIF Files list:

   - **Completed.** Import for this SIF file completed successfully.
   - **Failed.** Import for this SIF file encountered an error and did not complete successfully.

7. If a conflict occurs, then review the Import from Archive(s) - Review Conflicts and Actions dialog box, and then click Continue.

Using the Command Line to Import an Archive

You can use the command line to import objects from an archive. You can also use the command line to do other work. For more information, see "Using the Command Line to Import, Export, or Compile Objects" on page 183.

To use the command line to import an archive

1. In Siebel Tools, make sure the projects that the import affects are checked out or locked.

   You cannot use the command line to check out or lock these projects. For more information, see "Checking Out and Locking Projects Before You Import an Archive" on page 196.

   For more information about projects, see Chapter 4, "Checking Out and Checking In Projects and Objects."

2. Open a command line, and then navigate to the following folder:

   SIEBEL_TOOLS_ROOT`\BIN`

3. Run siebdev.exe. Use the following command:

```
siebdev.exe /c config_file /d database /u user_name /p password /batchimport
repository_name import_mode .sif file1, .sif file2, .sif fileN; or folder where
SIF files can be found log_file
```

You can use the full path or the relative path to the current folder to specify the SIF file and the log file.

Example of Using the Command Line to Import an Archive

The following example imports an archive:

```
siebdev.exe /c tools.cfg /d sample /u sadmin /p sadmin /batchimport "siebel
repository" overwrite ..\import1.sif "C:\Program
Files\Siebel\8.0\Tools\import2.sif" import.log
```

This command imports the following files:

- The import1.sif file from the parent folder
- The import2.sif file from the Siebel Tools installation folder

It uses the overwrite option for conflict resolution. It logs the results to the import.log file. For more information, see "Options You Can Choose to Resolve a Conflict" on page 197.

Example of Importing All Files

The following example imports all files:

```
siebdev.exe /c tools.cfg /d sample /u sadmin /p sadmin /batchimport "siebel
repository" merge "C:\Program Files\Siebel\8.0\Tools\importfiledir" import.log
```

This command imports all files in the following folder:

```
C:\Program Files\Siebel\8.0\Tools\importfiledir
```

It uses the merge option for conflict resolution. It logs the results to the import.log file.

Importing Only Modified Objects from an Archive

This topic describes how to use an SDF file to import only the modifications that you make to an object instead of the entire definition of this object. For more information about:

- Using an SDF file to export only the modifications that you make, see "Exporting Only Modified Objects to an Archive" on page 191.
- SDF files, see "How Siebel Tools Minimizes the Data That it Processes during Exports and Imports" on page 49.

To import one SDF file

1. Set up Siebel Tools to use iPackMode.

   For more information, see "Enabling Object Tagging in Siebel Tools" on page 57.

2 Log in to Siebel Tools.

3 Prepare the Siebel Tools environment to import an archive.

   For more information, see "Preparing the Siebel Tools Environment to Import Objects from an Archive" on page 195.

4 Click the Tools menu, and then click Import From Archive.

   If Siebel Tools is connected to the server database, then it displays a dialog box that includes a message that is similar to the following:

```
This operation should only be performed while connected to your local database.
would you like to continue anyway? Yes or NO.
```

   Siebel Tools allows you to click Yes to proceed, but it is strongly recommended that you import an archive only to your local database.

5 In the Select Archive To Import dialog box, navigate to the folder that contains the SDF files you must import.

6 In the Files of Type window, choose the following value:

```
Siebel Delta Archive files (*.sdf)
```

7 Click the archive file you must import, and then click Open.

8 In the Import Wizard - Preview dialog box, click Next.

9 In the Import Wizard - Review Conflicts and Actions dialog box, in the Conflicting Objects section, choose an object, as necessary, and then click Next.

   For more information, see "Using the Review Conflicts and Actions Dialog Box of the Import Wizard" on page 202.

10 In the Import Wizard - Summary dialog box, click Finish.

11 In the Siebel dialog box, click Yes.

12 Wait for Siebel Tools to finish the import.

   Siebel Tools displays and updates the status of the import in the Import Wizard - Summary dialog box while it does the import.

13 Make sure the Import Wizard - Summary dialog box displays a status that is similar to the following:

```
Done applying modifications to the repository at 06/18/13 02:57:47
These results are saved in the D:\Tools_08\Tools\Temp\importlog.txt file
```

14 Click Finish.

To import multiple files

1 Set up Siebel Tools to use iPackMode.

   For more information, see "Enabling Object Tagging in Siebel Tools" on page 57.

2 Log in to Siebel Tools.

3 Prepare the Siebel Tools environment to import an archive.

   For more information, see "Preparing the Siebel Tools Environment to Import Objects from an Archive" on page 195.

4 Click the Tools menu, and then click Import From Archive(s).

   Make sure you click Import From Archive(s), and not Import From Archive.

5 In the Siebel dialog box, click Yes.

6 In the Select Archive(s) To Import dialog box, navigate to the folder that contains the files you must import.

7 In the Files of Type window, choose the following value:

```
Siebel Delta Archive files (*.sdf)
```

   You can also import SDF and SIF files. To do this, choose the following value:

```
All files (*.*)
```

8 Hold down the CTRL key, and then click each of the files that Siebel Tools must import.

9 Release the CTRL key, and then click Open.

10 In the Import from Archive(s) dialog box, click Import.

   Siebel Tools displays the Import Wizard - Review Conflicts and Actions dialog box for the first file that it lists in the Import from Archive(s) dialog box. Do the following:

   a In the Conflicting Objects section, choose an object, as necessary, and then click Continue.

      Siebel Tools displays the Import from Archive(s) dialog box momentarily, and then displays the Import Wizard - Review Conflicts and Actions dialog box for the next file that it lists in the Import from Archive(s) dialog box. For more information, see "Using the Review Conflicts and Actions Dialog Box of the Import Wizard" on page 202.

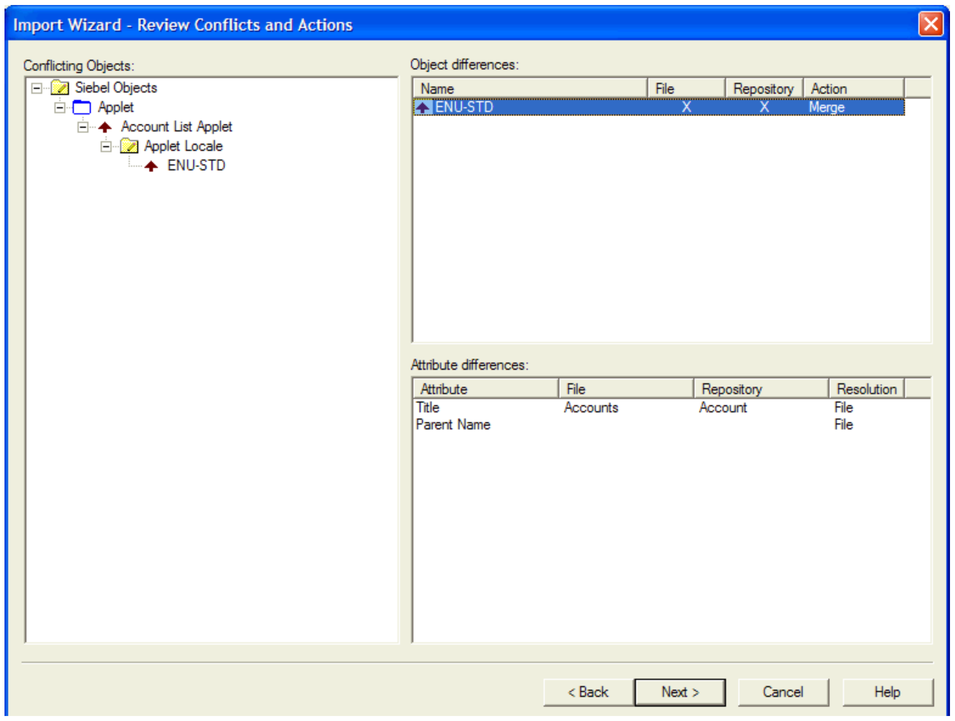
   b Click an object, as necessary, and then click Continue.

   Repeat Step a and Step b until Siebel Tools finishes importing all files.

11 In the Import from Archive(s) dialog box, click Finish.

Using the Review Conflicts and Actions Dialog Box of the Import Wizard

Figure 7 includes the dialog box that Siebel Tools shows if it determines that a difference exists between an object in the repository and an object in the archive. You can use this dialog box to review differences and to modify the work that Siebel Tools does to resolve the conflict.



Name	File	Repository	Action
ENU-STD	X	X	Merge

Attribute	File	Repository	Resolution
Title	Accounts	Account	File
Parent Name			File

Figure 7. Review Conflicts and Actions Dialog Box of the Import Wizard

Conflicting Objects Section

The Conflicting Objects section displays the hierarchy of objects that include differences. This hierarchy uses the same structure as the object type hierarchy in the repository but it displays only objects that include conflicts rather than all repository objects. The objects in the Object Differences dialog box include objects for all hierarchical levels, not just top-level objects.

Object Differences Section

The Object Differences section lists the objects that include a difference. It includes the following columns:

- **File column.** An X in this column indicates that the object exists in the archive.
- **Repository column.** An X in this column indicates that the object exists in the repository. An object can exist in the archive and in the repository. These columns provide only information. You cannot modify them.
- **Action column.** Indicates the proposed resolution for the object. Siebel Tools uses the choices you make in the Conflict Resolution section of the Preview pane to determine the work that it does to display information in this column. To modify this work, you can right-click the value in the Action column, and then choose another action. For a description of these values, see "Options You Can Choose to Resolve a Conflict" on page 197.

Attribute Differences Section

The Attribute Differences section displays the property value conflicts for the object you choose in the Object Differences section. It lists only the properties that include a conflict. Siebel CRM uses the terms attribute and property. These terms have the same meaning.

Table 19 describes the columns in the Attribute Differences section.

Table 19. Columns in the Attribute Differences Section

Column	Description
Attribute	Name of the object.
File	Value of the object in the archive.
Repository	Value of the object in the repository.
Resolution	Includes one of the following values: ■ **File.** Siebel Tools deletes the value in the repository during the import and replaces it with the value from the archive. ■ **Repository.** Siebel Tools does not modify the value in the repository. You can modify this resolution depending on the following value that the Action column in the Object Differences contains: ■ **Merge.** You can modify the resolution. To modify it, you can right-click the value in the Resolution column, and then click Repository or File. ■ **Not Merge.** You cannot modify the resolution.

Using the Application Deployment Manager

The *Application Deployment Manager* (ADM) is a feature that you can use to administer a Siebel CRM deployment. Starting with Siebel CRM release 8.0, Siebel Tools supports ADM through the following business service:

Siebel Tools Export Support for ADM

It allows you to export individual objects to a hotfix or to export all objects that Siebel CRM modified after a particular date and time to a mid-level release. For more information, see *Siebel Application Deployment Manager Guide*.

