Siebel CRM Desktop for Microsoft Outlook Administration Guide

Version 8.0, Rev A

June 2011

ORACLE

Contents

Chapter 1: What's New in This Release

Chapter 2: Overview of Siebel CRM Desktop for Microsoft Outlook

Benefits of Using Siebel CRM Desktop 15

Scenarios for Using Siebel CRM Desktop 15

- Scenario for Working with an Activity That Is Associated with an Opportunity 16
- Scenario for Managing Contact Information 16
- Scenario for Managing Account Information 17
- Scenario for Associating an Email with an Opportunity 17
- Scenario for Managing an Opportunity 18

Chapter 3: How Siebel CRM Desktop Works

Overview of How Siebel CRM Desktop Works 19

- Extensions to the Microsoft Outlook User Interface 19
- Infrastructure That Siebel CRM Desktop Uses 20
- Architecture Components That Siebel CRM Desktop Uses 21

How Siebel CRM Desktop Uses the Siebel Enterprise 24

- Siebel Enterprise Components That Siebel CRM Desktop Uses 25
- About the Web Service API 26
- About the PIM Client Sync Service Business Service 27
- About the EAI Siebel Adapter Business Service 27
- About Integration Objects 28
- User Details Business Component 29
- About Authentication and Session Management 30

Metadata That Describes the Siebel CRM Desktop Application 30

- Relationships Between Users, Responsibilities, Customization Packages, and Metadata Files 30
- About the Customization Package 32
- About Metadata Files 33

Chapter 4: How Siebel CRM Desktop Handles Siebel CRM Data

How Siebel CRM Desktop Handles an Activity 35
- Overview of How Siebel CRM Desktop Handles an Activity 35
- How an Activity Is Created or Modified 37
- How Siebel CRM Desktop Processes an Activity 37
- How Siebel CRM Desktop Resolves Participants and Email Recipients of an Activity 39
- How Siebel CRM Desktop Displays an Activity in Microsoft Outlook 41
- How Siebel CRM Desktop Sets the Primary Employee of an Activity 41
- How Siebel CRM Desktop Handles an Attachment 44

How Siebel CRM Desktop Handles a Shared Activity 44
- How the Origin of an Activity Affects Handling 44
- Example of How Siebel CRM Desktop Handles a Microsoft Outlook Meeting That Contains Multiple Attendees 46
- Example of How Siebel CRM Desktop Handles a Shared Microsoft Outlook Appointment That Is Declined 47

How Siebel CRM Desktop Handles Microsoft Outlook Calendar 47
- How Siebel CRM Desktop Handles a Microsoft Outlook Calendar Item That the User Saves, Changes or Deletes 48
- How Siebel CRM Desktop Handles a Siebel CRM Activity That the User Saves, Changes, or Deletes 48
- How Siebel CRM Desktop Handles an Appointment 49
- How Siebel CRM Desktop Handles a Recurring Appointment 55

How Siebel CRM Desktop Handles a Microsoft Outlook Task 56

How Siebel CRM Desktop Handles a Microsoft Outlook Email Message 56

How Siebel CRM Desktop Handles Items If the User Removes the Siebel CRM Desktop Add-In 57

How a User Can Link a Siebel CRM Record to a Microsoft Outlook Record 58

Chapter 5: How Siebel CRM Desktop Synchronizes Data

How Siebel CRM Desktop Synchronizes Data Between the Client and the Siebel Server 59
- How Siebel CRM Desktop Synchronizes Data During the Initial Synchronization 59
- How Siebel CRM Desktop Synchronizes Data During an Incremental Synchronization 60
- How Siebel CRM Desktop Synchronizes Siebel CRM Data 62
- How Siebel CRM Desktop Manages Synchronization Duration 63
- Situations Where Siebel CRM Desktop Reinstalls the Data Structure 64

Factors That Determine Which Data Siebel CRM Desktop Synchronizes 66
- Factors That Determine Which Data a Siebel CRM Desktop User Can Access 66

How Differences in Data Between Microsoft Outlook and the Siebel Server Affect Synchronization 69
How You or the User Can Modify Synchronization Frequency 70
How Siebel CRM Desktop Avoids Duplicate Data 70
How Siebel CRM Desktop Handles a Conflict During Synchronization 71
How Siebel CRM Desktop Prevents Cyclical Synchronization 71
How Siebel CRM Desktop Prevents Duplicate Records 73
How Siebel CRM Desktop Handles a Synchronization Error 74

Chapter 6: Installing Siebel CRM Desktop

Roadmap for Installing Siebel CRM Desktop 77
Process of Preparing the Siebel Server 77
Preparing the Implementation Environment for Siebel CRM Desktop 77
Administering Metadata Files 78
Creating and Publishing the Customization Package 82
Administering Server Variables 84
Overview of Installing the Siebel CRM Desktop Add-In 85
About Files, File Locations, and Profiles 86
Changes That Siebel CRM Desktop Makes During Installation 87
Process of Installing the Siebel CRM Desktop Add-In 89
Verifying the Network and Infrastructure 89
Installing the Siebel CRM Desktop Add-In for a Single User 91
Options for Installing the Siebel CRM Desktop Add-In 92
Customizing the First Run Assistant 93
Guidelines for Installing the Siebel CRM Desktop Add-In for Multiple Users 99
Using the DOS Command Line to Set Optional Parameters 101

Chapter 7: Administering Siebel CRM Desktop

Changing the Behavior of Siebel CRM Desktop 105
Modifying the Windows Registry to Change Siebel CRM Desktop Behavior 106
Overriding Windows Registry Parameters That Locate the Siebel Server 110
Setting Behavioral Limits for Siebel CRM Desktop 111
Defining the URL That Siebel CRM Desktop Uses to Open the Siebel Web Client 113
Controlling the Object Types That Siebel CRM Desktop Displays in the Filter Records Tab 114
Controlling the Synchronization Intervals That Siebel CRM Desktop Displays in the Synchronization Tab 114
Controlling the Fields That Are Available in a Filter 115
Controlling How Siebel CRM Desktop Converts Contacts 116
Controlling How Siebel CRM Desktop Deletes Records During Synchronization 116

Configuring the Exclusions List 122
Controlling How Siebel CRM Desktop Handles an Archived Item 123
Removing and Upgrading the Siebel CRM Desktop Add-In 124
Removing the Siebel CRM Desktop Add-In for a Single User 125
Removing the Siebel CRM Desktop Add-In for Multiple Users 126
Upgrading the Siebel CRM Desktop Siebel CRM Desktop Add-In 126
Administering Metadata 127
Overview of Administering Metadata 127
Republishing a Customization Package 127
Administering Features That Affect Performance 128
Controlling the Time and Day to Perform Synchronizations 128
Regulating the Size and Type of Synchronized Records 130
Regulating the Number of Records That Siebel CRM Desktop Synchronizes 131
Administering Batching Options 132
Administering Tracing and Logging 132
Administering an Appointment That a Non-Siebel User Creates 133
Troubleshooting Problems That Occur with Siebel CRM Desktop 134
Enabling SOAP Data Dumps 134
Troubleshooting Problems That Occur When Siebel CRM Desktop Connects to the Siebel Server 136
Troubleshooting Problems That Occur During Synchronization 137
Troubleshooting Problems That Occur When You Customize Siebel CRM Desktop 139

Chapter 8: Customizing Siebel CRM Desktop

Customizing the Siebel CRM Desktop Application 143
Overview of Customizing Siebel CRM Desktop 144
About Files in the Customization Package 144
Customizing How Siebel CRM Desktop Maps Fields 148
Customizing Which CRM Data Siebel CRM Desktop Removes if the User Removes Siebel CRM Desktop 149
Customizing Synchronization 149
Customizing a Form 150
Customizing a Toolbar 151
Customizing a Dialog Box 151
Customizing a View 152
Customizing the SalesBook Control 152
Customizing Meta Information 152
Customizing Localization 153
Customizing the URL Protocol to Use HTTPS 154
Customizing the Email Address of the Support Team 154

Performing Typical Customization Work 155
Checking Out a Project in Siebel Tools 155
Displaying Object Types in Siebel Tools 156
Customizing the Product Name 156
Localizing Values 157
Displaying a Custom Siebel CRM Field in a Siebel CRM Desktop Form 159
Making a Field in Siebel CRM Desktop Read-Only 163
Adding a Default Value to a Field in Siebel CRM Desktop 164
Preventing the User From Deleting Records 165
Process of Customizing Objects in Siebel CRM Desktop 166
Defining the Custom Object 166
Defining Synchronization for a Custom Object 169
Adding Custom Views in Microsoft Outlook 171
Defining the User Interface 171
Defining the Validation Rules 179
Defining Validation Rules for a Phone Number 180
Adding Custom Logic 182
Defining the Toolbar 183
Defining Other Options 185
Defining Logic for the Custom Form 185
Process of Adding a Predefined Siebel Picklist to Siebel CRM Desktop 185
Identifying Predefined Picklist Objects in Siebel CRM 186
Creating an Integration Object for the Contact Method Picklist 188
Extending an Integration Object for the Contact Method Picklist 189
Adding a Field to the Customization Package 190
Customizing the Physical Layout for the Pick List 198
Publishing and Testing a Custom Picklist 202
Process of Adding an MVG Field 203
Identifying Predefined MVG Objects in Siebel CRM 204
Process of Making Siebel CRM Data Available to Add an MVG 206
Process of Modifying the Customization Package to Add an MVG 211
Publishing and Testing a Custom MVG Field 216
Example Code You Use to Add an MVG 217

Appendix A: How Siebel CRM Desktop Maps Fields Between Siebel CRM Data and Microsoft Outlook Data

How Siebel CRM Desktop Maps Fields Between Siebel Activities and Outlook Calendar 229

How Siebel CRM Desktop Maps Fields Between Siebel Activities and Outlook Tasks 232

How Siebel CRM Desktop Maps the Owner Field Between a Siebel CRM Activity and an Outlook Task 234
How Siebel CRM Desktop Maps the Status Field of an Activity 235
How Siebel CRM Desktop Maps Fields Between a Siebel CRM Activity and an Outlook Email 236
How Siebel CRM Desktop Transforms Objects Between Siebel CRM Data and Microsoft Outlook Data 238
How Siebel CRM Desktop Transforms a Calendar Event That Does Not Recur 238
How Siebel CRM Desktop Transforms a Recurring Event That Matches a Siebel Recurrence Pattern 239
How Siebel CRM Desktop Transforms a Recurrent Event That Does Not Match a Siebel Recurrence Pattern 241
How Siebel CRM Desktop Transforms a Siebel CRM Activity That Is Not Repeated 242
How Siebel CRM Desktop Transforms a Siebel CRM Activity That Is Repeated 242
How Siebel CRM Desktop Maps Fields Between Certain Siebel Appointments and Microsoft Outlook Appointments 244

Appendix B: XML Files Reference

XML Code to Map a Field 245
Example Code of the siebel_basic_mapping.xml File 246
Type Tag of the siebel_basic_mapping.xml File 247
Form Tag of the siebel_basic_mapping.xml File 247
Alt Message Classes Tag of the siebel_basic_mapping.xml File 248
Custom Views Tag of the siebel_basic_mapping.xml File 248
Field Tag of the siebel_basic_mapping.xml File 249
Writer Tag of the siebel_basic_mapping.xml File 249
XML Code to Customize the Data That Siebel CRM Desktop Deletes if You Remove Siebel CRM Desktop 250
XML Code to Customize Synchronization 251
Example Code for the connector_configuration.xml File 252
Types Tag of the connector_configuration.xml File 253
Type Tag of the connector_configuration.xml File 253
View Tag of the connector_configuration.xml File 253
Synchronizer Tag of the connector_configuration.xml File 254
Links Tag of the connector_configuration.xml File 254
Natural Key Tag of the connector_configuration.xml File 255
Filter Presets Tag of the connector_configuration.xml File 255
XML Code to Customize Forms 258
Form Tag of the forms_xx.xml File 259
Validation Rules Tag of the forms_xx.xml File 261
Script Tag of the forms_xx.xml File 262

Info Bar Tag of the forms_xx.xml File 263
Page Tag of the forms_xx.xml File 263
Stack Tag of the forms_xx.xml File 264
Control Tag of the forms_xx.xml File 265
Types of Controls for the Control Tag of the forms_xx.xml File 267

XML Code to Customize Toolbars 273
Example Code of the toolbars.xml File 273
Toolbars Tag of the toolbars.xml File 273

XML Code to Customize Dialog Boxes 275
Dialog Tag of the dialogs.xml File 275
Layout Tag of the dialogs.xml File 275
Appearance Tag of the dialogs.xml File 275

XML Code to Customize Views 276

XML Code to Customize the SalesBook Control 277
Example Code of the lookup_view_defs.xml File 278
Array Tag of the lookup_view_defs.xml File 278
Lookup View Definition Tag of the lookup_view_defs.xml File 279

XML Code for Meta Information 279
SiebelMetaInfo Tag of the siebel_meta_info.xml File 280
Common_settings Tag of the siebel_meta_info.xml File 280
Object Tag of the siebel_meta_info.xml File 281
Field Tag of the siebel_meta_info.xml File 282
Extra_command_options Tag of the siebel_meta_info.xml File 284
Open_with_url_tmpl Tag of the siebel_meta_info.xml File 284
Picklist Tag of the siebel_meta_info.xml File 285
Master_filter_expr Tag of the siebel_meta_info.xml File 285

Appendix C: Additional Code in the Customization Example

XML Code That Defines a Set of Custom Fields 287

XML Code That Defines a Many-To-Many Relationship 289

XML Code That Defines a List 291

XML Code That Creates Cells 293

Glossary

Index

1 What's New in This Release

What's New in Siebel CRM Desktop for Microsoft Outlook Administration Guide, Version 8.0, Rev A

Table 1 lists changes described in this version of the documentation to support release 8.0 of the software.

Table 1. What's New in Siebel CRM Desktop for Microsoft Outlook Administration Guide, Version Version 8.0, Rev A

Topic	Description
"How Siebel CRM Desktop Uses the Microsoft Exchange Server" on page 23	Modified topic. Includes more details about how Siebel CRM Desktop uses the Microsoft Exchange Server.
"How Siebel CRM Assigns the Meeting Organizer" on page 42	New topic. Describes how Siebel CRM assigns the meeting organizer.
"How Siebel CRM Desktop Handles an Appointment" on page 49	Modified topic. Describes how Siebel CRM Desktop uses keys to correlate a Siebel CRM activity with PIM data in Microsoft Outlook.
"How Siebel CRM Desktop Prevents Data Loss if the User Deletes Customization Package Files" on page 65	New Topic. If the user deletes customization package files, then Siebel CRM Desktop restores the customization package files from local storage.
"Administering Metadata Files" on page 78	Modified topic. You are no longer required to add one metadata file at a time. You can now upload all metadata files at one time. Also, the number of metadata files you must upload has changed, including files for individual languages.
"Uploading All Metadata Files at One Time" on page 81	New topic. You can upload all metadata files at one time.
"Guidelines for Assigning a Responsibility to a Customization Package" on page 84	Modified topic. A customization package requires a responsibility. A customization package cannot contain another customization package. A metadata file can belong to more than one customization package.
"Caution About Changing the Default Mail Delivery Location" on page 86	New topic. You must not change the default email delivery location in the Microsoft Outlook profile where you install Siebel CRM Desktop.
"Changes That Siebel CRM Desktop Makes During Installation" on page 87	New topic. Describes changes that Siebel CRM Desktop makes during installation of Siebel CRM Desktop to the file system, Windows Registry, and settings in Microsoft Outlook.