Using a Hotfix to Deploy Fixes to the Production Environment

This topic describes how to export objects to a hotfix. A *hotfix* is a Siebel Tools feature you can use to quickly update the production environment. You can use it to update a small number of object definitions, and then export and migrate them to the target enterprise. It is most useful if you must fix a small programming error. For example, if you discover an error in the script of a business service method that you must quickly fix and deploy to the production environment in a short time frame.

To use a hotfix to deploy fixes to the production environment

1. In the Object List Editor, locate the object you must modify.

   For more information, see "Locating and Modifying Object Definitions in the Object List Editor" on page 24.

2. Right-click the object you located in Step 1, and then click Add to Hot-Fix.

   Siebel Tools displays the Generate Hot-Fix dialog box and adds the object you choose to the Objects to Include in Hot-Fix list. You can export a workflow process or task UI only if the status property of the workflow process or task UI is Completed.

3. If you must add more objects, then repeat Step 1 and Step 2 for each object. Do not close the Generate Hot-Fix dialog box while you add these objects.

   You can add objects of the same object type or you can add objects of another object type.

4. Add text in the Hot-Fix Label window that describes this hotfix.
5. Click Export.

   Siebel Tools displays a dialog box that confirms that it successfully created the hotfix in the following folder:

   SIEBEL_TOOLS_ROOT\ADM*hot_fix_label*

   This folder includes a SIF file, an XML description of the hotfix contents, and a log file.

6. Click OK.

Using the Command Line to Export Objects to a Hotfix

This topic describes different ways to use the command line to export objects to a hotfix.

Sending All of the Arguments in the Command Line for a Hotfix

This topic describes how to send all of the arguments in the command line.

To send all of the arguments in the command line for a hotfix

- Open a command line, and then navigate to the following folder:

```
SIEBEL_TOOLS_ROOT\BIN
```

You use the following command for the consoleapp executable to access the command line. You cannot include a space before or after a comma:

```
consoleapp SIEBEL_TOOLS_ROOT\BIN\ENU\config_file.cfg language username password
"BusinessServiceName" "MethodName:argument_list"
```

For example:

```
consoleapp "C:\Siebel\8.0\Tools\BIN\ENU\tools.cfg" ENU SADMIN SADMIN "Siebel
Tools Export Support for ADM" "Export:Repository=Siebel Repository,
LogFile=C:\Siebel\8.0\Tools\ADM\admtest\admtest.log,
ExportFile=C:\Siebel\8.0\Tools\ADM\admtest\admtest.sif,
DescriptorFile=C:\Siebel\8.0\Tools\ADM\admtest\admtest_desc.xml,Object_1=Accoun
t List Applet,Type_1=Applet,ExportCount=1"
```

Sending Some of the Arguments in an XML File for a Hotfix

The command you use to send some of the arguments in an XML file is similar to the command you use in "Sending All of the Arguments in the Command Line for a Hotfix" on page 205, but instead of naming the business service and the method name, and including a list of arguments, you use the /f switch and provide an XML file that includes the business service name and parameters for the business service methods.

To send some of the arguments in the command line for a Hotfix

1 Open a command line, and then navigate to the following folder:

```
SIEBEL_TOOLS_ROOT\BIN
```

2 Enter the following command. You cannot include a space before or after a comma:

```
consoleapp <SIEBEL_TOOLS_ROOT\BIN\ENU\config_file.cfg language username password
/f export_argument_file.xml
```

where:

- *export_argument_file.xml* includes the following code:

```
<BusinessService Name="Siebel Tools Export Support for ADM" Method="Export">
  <Param Name="Repository" Value="Siebel Repository"/>
  <Param Name="LogFile" Value="C:\Siebel\8.0\Tools\ADM\admtest2\admtest2.log"/>
  <Param Name="ExportFile"Value="C:\Siebel\8.0\Tools\ADM\admtest2\admtest2.sif"/>
  <Param Name="DescriptorFile"
     Value="C:\Siebel\8.0\Tools\ADM\admtest2\admtest2_desc.xml"/>
  <Param Name="ExportCount" Value="3"/>
  <ExportObjects>
     <Object Name="Account List Applet" Type="Applet"/>
     <Object Name="Account" Type="Business Component"/>
     <Object Name="Contact" Type="Business Component"/>
  </ExportObjects>
</BusinessService>
```

For example:

```
consoleapp "C:\Siebel\8.0\Tools\BIN\ENU\tools.cfg" ENU SADMIN SADMIN /f
"C:\Siebel\8.0\Tools\ADM\admtest2\exportargs.xml"
```

Exporting Objects to a Mid-Level Release

This topic describes how to export objects to a mid-level release. A *mid-level release* is a type of release that includes objects you modify from a time frame that you specify. For example, assume from January 15 at 9:00 AM to January 30 you create and modify several dozen repository objects. You can use a mid-level release to export only these objects to an archive. Siebel CRM considers modifications that you make to the database schema as part of a major release. Examples of a database schema modification include creating a new table or adding a column to an existing table. You must not use ADM to migrate these modifications.

To export objects to a mid-level release

1. Log in to Siebel Tools.
2. Set the starting date that Siebel Tools uses to include objects in the mid-level release:
   a. Click the View menu, and then click Options.
   b. On the General tab, set values in the Changed Date section.

      When you create a mid-level release in Step 8, Siebel Tools exports only objects that Siebel CRM modified since the date you enter.
3. Click the Tools menu, and then click Generate Mid-Level Release.

   Siebel Tools displays the Generate Mid-Level Release dialog box and sets the value in the Start Date field to the value you set in Step 2. For more information, see "Using the Changed Property" on page 29.
4. (Optional) To remove an object from the list, click it, and then press the DELETE key.

   To choose more than one record, see "Choosing More Than One Record in the Object List Editor" on page 44.
5. (Optional) To display a list of the modified objects, click Generate List.
6. In the Mid-Level Release Label field, type a name for the export, and then click Generate List.

   Siebel Tools displays the objects that you modified since the Start Date in the Objects to Include in Mid-Level Release section.
7. In the Export Options section, click one of the following:
   - **Single SIF for All Object.** Export to a single SIF file.
   - **One SIF Per Object.** Export to multiple SIF files.
8. Click Export.

   Siebel Tools displays a dialog box that confirms that it successfully created the archive in the following folder:

   SIEBEL_TOOLS_ROOT\ADM*mid_level_release_label*

A Reference Materials for Siebel Tools

This appendix includes reference information for Siebel Tools. It includes the following topics:

- Menus and Menu Items on the Menu Bar on page 209
- Buttons on the Toolbars on page 222
- Buttons in the Palettes Window on page 229
- Dialog Boxes That Check Out and Check In Projects and Objects on page 230
- Dialog Boxes That Validate Objects on page 237
- Dialog Boxes That Compare Objects on page 239
- Dialog Boxes That Set Development Options on page 241
- Parameters That Convert Symbolic Strings on page 245

Menus and Menu Items on the Menu Bar

This topic describes the menus and menu items that you can click from the menu bar in Siebel Tools. For more information, see "Using the Menu Bar" on page 31.

This topic includes the following information:

- File Menu on page 210
- Edit Menu on page 211
- View Menu on page 212
- Screens Menu on page 214
- Go Menu on page 215
- Query Menu on page 215
- Format Menu on page 216
- Debug Menu on page 216
- Tools Menu on page 217
- Window Menu on page 220
- Help Menu on page 221

File Menu

Table 20 describes the menu items that you can click from the File menu. To use this menu, you click the File menu on the Menu Bar.

Table 20. Menu Items That You Can Click from the File Menu

Menu Item	Description
Open Repository	If multiple repositories reside in the development database, then this menu item allows you to open a repository other than the repository that is currently open. Siebel Tools sets the repository you choose from the Open Repository menu item as the default repository that it opens each time you open Siebel Tools.
New Object	Starts the New Object Wizard that allows you to create a list applet, form applet, chart applet, tree applet, business component, report, table, command, picklist, MVG, or view.
Close (CTRL+F4)	Closes the Object List Editor.
Save (CTRL+S)	Saves modifications that the current editing window contains if you edit in one of the following windows: Layout, Menu, or Basic Scripts.
Save All	Saves modifications in all open editing windows.
Import	Imports text from an external text file into the Siebel VB Editor window. This text must use an SBL file format. Siebel Tools creates this format when it exports text from the Siebel VB editor.
Export	Allows you to create a text file in delimited or HTML format that lists the property values of an object or all objects that the Object List Editor currently shows.
Print Setup	Modifies the printer and printing options for printing diagrams from the object visualization view.
Print Preview	Opens a print preview window that displays an object visualization view.
Print (CTRL+P)	Prints the active object visualization view diagram.
Exit	Closes Siebel Tools.

Edit Menu

Table 21 describes the menu items that you can click from the Edit menu. The Edit menu items apply to individual objects in the Object List Editor. You can right-click an object in the Object List Editor to display a list of menu items. For more information, see "Using a Layout Editor" on page 42.

Table 21. Menu Items That You Can Click from the Edit Menu

Menu Item	Description
Undo (CTRL+Z)	Reverses the last modification you made to a property value in the Object List Editor or the Property window before you save the object.
Redo (CTRL+Y)	Reapplies modifications after the Undo command runs.
Undo Delete	Siebel Tools displays this menu item if you delete a record in the Object List Editor. It allows you to undo the deletion.
Undo Record	If you have not saved the record, then this menu item removes a new object that you create or reverses modifications you made to an existing object.
New Record (CTRL+N)	Creates a new object in the Object List Editor and positions the cursor in the first required property.
Copy Record (CTRL+B)	Creates a new object that is a copy of the object that you choose. It also creates copies of all child objects. It is recommended that you use the Copy Record menu item only if reusing an existing object is not practical.
Delete Record (CTRL+D)	Deletes the object that you choose and the child objects of the object you choose. It is recommended that you do not use the Delete Record menu item. Instead, use the Inactive property. For more information, see "Using the Inactive Property" on page 30.
Cut (CTRL+X)	If you use this menu item while your cursor is in a property that contains text, then Siebel Tools copies the text you choose to the clipboard and deletes the existing text. In the Applet Designer, it copies the control you choose to the clipboard and deletes the existing control.
Copy (CTRL+C)	If you use this menu item while your cursor is in a property that contains text, then Siebel Tools copies the text you choose to the clipboard. In the Applet Designer, it copies the control you choose to the clipboard.
Paste (CTRL+V)	Inserts text from the clipboard into a property at the insertion point. Inserts a control from the clipboard in the Applet Designer.
Delete (DEL)	If you use this menu item while your cursor is in a property that contains text, then Siebel Tools deletes the text you choose. In the Applet Designer, it deletes the control you choose.
Select All (CTRL+A)	Chooses all items. In the Applet Designer, it chooses all controls that the applet contains.
Change Records	Modifies multiple records simultaneously.