Table 1. What's New in Siebel CRM Desktop for Microsoft Outlook Administration Guide, Version Version 8.0, Rev A

Topic	Description
"Customizing How Siebel CRM Desktop Connects to the Internet" on page 95	New topic. You can customize how Siebel CRM Desktop connects to the Internet.
"Customizing How Siebel CRM Desktop Shares Native Microsoft Outlook Items" on page 98	New topic. You can customize Siebel CRM Desktop to share or not share any new native Microsoft Outlook items that the user creates in Microsoft Outlook.
"Guidelines for Installing the Siebel CRM Desktop Add-In for Multiple Users" on page 99	Modified topic. Includes new instructions.
"Controlling the Synchronization Intervals That Siebel CRM Desktop Displays in the Synchronization Tab" on page 114	New topic. Describes how to control the synchronization intervals that Siebel CRM Desktop displays in the Synchronization Tab of the CRM Desktop - Options dialog box.
"Controlling the Fields That Are Available in a Filter" on page 115	New topic. To control the fields that are available in a filter, you use the IsHidden property of the field in the siebel_meta_info.xml file.
"Controlling How Siebel CRM Desktop Converts Contacts" on page 116	New topic. First Run Assistant prompts the user to convert Microsoft Outlook contacts to Siebel CRM contacts. You can disallow this feature.
"How the Number of Deleted Records Determines Delete Confirmation" on page 117	New topic. Describes how the number of records that the user has deleted determines whether or not Siebel CRM Desktop displays the Confirm Synchronization tab.
"Configuring the Exclusions List" on page 122	New topic. The Exclusions List allows the user to exclude an individual record from synchronization. You can configure how Siebel CRM Desktop uses this list.
"Controlling How Siebel CRM Desktop Handles an Archived Item" on page 123	New topic. If Microsoft Outlook archives an item, then Siebel CRM Desktop interprets this action as a deletion, and it deletes the corresponding Siebel CRM record in the Siebel database. You can control this behavior.
"Troubleshooting Problems That Occur with Siebel CRM Desktop" on page 134	New topic. Describes how to troubleshoot problems that occur with Siebel CRM Desktop. It includes several new troubleshooting topics and a topic that describes how to enable SOAP dumps.
"JavaScript Files in the Customization Package" on page 147	New topic. Describes the JavaScript files that Siebel CRM Desktop includes in the customization package.
"Customizing Language for the Forms Files" on page 153	New topic. To customize the behavior of the forms_xx.xml file that comes predefined with Siebel CRM Desktop, you can use a forms file that is specific to a language.

Table 1. What's New in Siebel CRM Desktop for Microsoft Outlook Administration Guide, Version Version 8.0, Rev A

Topic	Description
"Customizing the Product Name" on page 156	New topic. You can customize the product name that Siebel CRM Desktop displays in the client.
"Making a Field in Siebel CRM Desktop Read-Only" on page 163	New topic. You can make a field in Siebel CRM Desktop read-only.
"Adding a Default Value to a Field in Siebel CRM Desktop" on page 164	New topic. Describes how to configure Siebel CRM Desktop to populate a field with a default value when the user creates a new record.
"Preventing the User From Deleting Records" on page 165	New topic. You can configure Siebel CRM Desktop to prevent the user from deleting records in the Siebel CRM Desktop client.
"Defining Validation Rules for a Phone Number" on page 180	Modified topic. Includes new XML code.
"Process of Adding a Predefined Siebel Picklist to Siebel CRM Desktop" on page 185	New topic. You can add a picklist to Siebel CRM Desktop.
"Process of Adding an MVG Field" on page 203	New topic. You can add an MVG field to Siebel CRM Desktop.
"How Siebel CRM Desktop Handles an Item That is Marked Private" on page 231	New Topic. Siebel CRM does not display the contents of an item to any user who interacts with the Siebel Server except the item owner.
"XML Code That Defines a Set of Custom Fields" on page 287	Modified topic. Includes new XML code.
"XML Code That Defines a Many-To-Many Relationship" on page 289	Modified topic. Includes new XML code.
How CRM Desktop Maps the Private Field of an Activity	Topic removed. The user cannot share a private Microsoft Outlook appointment or task in Siebel CRM Desktop. These items are not synchronized with the Siebel Server.
How CRM Desktop Maps the Alarm and Alarm Lead Fields	Topic removed. Siebel CRM Desktop no longer synchronizes reminders with Siebel alarms.

What's New in Siebel CRM Desktop for Microsoft Outlook Administration Guide, Version 8.0

Siebel CRM Desktop for Microsoft Outlook, Version 8.0 is the first release of this new product. Table 2 lists the chapters for this guide.

Table 2. Chapters in Siebel CRM Desktop for Microsoft Outlook Administration Guide, Version 8.0

Topic	Description
"Overview of Siebel CRM Desktop for Microsoft Outlook" on page 15	Describes an overview of the Siebel CRM Desktop solution, including descriptions, comparisons, and explanations of daily uses.
"How Siebel CRM Desktop Works" on page 19	Describes how Siebel CRM Desktop operates, including synchronization, how synchronization conflicts are handled, and which Siebel Server components Siebel CRM Desktop uses.
"How Siebel CRM Desktop Handles Siebel CRM Data" on page 35	Describes how Siebel CRM Desktop maps certain data between Siebel CRM and Microsoft Outlook.
"How Siebel CRM Desktop Synchronizes Data" on page 59	Describes how Siebel CRM Desktop uses Web services and integration objects during synchronization.
"Installing Siebel CRM Desktop" on page 77	Describes how to prepare the Siebel Server and install the Siebel CRM Desktop application.
"Administering Siebel CRM Desktop" on page 105	Describes how to administer Siebel CRM Desktop, including changing behavior of the Siebel CRM Desktop application, adding and removing Siebel CRM Desktop, and improving performance.
"Customizing Siebel CRM Desktop" on page 143	Describes how to customize an implementation.
"XML Files Reference" on page 245	Describes the code in the XML files, which is included in the customization package.
"How Siebel CRM Desktop Maps Fields Between Siebel CRM Data and Microsoft Outlook Data" on page 229	Describes how Siebel CRM Desktop maps fields between Siebel CRM data and Microsoft Outlook data.
"Additional Code in the Customization Example" on page 287	Describes XML code used in the customization example.
"Glossary" on page 297	Defines certain terms that this guide uses.

2 Overview of Siebel CRM Desktop for Microsoft Outlook

This chapter describes an overview of Oracle's Siebel CRM Desktop for Microsoft Outlook. It includes the following topics:

- Benefits of Using Siebel CRM Desktop on page 15
- Scenarios for Using Siebel CRM Desktop on page 15

Benefits of Using Siebel CRM Desktop

Siebel CRM Desktop for Microsoft Outlook (Siebel CRM Desktop) is an application that integrates Siebel CRM processes and data in Microsoft Outlook®. It allows the user to perform important and typical CRM work directly in a native Microsoft Outlook environment.

Siebel CRM Desktop centralizes essential business information in the familiar Microsoft Outlook environment. This centralization complements the existing capabilities that a Siebel application provides. These features help the user to perform common interactions. The user can use Siebel CRM Desktop to realize the following benefits:

- **Manage Siebel CRM data.** Manage Siebel CRM data and link this data to CRM records directly in Microsoft Outlook. The user can manage appointments, emails, contacts, accounts, activities, opportunities, and so forth directly in Microsoft Outlook.
- **Synchronize data.** Perform bidirectional, incremental synchronization between Siebel CRM Desktop on the Microsoft Outlook Client and the Siebel application on the Siebel Server, which helps to keep the data in these applications up-to-date and consistent.
- **Work while disconnected.** Perform work even when disconnected from the corporate network.

For your organization, some benefits include:

- Increased user adoption of your business processes and tools. Siebel CRM Desktop does not require the user to use an application that the user is not familiar with.
- Increased accessibility to data because the user is not required to log into the Siebel application to view and maintain CRM data.
- Decreased training costs. Most users are already familiar with Microsoft Outlook. You can focus your training on CRM processes instead of spending time and resources on learning how to perform simple interactions.

Scenarios for Using Siebel CRM Desktop

This topic describes several scenarios of how you can use Siebel CRM Desktop with Microsoft Outlook. It includes the following topics:

- "Scenario for Working with an Activity That Is Associated with an Opportunity" on page 16

- "Scenario for Managing Contact Information" on page 16
- "Scenario for Managing Account Information" on page 17
- "Scenario for Associating an Email with an Opportunity" on page 17
- "Scenario for Managing an Opportunity" on page 18

Scenario for Working with an Activity That Is Associated with an Opportunity

This scenario gives one example of how you might use Siebel CRM Desktop with an activity that is associated with an opportunity. You might use Siebel CRM Desktop differently, depending on your business model.

On Friday afternoon, a sales manager reviews the Big Deal opportunity and realizes that it has been inactive for some time. The Manager does the following work in native Microsoft Outlook to assign the task to a sales representative:

- Create a new Microsoft Outlook task
- Associate the relevant opportunity to the task and complete other details of the task
- Send the task to the assigned owner

On Monday, the representative uses Microsoft Outlook to view opportunities. The representative examines the opportunity and notices the new activity, notices the assignor, and realizes that the demonstration must be revised. To examine more information the representative can drill down on the activity record. On Thursday, the representative finishes revising the demonstration and changes the activity status to Done.

Scenario for Managing Contact Information

This scenario gives one example of how you might use Siebel CRM Desktop to manage contact information. You might use Siebel CRM Desktop differently, depending on your business model.

A sales representative works at High-Tech Office Expo and manages many customers, including a customer named Company Y. While at High-Tech Office Expo, the sales representative meets a new contact who is the CEO of Company X, which is a competitor of Company Y. The sales representative also encounters an old college friend who just moved to town. They trade contact information.

The sales representative creates a new contact in Microsoft Outlook and enters information about the CEO in this new contact record. While creating this contact, the representative links the contact to the existing Company X account. Next, the representative uses a scanner to scan the business card of the CEO, and then attaches the scanned image to the contact. Because the representative must share this contact with colleagues, the representative clicks the Sharing Bar. Siebel CRM Desktop then marks the contact as shared and changes the Sharing Bar to an orange color, which indicates that the contact is shared.

In the same Contacts folder, the sales representative creates a new contact for the old college friend. Because the default option does not share the contact with the Siebel Server, the contact remains private.

The sales representative must call the VP of Sales at Company Y, who is represented in the Siebel CRM data as another contact. The representative finds the contact record, and then finds the cell phone number for the contact. The contact record for the VP displays in Microsoft Outlook along with all the other contacts that the representative can normally view in the Siebel Web Client. Assume another user at High-Tech Office Expo updates the contact information for the VP. When the sales representative synchronizes, Siebel CRM Desktop displays this updated information directly in Microsoft Outlook.

Scenario for Managing Account Information

This scenario gives one example of how you might use Siebel CRM Desktop might to manage account information. You might use Siebel CRM Desktop differently, depending on your business model.

Company Z is one of the smaller accounts that a sales representative manages. Because Company Z recently relocated, the representative must update the address details that are associated with the account. The representative drills down on the account in the accounts view, and then enters the new address in the form.

To make sure that everyone on the account team is aware of the new location, the representative must send an email to the account team. To include all account team members, the representative uses the Email to Account Team button in Microsoft Outlook. Siebel CRM Desktop enters email addresses for the entire account team in an email message and then displays the message. The representative types in a brief note about the new location and then sends the email.

While the representative is in the account record for Company Z, the representative decides to add a new contact to the account. To do this, the representative types the contact name in the text box under the Contacts section, and then Siebel CRM Desktop displays the names of contacts that already exist. To create an association, the representative chooses one of these contacts and then clicks Add. As an alternative, the representative can click the SalesBook icon, and then choose an existing contact or create a new contact.

Scenario for Associating an Email with an Opportunity

This scenario gives one example of how you might use Siebel CRM Desktop to associate an email with an opportunity. You might use Siebel CRM Desktop differently, depending on your business model.

A sales representative opens an email from an external contact. This email explains that the purchasing director at the external contact changed. The representative knows that this information is important for the Big Deal opportunity. The representative shares this email with the Siebel Server and associates it with the opportunity, thus making sure that the entire sales team who is working on the opportunity is aware that there is a new purchasing director. Siebel CRM Desktop synchronizes this information with the Siebel Server as an activity record that is associated with the opportunity, along with the original email as an activity attachment.

When the representative shares the email with the Siebel Server, the representative can choose the sales data with which the email is associated. To associate the Big Deal opportunity with the email, the representative types the name in the Opportunity control. If the external contact is associated with at least one opportunity, then Siebel CRM Desktop presents a filtered list of values while the representative types the value. The representative chooses the record for the Big Deal opportunity, and then Siebel CRM Desktop links the email to the opportunity.

Scenario for Managing an Opportunity

This scenario gives one example of how you can use Siebel CRM Desktop to manage an opportunity. You might use Siebel CRM Desktop differently, depending on your business model.

After meeting with a contact at a key strategic account, a sales representative learns of an upcoming Request For Proposal (RFP) that might help to improve the sales pipeline for the next quarter. To complete this work, the representative uses Siebel CRM Desktop to create a new opportunity in Microsoft Outlook.

To begin, the representative clicks Opportunity on the toolbar which opens a new opportunity form. The representative enters details of the new opportunity, including the name, related account, lead quality, sales method, sales stage, close date, and so forth. The representative knows two contacts at the account, and that these contacts decide whether to place an order. Therefore, the representative associates these contacts with the opportunity. The representative can choose one or more products and associate them with the opportunity. To indicate the projected value of the opportunity and to describe how that value is distributed across related products for the opportunity, the representative can assign expected revenue values for the opportunity.

When the representative saves these details in Microsoft Outlook and then synchronizes with the Siebel Server, Siebel CRM makes the details available to other users who possess access to the account and contacts.

3 How Siebel CRM Desktop Works

This chapter describes how Siebel CRM Desktop works. It includes the following topics:

- Overview of How Siebel CRM Desktop Works on page 19
- How Siebel CRM Desktop Uses the Siebel Enterprise on page 24
- Metadata That Describes the Siebel CRM Desktop Application on page 30

Overview of How Siebel CRM Desktop Works

This topic describes an overview of how Siebel CRM Desktop works. It includes the following topics:

- "Extensions to the Microsoft Outlook User Interface" on page 19
- "Infrastructure That Siebel CRM Desktop Uses" on page 20
- "Architecture Components That Siebel CRM Desktop Uses" on page 21

Extensions to the Microsoft Outlook User Interface

Siebel CRM Desktop is a composite application that displays Siebel CRM sales data in Microsoft Outlook. To store and render Siebel CRM data, the Microsoft Outlook add-in framework deploys Siebel CRM Desktop to Microsoft Outlook and extends the Microsoft Outlook data model and user interface.

Extensions to the Microsoft Outlook user interface allow the user to display Siebel CRM data. The following are some examples of these extensions:

- Custom toolbar buttons
- Custom menu items
- Custom forms that display Siebel CRM data
- Custom controls that are embedded in Microsoft Outlook forms that display Siebel CRM data
- Personalization options dialog box
- Predefined Siebel CRM views in Microsoft Outlook folders. For example, the Opportunities by Account view.

To allow the user to perform a variety of work, Microsoft Outlook uses these extensions. The following are some examples of work that the user can perform:

- Create new Siebel CRM data in Microsoft Outlook.