Table 21. Menu Items That You Can Click from the Edit Menu

Menu Item	Description
Find (CTRL+F)	Finds the text that you specify in the Siebel Script Editor window.
Replace (CTRL+H)	Replaces the text that you specify with different text in the Siebel Script Editor window.

View Menu

Table 22 describes the menu items that you can click from the View menu. You use these menu items to display windows, toolbars, or visualization views.

Table 22. Menu Items That You Can Click from the View Menu

Menu Item	Subitem	Description
Windows	Palette	Displays the Palettes window.
	Properties Window	Displays the Properties window.
	Applets Window	Displays the Applets window.
	Controls Window	Displays the Controls/Columns window.
	Bookmarks Window	Displays the Bookmarks window.
	Web Templates Window	Displays the Web Templates Explorer window.
	Multi Value Properties Window	Displays the Multi Value Property Window.
	Refresh Windows	Requeries and updates the state of dockable windows.
	Reset Windows	Closes all dockable windows except the Object Explorer for the currently active editor. Does not close editor windows.
Editors	Web Applet Editor	Opens the applet you choose in the Applet Layout Editor, including the Controls/Columns and Palettes windows.
	Server Script Editor	Opens the Siebel Script Editor. You can specify the editor to use or you can use the default.
	Browser Script Editor	Opens the Siebel Web Script Editor that you use to access the scripts that control the presentation and behavior of applet controls and list columns in a Web applet template.

Table 22. Menu Items That You Can Click from the View Menu

Menu Item	Subitem	Description
Visualize	View Details	For more information, see "Viewing Object Relationships" on page 113.
	View Relationships	
	View Descendents	
	View Web Hierarchy	
Debug Windows	Calls (CTRL+L)	Opens the Calls window. This window displays the call stack of the Siebel VB script or the Siebel eScript script that you are currently debugging.
	Watch (SHIFT+F9)	Opens the Watch window. This window displays the values of local variables for items you are currently debugging, such as Siebel VB script, Siebel eScript, or Siebel Workflow.
	Errors	Opens the Errors window. This window displays the run-time errors in the Siebel VB script or Siebel eScript script that you are currently debugging.
Preview	Not applicable	Displays a preview of a Web view layout. This preview approximates how Siebel CRM displays the container page, screen bar, and view bar.
ActiveX Methods		Allows you to view the methods for the current ActiveX control in the Applet Designer.
Toolbars		Displays the following toolbars: Edit, History, List, Debug, Web Controls, and Configuration Context.
Status Bar		Displays the Status bar at the bottom of the Siebel Tools window.
Object Explorer (CTRL+E)		Displays the Object Explorer.
Options		Opens the Development Tools Options dialog box. For more information, see "Development Options for Debugging" on page 244.

Screens Menu

Table 23 describes the menu items that you can click from the Screens Menu. Siebel Tools displays the Screens menu only if you log on to Siebel Tools as a system administrator.

Table 23. Menu Items That You Can Click from the Screens Menu

Menu Item	Subitem	Description
Application Upgrader	Application Upgrade Object List	Displays the Application Upgrades list, Object Differences list, or Attribute Differences list in the Object List Editor. In the Attribute Differences list, you can right-click and select an option to show critical conflicts, non-critical conflicts, and all changes for the item selected in the Object Differences list.
	Application Upgrade Database Version	For internal Oracle use.
	Application Upgrade Attribute List	Displays the Application Upgrades list and the Attribute Differences list in the Object List Editor. In the Attribute Differences list, you can right-click and select an option to show critical conflicts, non-critical conflicts, and all changes for the item selected in the Application Upgrades list.
System Administration	System Preferences	Displays system preferences in the Object List Editor. This information is similar to the information that Siebel CRM shows in the System Preferences view in the Administration - Application screen in the Siebel client.
	Analytics Strings	For internal Oracle use.
	List of Values	Displays the lists of values that the development database contains.

Go Menu

Table 24 describes the menu items that you can click from the Go menu. The Go menu allows you to navigate records in a list.

Table 24. Menu Items That You Can Click from the Go Menu

Menu Item	Description
Back	Displays the previous screen in a sequence.
Forward	Displays the next screen in a sequence.
Previous Record (CTRL+UP)	Navigates to the first object that is above the current selection.
Next Record (CTRL+DOWN)	Navigates to the first object that is below the current selection.
First Record (CTRL+PAGE UP)	Navigates to the first object in the list.
Last Record (CTRL+PAGE DOWN)	Navigates to the last object in the list.
Add Bookmark	Displays the Add Bookmark dialog box that allows you to create a bookmark to the object you choose. You can use this menu item to create a *bookmark*, which is a feature that allows you to return directly to an item.
Bookmark List	Displays the Bookmarks dialog box that allows you to choose an existing bookmark. You can also use this dialog box to rename or delete an existing bookmark.

Query Menu

Table 25 describes the menu items that you can click from the Query menu. This menu allows you to create and refine an Object List Editor query. You can use this query to filter the list of objects that Siebel Tools shows in the current Object List Editor.

Table 25. Menu Items That You Can Click from the Query Menu

Menu Item	Description
New Query (CTRL+Q)	Allows you to filter the set of objects that Siebel Tools shows in the Object List Editor.
Refine Query (CTRL+R)	Allows you to add more filters to the current query.

Table 25. Menu Items That You Can Click from the Query Menu

Menu Item	Description
Execute Query (ENTER)	Runs the query.
Sort Order	Opens the Sort Order dialog box. This dialog box allows you to specify the order that the Object List Editor uses to display the records.

Format Menu

Table 26 describes the menu items that you can click from the Format menu. This menu in the Applet Layout Editor allows you to position controls, configure the grid, and adjust tab or list column order.

Table 26. Menu Items That You Can Click from the Format Menu

Menu Item	Description
Align	Aligns the items you choose.
Make Same Size	Makes all items you choose the same size.
Horizontal Spacing	Adjusts horizontal spacing between items.
Vertical Spacing	Adjusts vertical spacing between items.
Center in Applet	Centers the items you choose horizontally or vertically.
Set Label Alignment	Aligns labels in applets for Web templates that use a grid layout.
Set Tab Order	Sets the tab order for fields in a form applet. This item is not available for list applets.

Debug Menu

Table 27 describes the menu items that you can click from the Debug menu. This menu allows you to control the debugger for Siebel VB or Siebel eScript.

Table 27. Menu Items That You Can Click from the Debug Menu

Menu Item	Description
Check Syntax	Compiles the current script and verifies syntax.
Start (F5)	Starts the Siebel client. It also displays a dialog box that includes startup parameters.
Break (CTRL+BREAK)	Stops the script that is currently running. If Siebel VB or Siebel eScript is not running, then Siebel Tools does nothing.
End	Stops the Siebel client and returns to the Siebel Script Editor window.

Table 27. Menu Items That You Can Click from the Debug Menu

Menu Item	Description
Restart (SHIFT+F5)	Restarts the Siebel client if a break occurs.
Toggle Breakpoint (F9)	Sets or removes a breakpoint on a line of code.
Clear All Breakpoints (CTRL+SHIFT+F9)	Removes all breakpoints from the current script.
Watch (SHIFT+F9)	Displays script variables and their values. For more information, see "Using the Watch Window While it Monitors a Script" on page 136.
Calls (CTRL+L)	Contains a list of subroutine and function calls that Siebel Tools runs prior to the current line. If you choose an item in this list, then the interpreter shifts to this item.
Step Into (F8)	Runs the next line of a script. If this line calls a subroutine or procedure, then Siebel Tools runs this subroutine or procedure.
Step Over (SHIFT+F8)	Runs the first line of script that occurs after the current subroutine or procedure. Siebel Tools continues to run this script in the current subroutine or procedure.
Step To Cursor (CTRL+F8)	Runs all lines of code up to the line that includes the cursor.

Tools Menu

Table 28 describes the menu items that you can click from the Tools menu.

Table 28. Menu Items That You Can Click from the Tools Menu

Menu Item	Subitem	Description
Compile (F7)		Opens the Object Compiler dialog box to compile one or more projects to a repository. For more information, see "Compiling Your Modifications" on page 179.
Compile Selected Objects (CTRL+F7)		Opens the Object Compiler dialog box to compile the objects you choose to a repository.
Check Out (F10)		Opens the Check Out dialog box to copy one or more projects from the Siebel Server to the local database. For more information about projects, see Chapter 4, "Checking Out and Checking In Projects and Objects."
Check In (CTRL+F10)		Opens the Check In dialog box to copy one or more projects from the local database to the Siebel Server.

Table 28. Menu Items That You Can Click from the Tools Menu

Menu Item	Subitem	Description
Lock Project (ALT+L)		Locks the project that the object you choose references.
Unlock Project (ALT+U)		Unlocks the project that the object you choose references.
Add To Archive		Opens the Export To Archive dialog box to add the top-level objects that you choose or projects to an archive.
Import From Archive		Starts the Import wizard to import objects from an archive.
Compare Objects	Selected	Compares two objects that you choose. It uses a list of object names and properties to display similarities and differences.
	Selected vs. Repository	Compares the object you choose to the corresponding object in the repository and displays similarities and differences.
	Selected vs. Archive	Compares the object you choose to the corresponding object in an archive and displays similarities and differences.
	Archive vs. Archive	Compares two files that you choose and displays similarities and differences.
Convert to Grid Layout		Converts a form applet that uses a nongrid layout to grid layout.
Search Repository		Opens the Search Repository dialog box to search for objects according to the object name, another property, or the object type.
Validate Object		Validates the object you choose. Lists errors according to severity, rule number, object name, and error description. Allows you to modify options for rules, severity, and enforcement.

Table 28. Menu Items That You Can Click from the Tools Menu

Menu Item	Subitem	Description
Upgrade	Maintenance Update	Not applicable starting with Siebel CRM version 8.0.
	Prepare Repository	Used to upgrade from a Siebel CRM version that occurs before version 7.x to version 8.0. The Prepare Repository utility runs before it does a repository merge. It migrates strings from the S_MSG table, merges labels and fields, and merges templates to applets for the language that you specify. For more information, see *Siebel Database Upgrade Guide*.
	Migrate ICL Objects to Standard	Applicable if you choose the Incorporate Custom Layout (ICL) option. You choose this option to preserve the layouts of custom objects during a previous upgrade. Before you can perform a subsequent upgrade, you must migrate the ICL objects to the predefined repository. For more information, see "About Predefined Objects" on page 24 and *Siebel Database Upgrade Guide*.
	Upgrade Application	Displays the Application Objects Upgrade List in the Application Upgrader screen of Siebel Tools and opens the Merge Repositories dialog box. You use this menu item to merge predefined and custom repositories. For more information, see *Siebel Database Upgrade Guide*.
	Generate EIM Processing Columns	Opens the EIM Processing Column Generator dialog box. You can use this dialog box to create missing EIM processing columns and indexes after you merge the repository.
	Web Client Migration	Used to upgrade from Siebel CRM version 6.x to version 7.x or version 8.0. It associates Web templates to a group of applets and views so that Siebel CRM can use them in the Siebel client. For more information, see *Siebel Database Upgrade Guide*.