- Mark a Microsoft Outlook item to share with Siebel CRM data and associated sales data. Siebel CRM Desktop shares accounts and opportunities by default. As an option, it can also share or unshare calendar appointments, tasks, contacts, and email messages.
- View and edit sales data.
- Initiate a standard Microsoft Outlook action, such as sending an email or scheduling a meeting in the context of a sales item.

Siebel CRM Desktop performs a variety of validations on data entry. The following are some examples of validation that Siebel CRM Desktop performs:

- Confirm that the data type is valid for a given field.
- Make sure each required field includes information.
- Make sure certain fields are disallowed, depending on the access rules for conditional data.

Infrastructure That Siebel CRM Desktop Uses

Figure 1 illustrates the infrastructure that Siebel CRM Desktop uses.

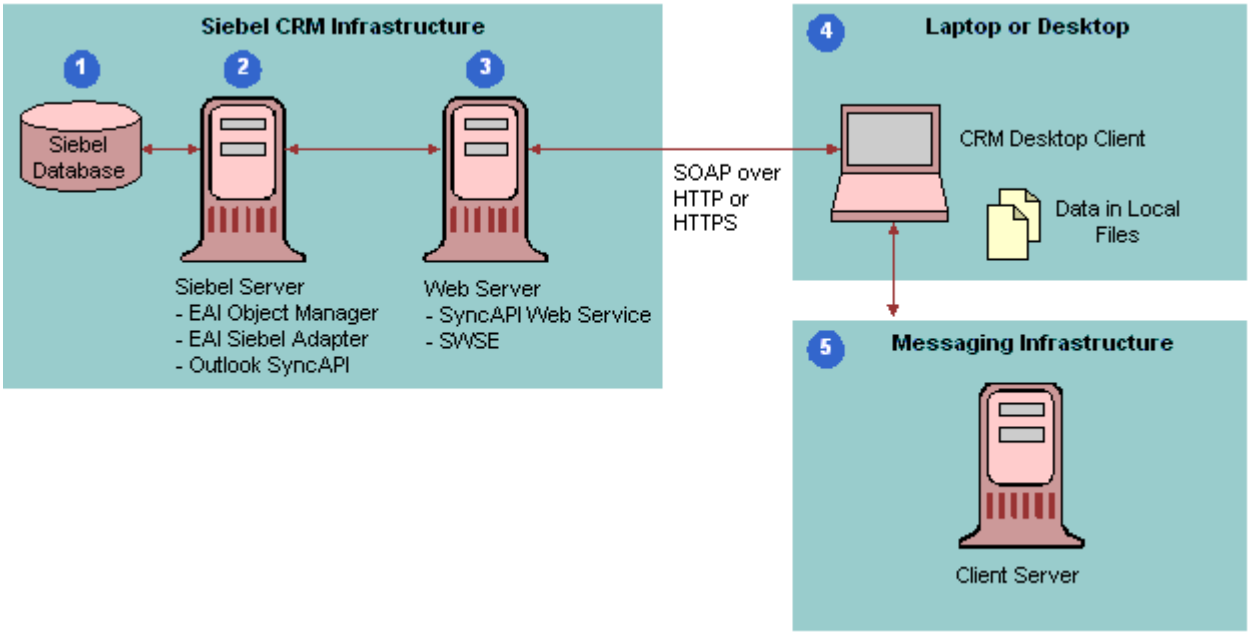


Figure 1. Infrastructure That Siebel CRM Desktop Uses

Siebel CRM Desktop uses the following components:

1 **Siebel Database.** Stores information about opportunities, contacts, service requests, accounts, and so forth.

2 **Siebel Server**. Runs the server components for Siebel CRM Desktop and manages synchronization sessions with the Web Server. Hosts the processes that Siebel CRM Desktop requires to support synchronization and handles incoming synchronization requests from the client through the Web Server. For more information, see "How Siebel CRM Desktop Uses the Siebel Enterprise" on page 24.

3 **Web Server**. Hosts the Siebel Web Server Extension that establishes a user session with the Siebel Server. Handles incoming requests to Siebel Web services, and helps route the client synchronization request to the proper component in the Siebel CRM infrastructure that supports the Web service. For more information, see "About the Web Service API" on page 26.

4 **Laptop or Desktop.** The computer where Siebel CRM Desktop is installed. Siebel CRM Desktop is implemented as a Microsoft Outlook add-in. After you install Siebel CRM Desktop, it deploys binary files that support the integration with Microsoft Outlook, custom user interface capabilities, and the Synchronization Engine. After the first synchronization, CRM Desktop metadata and Siebel CRM data is present on the computer. The customization package includes features that allow synchronization and customization. For more information, see "Customizing the First Run Assistant" on page 93.

5 **Messaging Infrastructure.** Handles local email, calendar items, contacts, and tasks. The messaging infrastructure provides support for mobile and Web access to information and for storing data. For more information, see "How Siebel CRM Desktop Uses the Microsoft Exchange Server" on page 23.

Architecture Components That Siebel CRM Desktop Uses

Figure 2 illustrates the major architectural components that Siebel CRM Desktop uses. For more information, see *Siebel Deployment Planning Guide*.

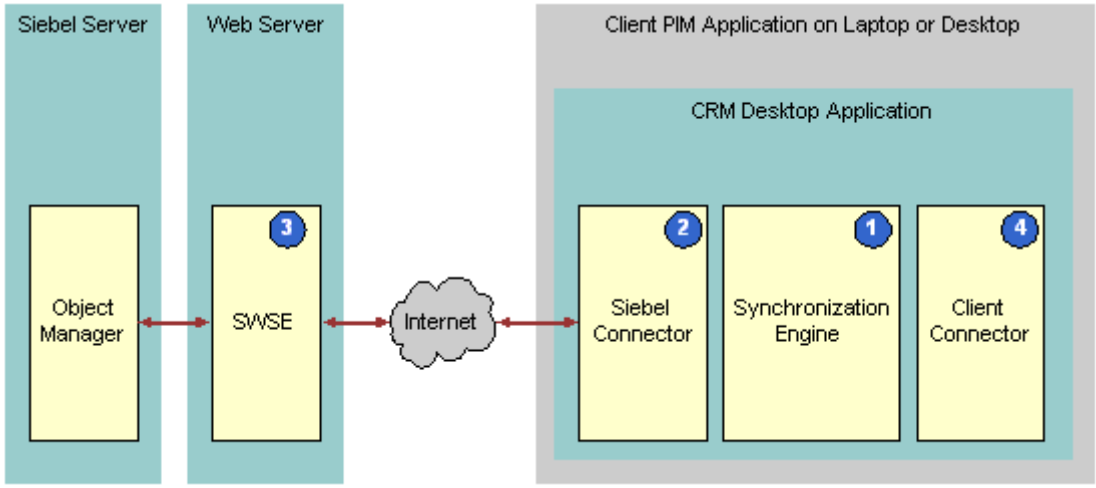


Figure 2. Architecture Components That Siebel CRM Desktop Uses

Siebel CRM Desktop uses the following components:

1. **Synchronization Engine.** Starts the synchronization process. Determines the changes that Siebel CRM Desktop requires to synchronize between the client and the Siebel Server, as determined by the differences between the data sets that are available in each system. To get information from the Siebel Server, submits requests to the connector and then, to determine the required data changes, processes the replies. Works with the Microsoft Outlook connector to make the necessary data changes in the data storage in Microsoft Outlook.
2. **Siebel Connector.** Connects the personal information manager (PIM) client to the Siebel Server. Submits requests and receives replies and works with the Synchronization Engine. The connector interfaces with the Siebel Server through the Web service infrastructure.
3. **SWSE (Siebel Web Server Extension).** Brokers requests from the Siebel Connector to the Siebel Server.
4. **Client Connector**. Provides the Synchronization Engine with access to the data storage in Microsoft Outlook. Supports queries, inserts, updates, and deletes of data in this data storage.

Overview of How Siebel CRM Desktop Synchronizes Data

To synchronize data between Microsoft Outlook and the Siebel Server, Siebel CRM Desktop uses a process that the client controls. Once installed, the client initializes the Siebel CRM data that is available in Microsoft Outlook through the initial synchronization. An incremental synchronization synchronizes subsequent changes that occur in Microsoft Outlook or on the Siebel Server.

To make Siebel CRM data available in Microsoft Outlook, the Siebel CRM Desktop user must perform the initial synchronization with the Siebel Server. *First Run Assistant* is a wizard that guides the user through the setup of the Siebel CRM Desktop application in Microsoft Outlook. It displays when the user starts Microsoft Outlook for the first time after you install the Siebel CRM Desktop add-in. It starts the initial synchronization.

While using the First Run Assistant, the user can choose among several preferences, and then start the initial synchronization. Siebel CRM Desktop does the following work:

1. Connects Microsoft Outlook to the Siebel Server and authenticates the user.
2. Performs a check to determine the configuration to which the user possesses access. An association with a responsibility that is in turn associated with a customization package determines this access. For more information, see "Relationships Between Users, Responsibilities, Customization Packages, and Metadata Files" on page 30.
3. Downloads and applies the configuration.
4. Synchronizes the appropriate data. The connector configuration, synchronization mappings, visibility rules, default internal filters, and default user filters determine this synchronization.

For more information, see "How Siebel CRM Desktop Synchronizes Data Between the Client and the Siebel Server" on page 59.

About Web Service Usage During Synchronization

A component of the Synchronization Engine that you deploy to the client supports synchronization. This component connects to the Siebel Server through the Web service infrastructure. Web services provide access to synchronization for metadata and synchronization for Siebel CRM data. Siebel CRM Desktop provides access to individual objects through the standard Siebel EAI repository objects, such as integration objects and integration components. These objects acquire data through their relations with business objects and business components.

How Siebel CRM Desktop Uses the Microsoft Exchange Server

Figure 3 illustrates how Siebel CRM Desktop uses the Microsoft Exchange Server.

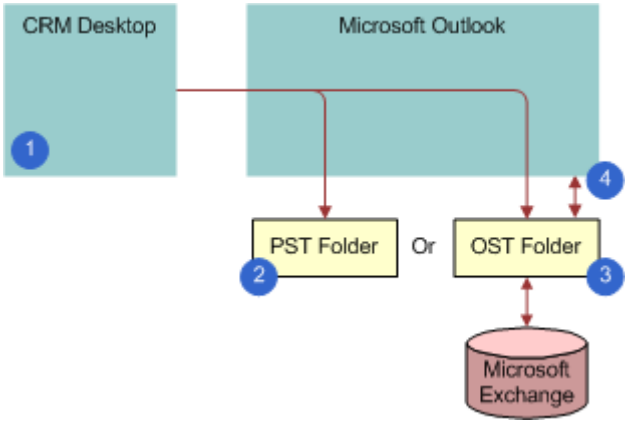


Figure 3. How Siebel CRM Desktop Uses the Microsoft Exchange Server

Siebel CRM Desktop uses the Microsoft Exchange Server in the following ways:

1. You install Siebel CRM Desktop as a COM add-in Microsoft Outlook on the client computer.
2. If the user uses a POP3 email account, then Siebel CRM Desktop accesses a Microsoft Outlook Personal Folders (.pst) file.
3. If the user uses an Exchange Server email account in cached mode, then Siebel CRM Desktop accesses an Offline Storage (.ost) file. It synchronizes Siebel CRM data between Siebel CRM Desktop and the Siebel Server. It stores this data in the.ost file. Microsoft Outlook synchronizes data in the .ost file to the Exchange Server.
4. Siebel CRM Desktop does not interfere with communication between Microsoft Outlook and the Microsoft Exchange Server. Microsoft Outlook synchronizes with the Microsoft Exchange Server just as it does if you do not install Siebel CRM Desktop.

Siebel CRM Desktop does not support an Exchange Server email account that is not cached.

To install and use Siebel CRM Desktop, it is not necessary for you to customize the Microsoft Exchange Server or acquire permissions to access it.

How Siebel CRM Desktop Displays Data in Microsoft Outlook

Siebel CRM Desktop stores Siebel CRM data in files that are native to Microsoft Outlook. It stores this data in one of the following files depending on the data file that the user sets as the default email delivery location in the Microsoft Outlook profile in which you install Siebel CRM Desktop:

- A Microsoft Exchange folder file (.ost file) that is offline and cached
- A personal folders file (.pst file)

Microsoft Outlook *data* is data that is created in the native Microsoft Outlook application. Examples include an appointment or task. *Siebel CRM data* is data that can include the following items:

- Business data that the user creates in the CRM Desktop add-in
- Data that a user creates in the client of a Siebel application, such as Siebel Call Center
- Data that resides in the Siebel database on the Siebel Server

Examples of Siebel CRM data include an opportunity, account, or activity. Because Siebel CRM Desktop uses native Microsoft Outlook data files, Microsoft Outlook displays Siebel CRM data through native Microsoft Outlook user interface elements, such as lists and forms. Microsoft Outlook can display this data simultaneously with other Microsoft Outlook data while using the same user interface concept, such as a mailbox folder. The Siebel CRM Desktop user can choose a folder that displays Siebel CRM data, and can also view Microsoft Outlook data in an Microsoft Outlook list view.

Siebel CRM Desktop displays Siebel CRM data in the following contexts:

- Microsoft Outlook forms, which are extensions to Microsoft Outlook calendar, contact, email, and tasks.
- Microsoft Outlook items, such as the details of an account or opportunity that is associated with a Microsoft Outlook appointment that is shared with Siebel CRM Desktop.

When disconnected from the Siebel Server, the user interacts with data that the user can access locally in Microsoft Outlook.

How Siebel CRM Desktop Uses the Siebel Enterprise

This topic describes some of the major components in the Siebel Enterprise that Siebel CRM Desktop uses. It includes the following topics:

- "Siebel Enterprise Components That Siebel CRM Desktop Uses" on page 25
- "About the Web Service API" on page 26
- "About the PIM Client Sync Service Business Service" on page 27
- "About the EAI Siebel Adapter Business Service" on page 27
- "About Integration Objects" on page 28
- "User Details Business Component" on page 29
- "About Authentication and Session Management" on page 30

■ "How Siebel CRM Desktop Uses the Microsoft Exchange Server" on page 23

For an illustration of how parts of the Siebel Enterprise fit in Siebel CRM Desktop, see "Siebel Enterprise Components That Siebel CRM Desktop Uses" on page 25.

Siebel Enterprise Components That Siebel CRM Desktop Uses

Figure 4 illustrates Siebel Enterprise components that Siebel CRM Desktop uses.