Table 28. Menu Items That You Can Click from the Tools Menu

Menu Item	Subitem	Description
Utilities	Generate Help IDs	Oracle uses this subitem internally to create the sshelp.hm file for Tools Online Help. This file includes information about context Id numbers and text help identifiers that the Help Id objects specify.
	Locale Management	Allows you to use the Local Management Utility to import or export translatable strings and locale properties.
	Map Fax Properties	If you click business component in the Object Explorer, then this menu item opens the Map Fax Properties dialog box for the business component that you choose. You can use this dialog box to create mappings between fields in the business component and sheet properties for the fax software. These mappings allow you to customize the fax cover sheet and the fax message.
	Export View Previews	Exports the view that the Preview mode of the View Layout Editor shows to an HTML file.
	Case Insensitivity	Opens the Case and Accent Insensitivity Wizard. It allows you to do a case-insensitive or accent-insensitive search on columns in the Siebel schema. For more information, see *Configuring Siebel Business Applications* and *Siebel Database Upgrade Guide*.
	Build Patch	Starts the Patch Builder wizard that allows you to create a patch file.
	Apply Patch	Opens the Apply Patch window that allows you to apply the patch.

Window Menu

The Window menu lists the currently open Object List Editor, Application Designer, visualization view, and other windows, and allows you to navigate to windows that Siebel Tools does not currently show. If one of these windows is open, then Siebel Tools displays the Close menu item as the first item. The Close menu item closes the window that is currently active.

Help Menu

Table 29 describes the menu items that you can click from the Help menu.

Table 29. Menu Items That You Can Click from the Help Menu

Menu Item	Description
Contents	Opens the Siebel Tools Online Help.
Using Help	
Technical Support	Displays the Technical Support Information dialog box that includes information that Technical Support might require, such as the version number of your Siebel Tools installation.
About Record	Opens a dialog box that displays information about the current object, including the object creator and creation date.
About SRF	Opens a dialog box that displays information about the most recent full incremental compile.
About View	Opens a dialog box that displays information about the current screen, business object, and view, including applet layout.
About Visible Views	Displays the list of views in the repository and if each view is visible. Visibility for each view depends on the license key you use when you install Siebel Tools. For more information, see "Description of the About Visible Views Dialog Box" on page 221.
About Siebel Tools	Opens a dialog box that identifies the version of Siebel Tools.

Description of the About Visible Views Dialog Box

Table 30 describes the information that the About Visible Views dialog box shows. To view this dialog box, you click the Help menu and then click About Visible Views.

Table 30. Description of the About Visible Views Dialog Box

Column	Description
View Name	Displays the view name. The views in this column are not the predefined views in Siebel Tools.
Visible	Shows whether or not the view is visible in Siebel Tools.
Application	Shows whether or not the view is associated with Siebel Tools.
View	Shows whether or not the view always appears as a view in Siebel Tools.
Responsibility	Shows whether or not the view is associated with user responsibilities.

Table 30. Description of the About Visible Views Dialog Box

Column	Description
License	Shows whether or not the view visibility depends on the license key that you use when you install Siebel Tools.
Platform	Shows whether or not the view visibility depends on the platform that Siebel Tools uses.

Buttons on the Toolbars

This topic describes the buttons that you use on the toolbars in Siebel Tools. For more information, see "Using the Toolbars" on page 40.

This topic includes the following information:

- Edit Toolbar on page 222
- List Toolbar on page 223
- History Toolbar on page 224
- Debug Toolbar on page 225
- Simulate Toolbar on page 226
- Format Toolbar on page 226
- WF/Task Editor Toolbar on page 228
- Configuration Context Toolbar on page 228

Edit Toolbar

Table 31 describes the buttons that you can click on the Edit toolbar.

Table 31. Buttons You Can Click on the Edit Toolbar

Button	Description	
📄	New	Starts the New Object Wizard that allows you to create applets, views, charts, and other objects.
💾	Save	Saves modifications in the current editing window if you use Layout, Menu, or Basic Scripts.
💾	Save All	Saves modifications in all open editing windows.

Table 31. Buttons You Can Click on the Edit Toolbar

Button	**Description**	
✂	Cut	If you use this button while the cursor is in a property that contains text, then Siebel Tools copies the text you choose to the clipboard and deletes the existing text. In the Applet Designer, it copies the control you choose to the clipboard and deletes the existing control.
⧉	Copy	If you use this button while the cursor is in a property that contains text, then Siebel Tools copies the text you choose to the clipboard. In the Applet Designer, it copies the control you choose to the clipboard.
📋	Paste	Inserts text from the clipboard to a text property at the insertion point. In the Applet Designer, it inserts a control from the clipboard.
↶	Undo	If you have not saved the object, then this button reverses the last modification you made to a property value in the Object List Editor or Property window.
↷	Redo	Reapplies modifications after the Undo command runs.

List Toolbar

Table 32 describes the buttons that you can click on the List toolbar.

Table 32. Buttons You Can Click on the List Toolbar

Button	**Description**	
▤	Add New Record	Creates a new object in the Object List Editor and positions the cursor in the first required property.
⏮	First Record	Goes to the first object in the list.
◀	Previous Record	Goes to the object above the current selection.
▶	Next Record	Goes to the object below the current selection.
⏭	Last Record	Goes to the last object in the list.
🔍	New Query	Allows you to specify one or more filters on the set of objects that the Object List Editor shows.

Table 32. Buttons You Can Click on the List Toolbar

Button	Description	
🔍	Execute Query	Runs the query you specify. This button does the same as pressing ENTER.
A Z ↓	Sort Ascending	Modifies the order that Siebel Tools uses to display objects. It sorts them in ascending order according to the currently chosen property column.
Z A ↓	Sort Descending	Modifies the order that Siebel Tools uses to display objects. It sorts them in descending order according to the currently chosen property column.
▽	Filter Version	Displays only the most recent version of each workflow process or task UI that the Object List Editor shows.

History Toolbar

Table 33 describes the buttons that you can click on the History toolbar.

Table 33. Buttons You Can Click on the History Toolbar

Button	Description	
⬅	Go Back	Returns to the previously displayed screen.
➡	Go Forward	Returns to the subsequent displayed screen.
+📌	Add Bookmark	Displays the Add Bookmark dialog box that allows you to create a bookmark to the object you choose. You can use this menu item to create a *bookmark*, which is a feature that allows you to return directly to an item.
☰📌	Bookmark List	Displays the Bookmarks dialog box that allows you to choose an existing bookmark. You can also use this dialog box to rename or delete an existing bookmark.

Debug Toolbar

Table 34 describes the buttons that you can click on the Debug toolbar.

Table 34. Buttons You Can Click on the Debug Toolbar

Button	Description	
	Check Syntax	Compiles the current script and verifies syntax.
▶	Start	Starts the Siebel client. It also displays a dialog box that includes startup parameters.
❚❚	Break	Stops the script that is currently running. If Siebel VB or Siebel eScript is not running, then Siebel Tools does nothing.
■	End	Stops the Siebel client and returns to the Siebel Script Editor window.
✋	Toggle Breakpoint	Sets or removes a breakpoint on a line of code.
👓	Watch	Displays script variables and their values. For more information, see "Using the Watch Window While it Monitors a Script" on page 136.
	Calls	Contains a list of subroutine and function calls that Siebel Tools runs prior to the current line. If you choose an item in this list, then the interpreter shifts to this item.
	Step Into	Runs the next line of a script. If this line calls a subroutine or procedure, then Siebel Tools runs this subroutine or procedure.
	Step Over	Runs the first line of script that occurs after the current subroutine or procedure. Siebel Tools continues to run this script in the current subroutine or procedure.

Simulate Toolbar

Table 35 describes the buttons that you can click on the Simulate toolbar.

Table 35. Buttons You Can Click on the Simulate Toolbar

Button	Description	
▶	Start Simulation	Starts the simulation of a workflow process.
❙▶	Simulate Next	Simulates the next workflow process step.
▶❙	Complete Simulation	Completes the simulation of a workflow process.
🟥	Stop Simulation	Stops the Workflow Simulator.

Format Toolbar

Table 36 describes the buttons that you can click on the Format toolbar.

Table 36. Buttons You Can Click on the Format Toolbar

Button	Description
	Aligns the left edges of controls.
	Aligns the centers of controls along a vertical axis.
	Aligns the right edges of controls.
	Aligns the tops of controls.
	Aligns the middles of controls along a horizontal axis.
	Aligns the bottom of controls.
	Makes the controls the same width.

Table 36. Buttons You Can Click on the Format Toolbar

Button	Description
	Makes the controls the same height.
	Makes the controls the same size.
	Makes the horizontal spacing between controls equal.
	Increases the horizontal spacing between controls.
	Decreases the horizontal spacing between controls.
	Removes the horizontal spacing between controls.
	Makes the vertical spacing between controls equal.
	Increases the vertical spacing between controls.
	Decreases the vertical spacing between controls.
	Removes the vertical spacing between controls.
	Centers the controls vertically.
	Centers the controls horizontally.
	Aligns the labels to the left.
	Centers the labels.
	Aligns the labels.

WF/Task Editor Toolbar

Table 37 describes the buttons that you can click on the WF/Task Editor toolbar.

Table 37. Buttons You Can Click on the WF/Task Editor Toolbar

Button	Description	
	Publish/Activate	Publishes and activates a workflow process or task UI in a single step. This button is available only with the Siebel Web Client. You cannot use it to activate a workflow process or task UI in the production environment.
	Publish	Makes a workflow process or task UI available to activate from the Siebel client.
	Revise	Revises a workflow process or task UI.
	Expire	Makes a workflow process or task UI inactive.

Configuration Context Toolbar

Table 38 describes the items that you can use on the Configuration Context toolbar.

Table 38. Items You Can Use on the Configuration Context Toolbar

Drop-Down List	Description
Target Browser	Allows you to choose a target browser for layout editing and for scripting.
Application	Allows you to configure objects for a specific Siebel application. Typically, you use All Applications. If you choose a single application from the list, then you can configure objects, such as applets or views, to show or behave differently for only the application you choose.
Interactivity	Allows you to choose High Interactivity or Standard Interactivity. You can configure Web layouts differently, depending on the type of interactivity you choose.
Variable	Allows you to specify a display style for an applet for previewing, such as parent, child, or grandchild. Siebel Tools might display an applet differently depending on the underlying Web template. For example, an applet header might not appear if Siebel Tools displays it as a grandchild.

Buttons in the Palettes Window

Table 39 describes the buttons that the Palettes window shows when you use the Applet Layout Editor. For more information, see "Using the Palettes Window" on page 37.