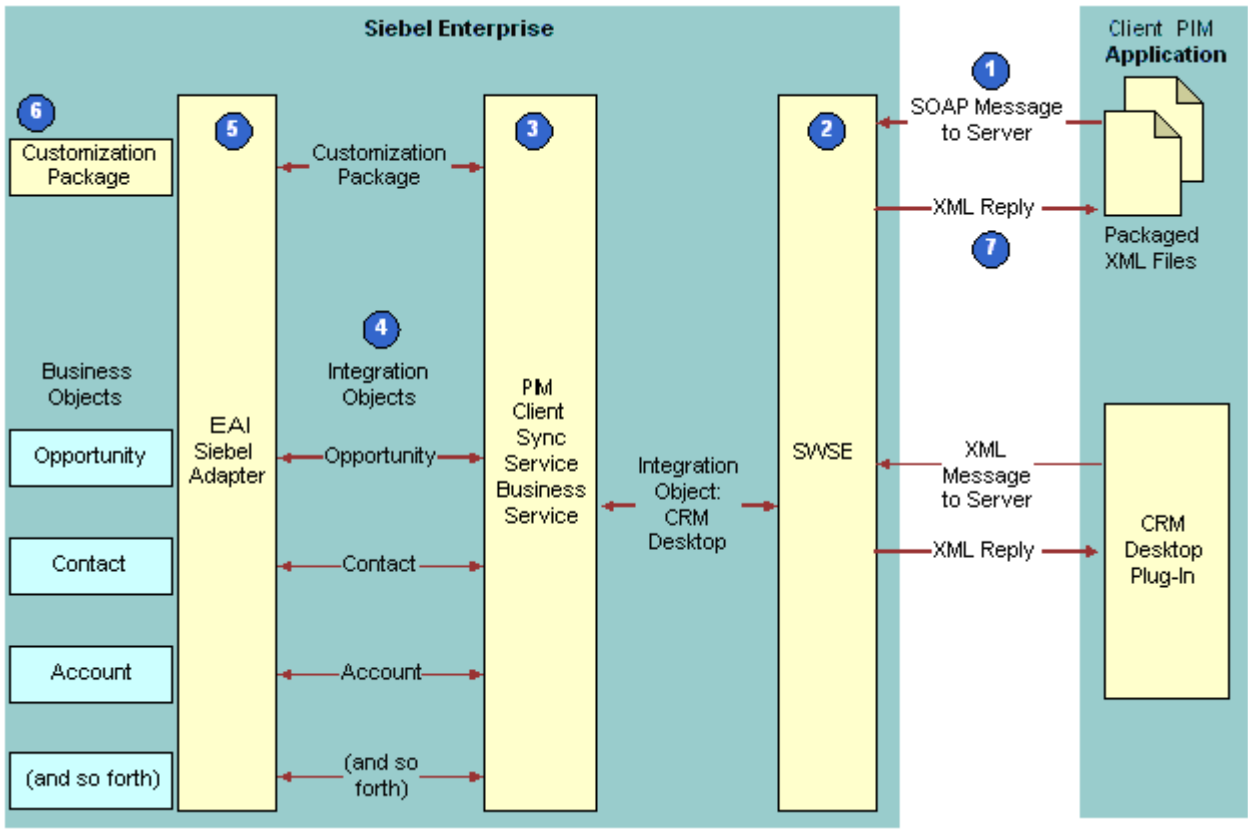


Figure 4. Siebel Enterprise Components That Siebel CRM Desktop Uses

Siebel CRM Desktop uses the following Siebel Enterprise components:

1 **SOAP Message to Server.** To request metadata, sends a SOAP (Simple Object Access Protocol) message to the Siebel Server over HTTP (Hypertext Transfer Protocol) or HTTPS (Hypertext Transfer Protocol Secure). The payload is a Siebel message that **Siebel CRM Desktop** sends in a SOAP envelope.

2 **SWSE (Siebel Web Service Extension)** To handle requests from **Microsoft Outlook** for data and metadata synchronization, **Siebel CRM Desktop** does the following:

- Uses the Web Service API as the endpoint of the Web service.
- Accepts SOAP over HTTP or HTTPS from the Synchronization Engine in Microsoft Outlook.
- To connect to the EAI Object Manager component or another component that provides access to the Web services, uses the configured endpoint.

For more information, see "About the Web Service API" on page 26, and *Configuring Siebel Business Applications*.

3 **PIM Client Sync Service Business Service**. Handles the data synchronization request from the Synchronization Engine, and then passes the request to the EAI Siebel Adapter business service for processing. This business service supports batching, error handling, and a few custom functions, such as providing counts of records and resolving appointment attendees that reside on the Siebel Server. For more information, see "About the PIM Client Sync Service Business Service" on page 27.

4 **Integration Objects**. Requests certain processes through the EAI Siebel Adapter business service. To describe the underlying Siebel data structure, uses integration objects. Specifies objects and fields that are available for synchronization. For more information, see "About Integration Objects" on page 28.

5 **EAI Siebel Adapter.** Sends requests to the data and metadata synchronization Web services. To process the requests through the EAI Siebel Adapter business service, these services use the Execution method and various operations. Example operations include querypage, insert, update, and delete. For more information, see "About the EAI Siebel Adapter Business Service" on page 27.

6 **Customization package.** A collection of XML and JavaScript files that describe the application that runs in Microsoft Outlook. It verifies that the latest application metadata for the user is available on the Siebel Server. For more information, see "About the Customization Package" on page 32.

7 **XML Reply to** Microsoft Outlook. Sends an XML message that contains the metadata that Siebel CRM Desktop requests in Step 1. Siebel CRM Desktop sends this message to Microsoft Outlook over HTTP or HTTPS.

For more information, see "How Siebel CRM Desktop Synchronizes Data Between the Client and the Siebel Server" on page 59.

About the Web Service API

The client calls operations in the web service and then the web service calls the related business service. The web service does not itself implement integration objects and business services. Instead, it references them. To communicate with the Siebel Server, Microsoft Outlook communicates with the Web Service API on the Siebel Server. This Web Service API references the PIM Client Sync Service business service. Communication between Microsoft Outlook and the Siebel Server occurs through SOAP packages that contain embedded Siebel messages. The client merely views the Web Service as an address. The Web Service API includes the following objects:

- Custom Integration Object for communication between the Siebel Server and Microsoft Outlook. For more information, see "About Integration Objects" on page 28.

- PIM Client Sync Service business service that supports synchronization with Microsoft Outlook.
- Integration Objects that support Siebel objects.

The business service that the Web Service API references provides the following functionality:

- Performs bulk insert, update, and delete operations in the Siebel database
- Queries and retrieves objects as determined by a given criterion
- Queries and retrieves objects as determined by a set of object IDs
- Returns a count of the number of object records that match a given criterion
- Retrieves list data for attributes of an object in the Siebel database

About the PIM Client Sync Service Business Service

The PIM Client Sync Service business service delegates calls that Siebel CRM Desktop receives from Microsoft Outlook and contains methods to address calls from the Siebel CRM Desktop application. Each method is generic and is not tightly coupled to a specific Siebel object. A Web service provides access to the business service methods. Siebel CRM Desktop accesses the Web service through SOAP messages. The business service receives and sends the custom integration object as parameters of the method for the business service.

The PIM Client Sync Service business service performs the following steps:

1. Parses the input hierarchy of the object instance.
2. Retrieves commands from the Siebel message, which are embedded in the incoming instance of the integration object.
3. Processes the Siebel messages with the help of methods on the EAI Siebel Adapter business service.
4. Embeds the output of the Siebel message into the instance of the integration object.

   The PIM Client Sync Service business service performs this step to send the information back to Microsoft Outlook through the web service interface.

For more information, see "About Integration Objects" on page 28.

About the EAI Siebel Adapter Business Service

The *EAI Siebel Adapter* business service is a predefined data interface that interacts with the Siebel Object Manager. It does this to access and modify data in the Siebel database. It uses the following process:

1. The EAI Siebel Adapter takes, as input, an XML document or a property set that conforms to the definition of an integration object in the Siebel application.
2. The EAI Siebel Adapter business service queries, inserts, updates, deletes, or synchronizes data with the Siebel business object layer.
3. The PIM Client Sync Service web service calls the PIM Client Sync Service business service.

4 The PIM Client Sync Service business service submits requests to the EAI Siebel Adapter to query, insert, update, or delete data in the Siebel database.

For more information, see "About Integration Objects" on page 28.

About Integration Objects

An *integration object* is an object that contains the contents of the messages that Siebel CRM Desktop exchanges between the Siebel Server and Microsoft Outlook.

Siebel Message Usage with the EAI Siebel Adapter

A *Siebel message* is an instance of an integration object that provides the input to the EAI Siebel Adapter business service. The integration object can carry multiple commands in a single call to the business service. The commands can be grouped together so that the business service can process the commands in a batch. When Siebel CRM Desktop sends an instance of an integration object from Microsoft Outlook, the object is encapsulated in an element of the Siebel message. Each of these messages contains a Siebel message header.