Table 39. Buttons the Palettes Window Shows with the Applet Layout Editor

Button	Description
☑	CheckBox. Creates an option.
◉	RadioButton. Creates a radio button.
▬	MiniButton. Creates a mini button.
▭	Field. Creates a field.
Aa:	FieldLabel. Creates a field label.
▤	ComboBox. Creates a combo box.
◁▶	RecNavNxt. Creates a control for navigating to the next record.
◀▷	RecNavPrv. Creates a control for navigating to the previous record.
abl	Text. Creates a text box.
▦	TextArea. Creates a text area.
☰	FormSection. Creates a section of a form.
ab	Hidden. Creates hidden HTML.
**	Password. Creates a text box where the user enters a password during logon.
🌐	Link. Creates an HTML link control.

Table 39. Buttons the Palettes Window Shows with the Applet Layout Editor

Button	Description
🖨	MailTo. Creates a mail-to link.
▭	Button. Creates a button.
A	Label. Creates a label on templates.
⌘	URL. Creates a link to an external URL on a template.
☻	ActiveX. Creates an ActiveX control on a template.
▦	Text List Column. Creates a list column that contains HTML text. Available only for a list applet.
☑	Checkbox List Column. Creates a list column that contains HTML options. Available only for a list applet.
✥	Custom Control. Creates a custom control on a template. To create the custom control, you can choose a custom control from the Control Type drop-down list, and then drag the Custom Control button to the designer.

Dialog Boxes That Check Out and Check In Projects and Objects

This topic describes elements of the dialog boxes that you use to check out and check in projects and objects. For more information about projects, see Chapter 4, “Checking Out and Checking In Projects and Objects.”

This topic includes the following information:

- Elements of the Project Check Out Dialog Box on page 231
- Elements of the Object Check Out Dialog Box on page 234
- Elements of the Check In Dialog Box on page 235

Elements of the Project Check Out Dialog Box

Table 40 describes the elements of the Project Check Out dialog box. This dialog box lists the projects that you can check out. It does not list individual objects in projects. For information about using this dialog box, see "Checking Out Projects from the Server Repository" on page 90.

Table 40. Elements of the Project Check Out Dialog Box

Element	Description
Repository drop-down list	Displays a list of repositories that the Siebel Server contains. The list of projects in the projects list displays the list of projects that reside in the server repository that you choose. If you choose a server repository that is different from the repository that is currently open in Siebel Tools, then Siebel Tools displays a warning and you must Get all projects or choose another repository.

Table 40. Elements of the Project Check Out Dialog Box

Element		Description
Projects list	Project	Displays the name of each project that the server repository contains.
	Updated	If the Siebel Server Locked By and Locked Date are different from the client version, then Siebel Tools displays a value of Yes. This value indicates that the local version of the project is not synchronized with the version that the Siebel Server contains.
	Server Locked By	Displays the Logon Id of the developer who most recently checked out this project on the Siebel Server.
	Server Locked Date	Displays the date of checkout.
	Client Locked By	Displays the logon Id of the developer who most recently locked this project locally.
	Client Language	Displays the language of the project that is currently locked in Siebel Tools. You can lock only one language at one time.
	Allow Object Locking	If the project allows object checkin or checkout, then Siebel Tools displays a value of Yes. The default value is Yes. For more information, see "Allowing Object Locking for a Project" on page 89.
	Owner Branch	Displays the owner branch for each project. If the project Owner Branch is not empty, then the Repository Branch that Siebel CRM assigns for this user must match. If these values do not match, then this user cannot check out the project or any objects that reference this project. Siebel Tools does not display this column in the Object List Editor, by default. To display it, you can do the following: ■ In the Object Explorer, click Project. ■ In the Projects list, right-click, and then click Columns Displayed. ■ In the Columns Displayed dialog box, move the Owner Branch column to the Displayed Columns window, and then click OK.

Table 40. Elements of the Project Check Out Dialog Box

Element		Description
Option buttons	Selected projects	Allows you to choose individual projects to check out or Get.
	All projects	If this check box contains a check mark, then Siebel Tools chooses all projects in the repository to check out or Get.
	Updated projects	If this option is active, then Siebel Tools chooses only projects that include an Updated value of Yes. This configuration allows you to check out or Get only those projects that reside on the Siebel Server that are new or are different from corresponding projects in the local repository. You typically do a Get to update your local repository.
Get Locale Specific Data Only		If this check box contains a check mark, then Siebel Tools gets only string translations and locale properties that Siebel CRM stores in the locale objects. It does not get data that Siebel CRM stores in the parent object of the locale object.
Buttons	Get	Copies the projects that you choose to the local repository. It replaces any versions of these projects that already exist in the local repository but it does not lock these projects on the Siebel Server. You can Get any projects from the Siebel Server, including projects that someone else locked. For more information, see "Allowing Object Locking for a Project" on page 89.
	Check Out	Copies all objects in the projects you choose to the local repository and locks them on the Siebel Server and in Siebel Tools. You cannot check out a project that is currently locked on the Siebel Server that someone else locked.
	Options	Opens the Development Tools Options dialog box with the Check In/Out tab chosen.
	Cancel	Cancels the checkout and closes the Check Out dialog box.

Elements of the Object Check Out Dialog Box

Table 41 describes the elements of the Object Check Out dialog box. This dialog box allows you to check out individual objects from the server database. For information about using this dialog box, see "Checking Out Objects from the Server Repository" on page 91.

Table 41. Elements of the Object Check Out Dialog Box

Element		Description
Repository Text Box		Displays the name of the current repository.
Object List	Type	Displays the type of new objects or checked out objects that the local repository contains. Objects that you download through a Get are not available for checkin and Siebel Tools does not list them. You can check in only projects that you previously checked out or that you created locally.
	Name	Displays the name of each object that Siebel Tools checks out.
	Updated	If the Siebel Server Locked By and Locked Date are different from the client version, then Siebel Tools displays a value of Yes. This value indicates that your version of the project is not synchronized with the version that the Siebel Server contains.
	Object Locking	If the project that this object references allows object checkin or checkout, then Siebel Tools displays a value of Yes. For more information, see "Allowing Object Locking for a Project" on page 89.
	Server Locked By	Displays the logon Id of the developer who currently checked out this object on the Siebel Server.
	Server Language	Displays the language you use when you check out the object on the Siebel Server. You can check out only one language at one time.
	Server Locked Date	Displays the date of checkout.
	Client Locked By	Displays the logon Id of the developer who currently locked this object locally.
	Client Language	Displays the language of the object that is currently locked in Siebel Tools. You can lock only one language at one time.
	Project Locked By	Displays the logon Id of the developer who most recently checked out the project that this object references.
Get Locale Specific Data Only		If this check box contains a check mark, then Siebel Tools gets only string translations and locale properties that Siebel CRM stores in the locale objects. It does not get data that Siebel CRM stores in the parent object of the locale object.

Table 41. Elements of the Object Check Out Dialog Box

Element		Description
Buttons	Get	Copies the objects that you choose to the local repository. It replaces any versions of these objects that already exist in the local repository but it does not lock these objects on the Siebel Server. You can Get any objects from the Siebel Server, including objects that someone else locked even if the Allow Object Locking field of the project that the object references contains a check mark. For more information, see "Allowing Object Locking for a Project" on page 89.
	Check Out	Copies all objects that you choose to the local repository and locks them on the Siebel Server and in Siebel Tools. You cannot check out an object that is currently locked on the Siebel Server that someone else locked.
	Options	Opens the Development Tools Options dialog box with the Check In/Out tab chosen.
	Cancel	Cancels the checkout and closes the Object Check Out dialog box.

Elements of the Check In Dialog Box

Table 42 describes the elements of the Check In dialog box. For information about using this dialog box, see "Checking In Projects or Objects to the Server Repository" on page 92.

Table 42. Elements of the Check In Dialog Box

Element	Description
Repository drop-down list	Lists the repositories in the local database. This list includes the same projects that the repository you choose contains. It also lists any projects that you create locally.

Table 42. Elements of the Check In Dialog Box

Element		Description
Projects list	Type	Displays the type of each new or checked out project or object that the local repository contains. Projects or objects that you download through a Get are not available for checkin and Siebel Tools does not list them. You can check in only projects or objects that you previously checked out or that you created locally.
	Name	Displays the name of the checked out object.
	Status	Contains the value New or Locked for each project or object. This value indicates if you created it or obtained it through a checkout.
	Lock/Creation Date	Displays the date and time for one of the following: ■ When you created the project or object ■ When you checked out the project or object from the Siebel Server
	Language	Displays the language you used to check out the project or object.
Option buttons	Selected Objects	If this check box includes a check mark, then you can manually choose an individual project or object to check in.
	Locked/New Objects	Chooses all of the projects or objects that the list contains. This list includes all the projects or objects you created or that you obtained through checkout.
Maintain Lock		Does not remove object locks on the Siebel Server or on the local database after checkin.
Buttons	Undo Check Out	Does not check in objects to the Siebel Server. This button releases the lock on the Siebel Server so that another developer can work on the objects. It does not unlock the locks on the local database.
	Validate	Validates the projects you choose.
	Check In	Starts the check-in process.
	Diff	Opens the Project Differences dialog box. This dialog box allows you to compare the objects that you check in to the versions of these objects that reside on the Siebel Server. For more information, see "Validating Objects" on page 109.
	Options	Opens the Developer Tools Options dialog box. This dialog box allows you to specify check-in and check-out settings. For more information, see "Specifying the Data Source That Siebel Tools Uses" on page 15.
	Cancel	Closes the Check In dialog box.

Dialog Boxes That Validate Objects

This topic describes dialog boxes that you use to validate objects. For more information, see "Validating Objects" on page 109.

This topic includes the following information:

- Elements of the Validate Dialog Box on page 237
- Elements of the Validation Options Dialog Box on page 238

Elements of the Validate Dialog Box

Table 43 describes the elements of the Validate dialog box.

Table 43. Elements of the Validate Dialog Box

Element	Description
Errors List	Displays the results of the validation process. Each row in the list identifies a rule violation for an object. You can do the following: ■ Double click the error in the Errors list to drill down on the object that causes the error. ■ Click a column heading to sort the rows. ■ Drag the right or left border of the heading cell to resize columns.
Severity Column	Includes an icon that indicates one of the following: ■ **Warning.** Yellow icon with an exclamation mark. ■ **Error.** Red icon with a minus sign. This error might cause a problem in Siebel CRM at run time.
Rule Column	Displays an integer that identifies the number of the violated rule. Siebel Tools lists rules sequentially according to the rule number.
Object Column	Displays the name of the object that failed validation.
Description Column	Displays a description of the error or warning.
Details Window	Displays more information about the error or warning message for the row that you choose in the Errors list.
Go To Button	To drill down on the object that causes an error, you can choose an error in the Error section, and then click Go To. You can also double-click the error message.
Log File Window	Displays the path and file name of the log file that includes the same list of validation errors and warnings that the Errors List shows. To save this list as a log file, you can click Save As, navigate to where you must save the file, and then specify a file name.

Table 43. Elements of the Validate Dialog Box

Element	Description
Load Button	Loads the contents of the log file that Siebel Tools saves when you use Save As in the Log File section. You can use the Load button to load the list of error and warning validations back into the Validation Tool at a later time.
Save As Button	Saves the list of validation rows that Siebel Tools currently shows in the Errors List as a log file.

Elements of the Validation Options Dialog Box

Table 44 describes the Rules area of the Validation Options dialog box. If you click Options in the Validate dialog box, then Siebel Tools displays the Validation Options Dialog box. To avoid validating objects that your configuration does not use, you must use the repository validator only in conjunction with the time filter.

Table 44. Elements of the Rules Area of the Validation Options Dialog Box

Element	Description
Rules List	Lists the rules that Siebel Tools enforces during validation. You can click a column heading to sort the rows. You can drag the right or left border of the heading cell to resize columns.
Severity Column	Displays an icon that indicates one of the following: ■ **Warning.** Yellow icon with an exclamation mark. ■ **Error.** Red icon with a minus sign. This error can cause a problem in Siebel CRM at run time.
Rule Column	Displays an integer that identifies the number of the violated rule. Siebel Tools lists rules sequentially according to the rule number.
Object Column	Displays the object type that Siebel Tools examines.
Description Column	Displays a description of the rule.
Enforce Column	Indicates if Siebel Tools enforces the rule. A Yes value validates all objects of the object type that the Object column identifies. If you use any of the following buttons in the Validation Options dialog box, then Siebel Tools modifies the value in the Enforce column: ■ Enforce ■ Ignore ■ Enforce All ■ Ignore All

Table 44. Elements of the Rules Area of the Validation Options Dialog Box

Element	Description
Save Button	Saves all the values for the current set of rules to a text file that you specify. If you press ENTER, and if the Validation Options dialog box is open, then Siebel Tools saves other settings to a preferences file.
Enforce Button	Modifies the value in the Enforce column in the row that you choose from No to Yes.
Ignore Button	Modifies the value in the Enforce column in the row that you choose from Yes to No.
Enforce All Button	Modifies all values in the Enforce column to Yes.
Ignore All Button	Modifies all values in the Enforce column to No. If you click Ignore All, then Siebel Tools does not validate any objects.
Details Text Box	Displays more information about the error or warning message for the row that you choose in the Errors list.
Last Validated Option	If this check box includes a check mark, then Siebel Tools validates only objects that Siebel CRM modified since the date you enter in the Last Validated window.
Custom Option	If this check box includes a check mark, then Siebel Tools validates only the objects that Siebel CRM modified according to the date range that you enter in the date and time fields that Siebel Tools shows next to the Custom option.
Do Not Report Warnings Option	If this check box includes a check mark, then Siebel Tools reports only errors. It does not report warnings. It also modifies the value in the Enforced column of all warning rules to No.
Abort Validation Option	If this check box includes a check mark, and if you enter a number in the window that Siebel Tools shows to the right of the option, then it stops validating after it reaches the number of errors you specify. If Siebel Tools identifies the number of errors that you specify in this window, then it stops validating the object and displays the Error dialog box.

Dialog Boxes That Compare Objects

This topic describes the dialog boxes that you use to compare objects.

Elements of the Compare Objects Dialog Box

Table 45 describes the elements of the Compare Objects dialog box.

Table 45. Elements of the Compare Objects Dialog Box

Element	Description
First Selection Section	Displays the object hierarchy as a tree. To expand a tree, you can use the controls in the First Selection window or the Second Selection window. For example, if you expand the tree in the First Selection window, then Siebel Tools does the following: ■ Expands the tree in the First Selection window. ■ Expands the tree in the Second Selection window. ■ Displays the child object types that each parent object contains. ■ Displays a dashed line to represent a child object that does not exist in an object.
Second Selection Section	
Properties Section	Displays the properties that are different for the objects that Siebel Tools compares.
Display Section	Determines how Siebel Tools displays items in the Compare Objects dialog box. You can choose the following options: ■ **Show All Objects.** Displays all child objects in the First Selection section and the Second Selection section. ■ **Show All User Properties.** Displays all user properties in the Properties section. ■ **Show System Properties.** Displays system properties in the Properties section. For example Created, Created By, Updated, and Updated By.
→	Synchronizes objects from the repository that the First Selection section represents to the repository that the Second Selection section represents. For more information, see "Comparing and Synchronizing Objects Between Repositories and Archives" on page 114.
←	Synchronizes objects from the repository that the Second Selection section represents to the repository that the First Selection section represents. For more information, see "Comparing and Synchronizing Objects Between Repositories and Archives" on page 114.
++	Expands the entire tree in the First Selection section and the Second Selection section.
−−	Collapses the entire tree in the First Selection section and the Second Selection section.
Delete Button	Deletes objects after a comparison.

Dialog Boxes That Set Development Options

This topic describes the dialog boxes that you use to set development options. It includes the following information:

- Development Options for Project Checkin and Project Checkout on page 241
- Development Options for Visualization Views on page 242
- Development Options for Scripting on page 242
- Development Options for Debugging on page 244

Development Options for Project Checkin and Project Checkout

Table 46 describes the development options that you can set for project checkin and project checkout. For information about using this dialog box, see “Specifying the Data Source That Siebel Tools Uses” on page 15 and “Configuring Third-Party Code Control” on page 96. For more information about projects, see Chapter 4, “Checking Out and Checking In Projects and Objects.”

Table 46. Development Options for Project Checkin and Project Checkout

Option	Description
Enable Source Control Integration	Allows Siebel Tools to create an archive for each project that you check in.
Show Execution of the Integration Batch File	Allows Siebel Tools to display a DOS window when the srcctrl.bat batch file runs. You can use this window for diagnostic purposes and to assist with debugging a custom batch file.
Integration Batch File	Specifies the location of the srcctrl.bat batch file that Siebel Tools uses. This file includes instructions that the code control software uses to do checkin or checkout.

Development Options for Visualization Views

Table 47 describes the development options that you can set for visualization views. For information about using this dialog box, see "Viewing Object Relationships" on page 113.

Table 47. Development Options for Visualization Views

Option	Description
Use System Font	Uses a system font for the visualization views.
Use a Custom Font	You can use the Font, Size, and Zoom drop-down lists to choose your preferred font.
Boxes with 3D borders	Displays boxes with a three dimensional border.
Icon and name only	Displays the object name and the same object icon that the Object Explorer shows.
Simple outline boxes	Displays each object name in a simple box.
Always print outline style	Displays visualization details in outline style.

Development Options for Scripting

Table 48 describes the development options that you can set for scripting. For more information, see "Setting Options for the Siebel Script Editor" on page 121.

Table 48. Development Options for Scripting

Section	Option	Description
Font	Name	Sets the font name that Siebel Tools shows for a script.
	Size	Sets the font size that Siebel Tools shows for a script.

Table 48. Development Options for Scripting

Section	Option	Description
Script Assist	Allows you to set Script Assist options. You must enable the ST eScript Engine so that you can use Script Assist. For more information, see "Overview of Using the ST eScript Engine" on page 123.	
	Enable Method Listing	Allows Script Assist to display a drop-down list that includes the methods and properties that are available for a declared object.
	Tab Width	Specifies the number of spaces that Siebel Tools uses for a tab character in the script. The default value is four spaces.
	Enable Auto Complete	If this check box contains a check mark, then Siebel Tools auto completes a method name or property name. It does this if you enter the minimal number of unique characters that are required to complete the name. If it locates strings that are not unique, then it displays a drop-down list.
	Auto Indent	If this check box contains a check mark, then Siebel Tools indents each succeeding line of script to the position that the current line sets.
	Enable Favorites	If this check box contains a check mark, then Siebel Tools displays the object, method, or property name that you use most frequently in italics at the top of the Script Assist window.
	Engine Settings	For more information, see "Setting Options for the ST eScript Engine" on page 127.
Language	Default Language for New Scripts	You can choose eScript or Visual Basic.

	Browser Script Compilation Folder	You can specify the folder where Siebel Tools stores the Browser script that it compiles. For example:
		
		`C:\Program Files\Siebel\8.0\web client\PUBLIC\enu`
		
		In this example, browser script files reside in the following folder:
		
		`C:\Program Files\Siebel\8.0\web client\PUBLIC\enu\`*`genbscript_time stamped`*`_folder\bscripts\all.`

Table 48. Development Options for Scripting

Section	Option	Description
Debugging	Allows you to set options for the Siebel Debugger. For more information, see "Setting Debug Options to Open the Siebel Client" on page 187.	
	Adjust Breakpoint to Next Valid Line	If this check box contains a check mark, and if you delete a breakpoint on an invalid code line, then Siebel Tools creates a breakpoint at the next valid line.
	Make Debugger Window Active When Debugging	If this check box contains a check mark, then Siebel Tools displays the Siebel Debugger window when it is in debug mode.
	Always Enter the Debugger When an Error Occurs	If this check box contains a check mark, and if a script error occurs, then Siebel Tools displays the Siebel Debugger window.

Development Options for Debugging

Table 49 describes the development options that you can set for debugging. To access this dialog box in Siebel Tools, you click the View menu, click Options, and then click the Debug tab. Siebel Tools stores the settings you make on the Debug tab in the following user preference file:

```
loginId&SiebelTools.spf
```

It stores this file in the *SIEBEL_TOOLS_ROOT*\BIN folder.

Table 49. Development Options for Debugging

Option	Description
Executable	Enter the name of the Siebel Web Client executable. For example: `siebel.exe` The default value is siebel.exe. Siebel Tools runs this executable in debug mode or automatically after compile finishes.
CFG File	Enter the name of the configuration file that the Siebel client uses. For example: `C:\Program Files\Siebel\8.0\web client\BIN\ENU\uagent.cfg` For more information, see "Specifying the Data Source That Siebel Tools Uses" on page 15.
Browser	Enter the path to the browser executable. For example: `C:\Program Files\Internet Explorer\iexplore.exe`

Table 49. Development Options for Debugging

Option	Description
Working Directory	Enter the Siebel root folder. This folder includes the Siebel executable and the DLLs (dynamic link libraries). For example: `C:\Program Files\Siebel\8.0\web client\BIN`
Arguments	Enter the options that Siebel Tools uses when it opens the watch window: ■ ***/h.*** Enables local debugging of Server scripts. ■ ***/s file name.*** Enable SQL spooling.
Prompt for This Information Each Time	If this check box contains a check mark, then Siebel Tools displays information each time it runs a debug operation. For example, it can display the name of the executable, the name of the CFG file, the browser configuration, and so on.
Show Workflow Primary Business Component Data	If this check box contains a check mark, then the Watch window in the Workflow Simulator displays information about the workflow process. It displays the name of each business component field and the value for each of these fields. These fields include fields from the primary business component of the business object that the workflow process references.
User Name	Enter the user name that Siebel CRM requires to log in to the Siebel application you are debugging.
Password	Enter the password that Siebel CRM requires to log in to the Siebel application you are debugging.
Data Source	Choose a default data source. The values you can choose depend on Oracle's Siebel Tools configuration file that you specify in the CFG File option. The Siebel Web Client connects to this local database. For more information, see "Specifying the Data Source That Siebel Tools Uses" on page 15.
Enable Profiler	For more information, see "Using the Script Profiler" on page 138.

Parameters That Convert Symbolic Strings

This topic describes the parameters that you can use to convert a symbolic string. You use these parameters with a conversion utility. For more information, see "Converting Symbolic Strings" on page 153.

This topic includes the following information:

- Parameters You Use with the Conversion Export Utility on page 246
- Parameters That You Can Use with the Conversion Import Utility on page 247

Parameters You Use with the Conversion Export Utility

Table 50 describes the parameters that you can use with the conversion export utility.

Table 50. Parameters You Use with the Conversion Export Utility

Parameter	Description
Filename	Required. Specifies the name of the export file.
Repository	Required. Specifies the name of the repository. The repository name is case sensitive.
`Object`	Required. Specifies the object type that contains the strings that this utility exports. For example: `Control` The object name is case sensitive.
LogFile	Specifies the name of the log file.
Language	Specifies the language that this utility uses as the primary language to match when it searches for duplicate symbolic strings. For example, assume each symbolic string includes the following child records: ■ English (ENU) ■ French (FRA) ■ German (DEU) If you set the Language parameter to ENU, then the conversion export searches for matches between the ENU records. If it finds matches, then it examines the other child records of the other languages. If all child records match, or if one language includes a superset of one of the other languages, then this utility considers them as matching symbolic strings.
MatchMin	Specifies the minimum number of matches in a set of matching symbolic strings before this utility writes them to the file. The default value is 2.
SQLLog	Specifies the SQL log file name. If you set this parameter, then this utility logs all SQL that it runs to this file.
ExcludeNull	Specifies a TRUE or FALSE value. If TRUE, then this utility excludes null values for conversion consideration. The default value is TRUE.
UseFullMatch	Specifies a TRUE or FALSE value. If TRUE, then this utility matches records against all the other possible match candidates before it discards them. The default value is TRUE.

Table 50. Parameters You Use with the Conversion Export Utility

Parameter	Description
UseExactMatch	Specifies a TRUE or FALSE value. If TRUE, then this utility considers records as a match only if all of the records include the same number of language records, and if the same values exist for each language. It does not consider partial matches. The default value is FALSE.
SkipInactive	Specifies a TRUE or FALSE value. If TRUE, and if the Inactive property of the record is set to Y, then this utility skips this record. The default value is TRUE.

Parameters That You Can Use with the Conversion Import Utility

Table 51 describes the parameters that you use with the conversion import utility.

Table 51. Parameters You Use with the Conversion Import Utility

Parameter	Description
Filename	Required. Specifies the name of the import file. You must use the same name that you use when you export symbolic strings.
Repository	Required. Specifies the name of the repository.
LogFile	Specifies the log file.
UnlockProjects	Specifies a TRUE or FALSE value. If TRUE, then this utility unlocks all projects when the conversion finishes. This configuration is useful if multiple instances of the conversion service run against the same database. The default value is TRUE.
SkipParentUpdates	Specifies a TRUE or FALSE value. If TRUE, then this utility does not update parent objects to use the symbolic string. The default value is FALSE. Set SkipParentUpdates to TRUE only if you do not simultaneously run multiple instances of the import. If you set SkipParentUpdates to TRUE, and if you run multiple instances, then errors might occur. The utility might abort an update or delete records because another instance updates the project at the same time.
SQLLog	Specifies the log file name. If you use this parameter, then this utility logs all SQL that it runs to the file you specify.
Project	Required. Specifies the name of the project in the repository that includes the new strings. Predefined symbolic strings reside in the Symbolic Strings project. You can configure this utility to import custom strings. For more information, see "About Predefined Objects" on page 24.

Table 51. Parameters You Use with the Conversion Import Utility

Parameter	Description
DeleteLocales	Specifies a TRUE or FALSE value. If TRUE, and if all translatable strings are NULL, and if language override is not enabled, then this utility deletes locale records. If FALSE, then this utility sets the locale record to Inactive. The default value is TRUE. For more information, see "Enabling Language Override" on page 20.
CheckTranslateFlag	Specifies a TRUE or FALSE value. If TRUE, and if the Translate property for the object is N, then this utility does not convert this object. The default value is TRUE.
LogErrorRecords	Specifies a TRUE or FALSE value. If TRUE, then this utility exports all error records to a separate log file. The default value is FALSE.

Glossary

Advanced Compile

A feature in Siebel Tools that you can use to assist with localization.

Applets window

A window that displays information about a view and allows you to add applets to that view.

Application Deployment Manager (ADM)

A feature that you can use to administer a Siebel CRM deployment.

bookmark

A feature that allows you to return directly to an item.

Bookmarks window

A window that allows you to navigate directly to an object that you use frequently.

breakpoint

A marker in a line of code that stops code from running at that line.

canvas

A background that Siebel Tools shows in different designers.

collection function

A type of function that includes a finite set of values.

compound query

A type of query that locates records according to more than one condition.

conflict

A situation that occurs if an object that Siebel Tools attempts to import from an archive does not match the corresponding object in the repository.

Controls/Columns window

A window that displays the controls and list columns that you can add to an applet layout if you use the Applet Layout Editor.

custom object

A new object that you create.

declarative configuration

A type of programming technique that uses objects and object properties in the Siebel repository to implement the logic that your business requires.

developer conflict

An error that occurs if two separate groups or developers update the same object.

full get

A type of Get that copies all projects from the server repository to your local repository.

get

The act of copying a project from the server repository to your local repository.

hidden object type

An object type that Siebel CRM does not show in the Siebel Web Client.

hotfix

A Siebel Tools feature you can use to quickly update the production environment.

Incremental Repository Merge

A feature that allows you to merge multiple repositories and apply patches.

language mode

A mode that allows you to configure Siebel CRM to display text in a language other than English.

language override

A nontranslatable locale property that you can configure differently for different locales.

Locale Management Utility (LMU)

A utility in Siebel Tools that you can use to manage how you configure Siebel CRM to localize text strings, such as field labels, and other locale properties, such as the height and width of controls.

mid-level release

A type of release that includes objects you modify from a time frame that you specify.

modified object

A predefined or custom object that you modify.

new object wizard

A feature you can use that guides you through the steps of creating a new object.

object definition

Implements one piece of the software. It consists of object properties, which are characteristics of this piece of software.

Object Explorer

A window in Siebel Tools that displays the Siebel object hierarchy.

Object List Editor

A window in Siebel Tools that displays the object definitions of the object type that you choose in the Object Explorer.

object property
A characteristic of a piece of software.

object tagging
A version control feature that Siebel Tools uses to associate a repository modification with a tag or group. You can use it to export all the work that a group of developers performs.

object type
An entity that includes a predefined set of properties.

Palettes window
A window that allows you to add items to an object.

parent and child relationship
A type of hierarchical relationship between one object type and another object type.

predefined object
An object that comes already defined when you first install Siebel CRM or Siebel Tools.

predefined symbolic string
A string that comes predefined with Siebel CRM.

prior standard repository
The repository that comes predefined with Siebel CRM. Siebel Tools uses it to determine if you modified any object in the repository since the prior release.

project
An object type that your development team can use to help make sure only one developer works on an object at one time.

property value
Information that you enter into the column of an object definition.

Properties window
A window that displays the properties and the property value for the object that you choose in the Object List Editor.

pseudolocalization prefix
A prefix that Siebel Tools uses to test the appearance of string in a language.

reference repository
A prior version of the repository.

right-click menu
A type of context-sensitive menu that Siebel Tools shows if you right-click an object or an element.

Running Tool Tip
A help feature in the Siebel Script Editor that you can use to write a script.

script library
A part of the ST eScript Engine that allows you to call a business service method from a script.

Siebel Application Upgrader
A utility you can use to get new features from the latest software release while preserving the custom configuration you created in the current repository.

Siebel Delta File (SDF)
A type of file that is similar to an SIF file except that an SDF file contains only information about modifications to an object.

Siebel Repository
A set of tables that includes Siebel objects and server scripts.

Siebel repository patch file (SPF)
A type of file that includes exported objects.

Siebel Script Editor
An editor that allows you to create and maintain scripts that use Siebel VB, Siebel eScript, or Browser Script.

Siebel Script Performance Profiler
A part of the ST eScript Engine that allows you to observe and monitor the performance of a script.

Siebel Tools
An integrated development environment that you can use to configure Siebel CRM.

simple query
A type of query that locates records according to one condition.

symbolic string
An object that you can use to store the value of a string.

symbolic string reference
A reference to a symbolic string.

top-level object type
An object type that resides at the top of the object hierarchy.

Touch
A version control feature in Siebel Tools that allows you to tag an object even if you do not modify this object.

translatable string
A type of string that Siebel CRM can translate to another language.

Type deduction
A feature of the ST eScript Engine that determines the type of local variables that a script uses.

Validate Tool

An error correction tool you can use to validate the semantic consistency of an object.

Web template editor

An application external to Siebel Tools that allows you to edit a Web template.

Web Template Explorer window

A window that displays the HTML code of a Siebel Web Template (.swt) file.

Index

A

Advanced Compile
- overview of using 175
- using 176

Allow Object Locking property 89
Applet Layout Editor, using 42
Applet Menu Layout Editor, using 42
applets
- controls and columns for editing 36
- using window for 35

Application Deployment Manager
- about using 204
- exporting objects to a mid-level release 207
- using a hotfix to deploy fixes 205
- using the command line to export objects to a hotfix 205

Archive versus Archive option 116
archives
- comparing objects in two different 116
- exporting objects to 190
- exporting only modified objects to 191
- importing 196
- importing multiple 198
- importing objects from 195
- importing only modified objects from 200
- preparing Siebel Tools environment to import objects from 195
- using command line to export objects to 190
- using command-line interface to import objects from 199
- using to export and import objects 189

B

batch files
- using to consolidate strings 160
- using to convert strings 160

bookmarks
- and Go menu 215
- and History toolbar 224

Bookmarks window, using 38
breakpoints, using 135
Browser Script Editor, about using 45
browser, choosing target 17

C

Calls window, using 135
Changed property, using 29
Check In dialog box, elements of 235
Check Out dialog box, elements of 231
checkout
- restarting editors after 101
- undoing for projects 94

checkout and checkin
- development options for 241
- guidelines for 86
- overview of 85
- preventing for objects 95
- setting options for 96
- using 89

command-line interface
- about using 45
- using to export objects to a hotfix 205
- using to export objects to archives 190
- using to export strings that require translation 174
- using to import a file 174
- using to import objects from archives 199
- using to import, export, and compile objects 183
- using to run Locale Management Utility 172

Compare Objects dialog box
- elements of 239
- using to compare object in current repository and archive file 115
- using to compare objects in archive files 116
- using to compare objects in different repositories 115
- using to compare objects in the same repository 114
- using to synchronize objects between repositories 116

compound queries, about creating 33
Configuration Context toolbar, about using 228
configuration file, modifying to support symbolic strings 147
Confirmation dialog boxes, controlling display of 20
conflict resolution
- displaying hierarchy of differences for objects 203
- displaying object definition to a row 203
- displaying property value conflicts for object

definitions 204
consoleapp.exe
running 156
using to import consolidated strings 159
using to run string conversion utility 153, 158
using to split export files into smaller files 157
Controls/Columns window, using 36
Conversion Export utility
and string conversion 157
parameters for 246
Conversion Import utility
and string conversion 157
parameters for 247

D
data source, specifying for Siebel Tools 15
database
setting options for 17
updating remote 78
Database Configuration Wizard, using to migrate repositories 75
Debug menu, options in 216
debug options
for development 244
setting 133
setting to open Siebel Web Client 187
Debug toolbar, buttons on 225
design environments, canvas-based 43
Detail tab, using 27
development environment
process of setting up 14
resetting options in 20
development server, recovering from failed 188
dialog boxes
that check out and check in projects and objects 230
that compare objects 239
that set development options 241
that validate objects 237

E
Edit menu, options in 211
Edit toolbar, buttons on 222
edit tools
displaying from Edit menu 211
displaying from Edit toolbar 222
editors
restarting after checkout 101
using 41, 43
environment settings, applying using View menu 212

F
File menu, options in 210
Fix and Go, using 128
Flat tab, using 28
Format menu, options in 216
Format toolbar, buttons on 226
function_name Is An Unknown Function error message 138

G
Go menu, options in 215

H
Help menu, options in 221
hidden windows, navigating to 220
History toolbar, buttons on 224
hotfix
using the command line to export objects to 205
using to deploy fixes 205
HTML source code, displaying for templates 37

I
identification numbers, adding to repository modifications 65
Import Wizard-Review Conflicts and Actions
about 202
Attribute Differences pane in 204
Conflicting Objects pane in 203
Object Difference pane in 203
Inactive property, using 30
incremental repository upgrade kits, using 183

L
languages
enabling overrides of 20
setting modes for 19
List toolbar, buttons on 223
local repository
locking objects in 93
locking projects in 93
viewing differences with server repository 95
Locale Management Utility
exporting strings and locale properties using 170
finding existing translations using 167
finding untranslated text strings using 166
importing strings and locale properties

using 171
marking as Redo modified records for project 169
modifying strings using 172
using 165
using commend line to run 172
using to find modified objects since export 169
locale objects, finding untranslated text strings for 166
locale properties
configuring non-translatable 161
exporting 170
importing 171
using command line to export 173
locales
configuring data for 161
displaying controls and list columns according to 162
locked objects, viewing in projects 95

M
menu bar
Debug menu in 216
Edit menu in 211
File menu in 210
Format menu in 216
Go menu in 215
Help menu in 221
menus on 209
Query menu in 215
Screen menu in 214
Tools menu in 217
using to display menus 31
View menu in 212
Window menu in 220
mid-level release, exporting objects to 207
modifications
compiling 179
overview of compiling 179
testing and troubleshooting 187
modified objects, finding 168
Multi Value Property Window, using 37

N
New Object wizard
about 222
using to create objects 107
non-repository configurations and data, migrating 78
non-translatable locale properties, configuring 161

O
Object Check Out dialog box, elements of 234
object comparison
about 114
in archive files 116
in current repository and archive file 115
in different repositories 115
in the same repository 114
using Compare Objects dialog box in 239
Object Compiler dialog box, accessing 180
object definitions
comparing using Compare Objects dialog box 239
display of 21
locating and modifying in Object List Editor 24
validating using Validate dialog box 237
validating using Validation Options dialog box 238
Object Explorer
overview of 21
using 21
using lists and tabs in 27
object hierarchies, display of 23
Object List Editor
controlling display of 29
displaying object types in 26
locating and modifying object definitions in 24
modifying multiple records in 25
overview of 21
using 21, 28
using the link in 30
Object List Editor window
List toolbar for 223
refining queries in 215
showing Visualization views in 113
using Edit menu to apply object definitions in 211
using Edit menu to apply objects in 211
using to modify objects 108
object locking, allowing for projects 89
object tagging
administering 56
and modifications for object definitions 55
and using Siebel Remote 62
enabling 57
for unmodified objects 59
procedure for 58
using to manage developer modifications 56
object types
display of 21

displaying hidden 26
displaying in Object List Editor 26
object wizards, using to create objects 39
objects
about comparison and synchronization for 114
associating with different projects 103
checking in to server repository 92
checking out from server repository 91
comparing in archive files 116
comparing in current repository and archive file 115
comparing in different repositories 115
comparing in the same repository 114
compiling single or group of 182
copying 108
creating 107
deleting 110
examining 111
exporting to a mid-level release 207
finding modified 168
finding modified since export 169
getting documentation about 34
locking in local repository 93
modifying 108
modifying the saved location of 103
overriding restrictions preventing modification of 65
overview of customizing 105
preventing checkin and checkout of 95
restricting those developers can modify 62
synchronizing between repositories 116
using archive files to export and import 189
using command line to import, export, and compile 183
using object wizards to create 39
using the command line to export to a hotfix 205
validating 109
viewing relationships for 113

P

Palettes window
buttons in 229
using 37
Project drop-down list, using 27
projects
allowing object locking for 89
associating objects with different 103
checking in to server repository 92
checking out from server repository 90
compiling 180
creating for symbolic strings 147
creating in server repository 102
creating new 101
customizing 101
defined and about 85
getting all from server repository 87
getting from server repository 86
getting some from server repository 88
locking in local repository 93
meaning of suffix names for 85
modifying the saved location of 103
renaming 102
undoing checkout of 94
unlocking 94
viewing locked objects in 95
Properties window, using to display property settings 35
properties, displaying hidden 26

Q

queries
about compound 33
about simple 32
running 31
Query menu, options in 215

R

records, determining when Siebel CRM created or updated 118
Redo, Locale Management Utility marking projects as 169
remote databases, updating 78
repimexp utility, using to import and export repositories 80
repositories
comparing two files for 52
deleting 83
exporting 70
exporting and importing 66
files you can use to manage 48
guidelines for migrating 82
identifying conflicts that occur during upgrades of 72
importing in UNIX environments 69
importing in Windows environments 68
importing or exporting later 71
managing 47
overview of exporting and importing 67
overview of managing 47
preparing to migrate 73
process of migrating 73
renaming 50
searching 111
upgrading 71

using Advanced Compile to localize 175
using Database Configuration Wizard to migrate 75
viewing information about current 49
repository modifications
adding identification numbers to 65
associating with developers 55
managing 54
results.slf, about 170
results.txt, about 170
right-click menus
about using 31
using to hide windows 38
Running Tool Tip
how it differs from Tool Tips in Script Assist 130
using 129

S

Screen menu, options in 214
Script Assist
about using 125
accessing the window for 128
using script libraries for 130
script libraries, using with ST eScript Engine 130
scripts
development options for 242
tracing 136
SDF files, about 49
server repository
checking in projects or objects to 92
checking out objects from 91
checking out projects from 90
creating projects in 102
getting all projects from 87
getting locale data from 89
getting projects from 86
getting some projects from 88
viewing differences with local repository 95
Server Script Editor, about using 45
Server scripting language, setting default 121
ServerEditRecordTimeStamp parameter, setting 63
Siebel Application Upgrader, about 71
Siebel Compiler, order considerations and error message in 138
Siebel Debugger
accessing 134
overview of using 132
using 132
using to display script variables and values 136
using to display subroutines and function calls 135
Siebel eScript
menu options of debugger for 216
using Debug toolbar to access debugger for 225
Siebel Script Editor
about 45
opening 122
overview of using 119
setting options for 121
using 119
Siebel Script Performance Profiler
about window for 139
enabling 141
overview of using 139
running 142
using 138
Siebel Tools
installing 14
overview of using 13
roadmap for setting up and using 14
specifying data source for 15
Siebel VB
menu options of debugger for 216
using Debug toolbar to access debugger for 225
Siebel Web Client, automatically opening 187
SIF file, about 49
simple queries, about 32
Simulate toolbar, buttons on 226
SRFDiff utility, about 52
ST eScript Engine
enabling 127
overview of using 123
setting options for 127
using 123
using Fix and Go with 128
status bar, showing and hiding 41
string consolidation utility
running 158
using to import consolidated strings 159
string conversion utility
running 153
using to split export files into smaller files 157
string overrides, entering 152
String Reference pick applet, using to set symbolic string reference 150
string references, fixing after upgrade 164
strings
exporting 170

finding existing translations for 167
finding modified objects since export 169
finding multiple properties for 166
finding untranslated 166
importing 171
modifying 172
using batch files to consolidate 160
using batch files to convert 160
using command line to export 173
symbolic string reference, using String Reference pick applet to set 150
symbolic strings
configuring 145
consolidating 158
converting 153
creating 148
creating a project for 147
how Siebel CRM references at run time 158
modifying predefined 148
modifying the configuration file to support 147
modifying to globally update display values 149
overview of configuring 145
parameters that convert 245
syntax, validating for scripts 134

T
target browser, choosing 17
tasks
accessing WF/Task Editor toolbar for 228
comparing different versions of task UIs 117
controlling versions of task UIs 18
third-party source control, configuring 96
toolbars
about using Configuration Context 228
buttons on 222
buttons on Debug 225
buttons on Edit 222
buttons on Format 226
buttons on History 224
buttons on List 223
buttons on Simulate 226
buttons on WF/Task Editor 228
showing and hiding 41
using 40
Tools environment configuration
configuring third-party source control for 96
restarting editors after checkout in 101
setting debug options in 187
setting options for Siebel Script Editor in 121
Tools menu, options in 217
translations
exporting strings for 174
finding existing 167
Types tab, using 27

U
untranslated text strings
finding 166
finding existing translations for 167

V
Validate dialog box, elements of 237
Validate Options dialog box, elements of 238
View Layout Editor, using 42
View menu
options in 212
showing Visualization views in 113
Visualization views
development options for 242
showing 113

W
W property, using 29
Watch window, using to display script variables and values 136
Web browser, defining layout 228
Web Page Layout Editor, using 42
Web template editor, specifying 42
Web Template Explorer window, using 37
Web templates, displaying list of 37
WF/Task Editor toolbar, buttons on 228
Window menu, options in 220
windows
docking and undocking 38
hiding docked as tabs 38
resetting 39
showing and hiding 38
stacking dockable 39
workflows
accessing WF/Task Editor toolbar for 228
comparing different versions of 117
controlling versions of 18