Integration Objects That Siebel CRM Desktop Uses

Siebel CRM Desktop uses integration objects that include the following prefix:

CRMDesktop

You must not create a new integration object for an existing object. Only create a new integration object for a new, custom object that does not already have an integration object. Consider the following examples:

- For accounts, do not create a new integration object. Instead, extend the CRMDesktopAccountIO integration object.
- For Channel Partner, you must create a new integration object. For more information, see "Creating an Integration Object for the Channel Partner MVG" on page 206.

The following list includes some of the integration objects that allow Siebel CRM Desktop to access Siebel CRM data, such as accounts, opportunities, and contacts. Note that this list is only an example. It does not include all the integration objects that Siebel CRM Desktop uses:

- CRMDesktopAccountIO
- CRMDesktopActionExceptionIO
- CRMDesktopActionIO
- CRMDesktopAssignmentGroupIO
- CRMDesktopBusinessAddressIO
- CRMDesktopContactIO
- CRMDesktopCurrencyIO
- CRMDesktopEmployeeIO
- CRMDesktopIndustryIO
- CRMDesktopInternalDivisionIO
- CRMDesktopInternalProductIO
- CRMDesktopListOfValuesIO
- CRMDesktopOpportunityIO
- CRMDesktopPickListGenericIO
- CRMDesktopPickListHierarchicalIO
- CRMDesktopPositionIO
- CRMDesktopSalesCycleDefIO
- CRMDesktopSystemPreferencesIO
- CRMDesktopUserDetailsIO

To meet other integration requirements, Siebel CRM Desktop uses the following integration objects:

- **CRMDesktopLocaleIO.** PIM locale setting integration objects that provide access to the locale settings.
- **CRMDesktopSystemPreferencesIO.** An integration object for a PIM system preference that provides access to the system preferences for the Siebel Server.
- **CRMDesktopUserDetailsIO.** For login user data.
- **PIMClientMetaData and PIMClientMetadataFile.** For metadata file download.
- **PIMClientSync.** A wrapper that includes other integration objects.

User Details Business Component

The User Details business component is a clone of the Employee business component except for the Currency, Login Id, and Language fields. Siebel CRM Desktop uses it to retrieve the default user values. Calculated fields in the User Details business component provide access to the Currency, Login Id, and Language fields. The User Details business object references the User Details business component. The CRMDesktopUserDetailsIO integration object provides access to the following fields in Microsoft Outlook:

- LoginId
- LoginName
- Currency
- Position
- PositionId
- OrganizationId

- OrganizationName
- Language

About Authentication and Session Management

The Siebel Server provides a lightweight context management facility for Web service authentication. To manage authentication with this facility, Siebel CRM Desktop uses a combination of user credentials and a SessionID token. When user credentials are presented in the SOAP header of a Web service request, Siebel CRM Desktop performs formal authentication before it runs the Web service operation. If the authentication succeeds, then the operation proceeds and Siebel CRM Desktop places a special SessionID token in the SOAP header of the Web service reply.

When Microsoft Outlook includes the SessionID in subsequent Web service requests, Siebel CRM Desktop uses that SessionID to restore cached session information, thus bypassing the substantially more expensive process of reexecuting the authentication. If presented with the SessionID and a valid set of user credentials, then Siebel CRM Desktop attempts to use the SessionID before it resorts to the user credentials and reauthentication. The session that the SessionID tracks is subject to expiration and other security checks.

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

Metadata That Describes the Siebel CRM Desktop Application

Siebel CRM Desktop is defined in metadata so that you can customize it to meet your business requirements. This topic describes the structure of the metadata that Siebel CRM Desktop uses and how you can modify the metadata to support a customization. It includes the following topics:

- "Relationships Between Users, Responsibilities, Customization Packages, and Metadata Files" on page 30
- "About the Customization Package" on page 32
- "About Metadata Files" on page 33

For more information, see "Customizing the Siebel CRM Desktop Application" on page 143.

Relationships Between Users, Responsibilities, Customization Packages, and Metadata Files

A Siebel CRM Desktop user is associated with a given Siebel CRM Desktop configuration through a responsibility that a customization package references. When the package is activated and published, a user who is associated with the responsibility can download the configuration that is defined in the package. This configuration is a collection of metadata files that Siebel CRM Desktop stores on the Siebel Server and downloads to Microsoft Outlook during synchronization.

Figure 5 illustrates an example of how several users, U1, U2, and U3, are related to several responsibilities, R1, R2, R3, R4, and R5, and how these responsibilities are related with several customization packages, P1, P2, and P3.

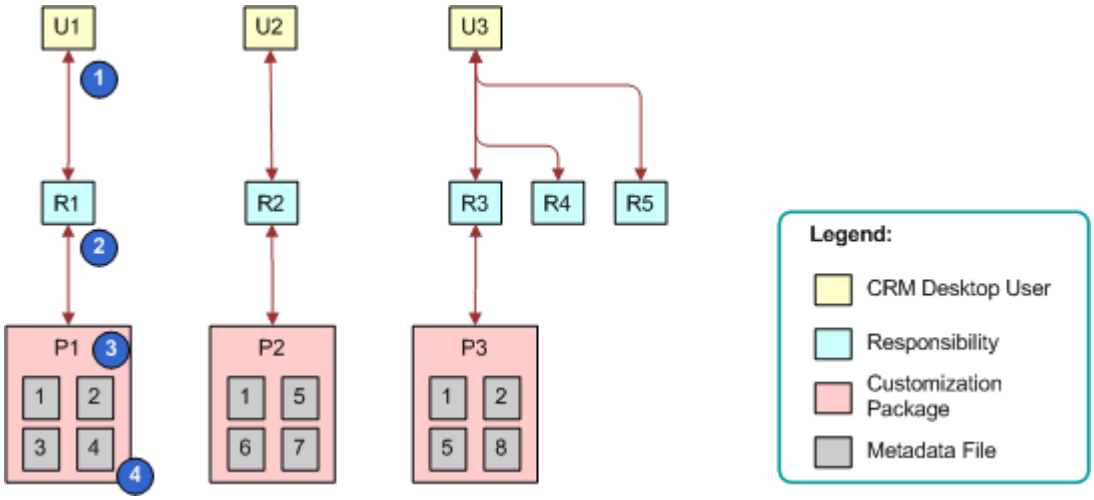


Figure 5. Example of Relationships Between Users, Responsibilities, Customization Packages, and Metadata Files

The following relationships exist between users, responsibilities, customization packages, and metadata files:

1. **CRM Desktop User**. The user of an implementation of Siebel CRM Desktop.
2. **Responsibility.** A Siebel responsibility, such as Sales Representative. It corresponds to the job role that the user performs.
3. **Customization package.** A collection of metadata files that Siebel CRM Desktop associates with a particular responsibility. For more information, see "About the Customization Package" on page 32.
4. **Metadata File**. A description of Siebel CRM Desktop that Siebel CRM Desktop deploys to Microsoft Outlook as XML code or JavaScript files. For more information, see "About Metadata Files" on page 33.

How Siebel CRM Desktop Allows the User to Access Siebel CRM Data

Siebel CRM Desktop allows the user to access Siebel CRM data in Microsoft Outlook in the following ways:

- **Through responsibility.** Similar to how a view allows the user to access data in the Siebel Web Client. Siebel CRM Desktop associates the user with a customization package through a responsibility. This association determines which application metadata Siebel CRM Desktop sends to the user through metadata synchronization. The metadata defines which objects Siebel CRM Desktop synchronizes with Microsoft Outlook. For more information, see "Guidelines for Assigning a Responsibility to a Customization Package" on page 84.

- **Through synchronization filters**. The customization package includes metadata files that specify which data access control and which filters to apply when Siebel CRM Desktop synchronizes data with the Siebel Server. For example, the default configuration specifies that the user can synchronize accounts, contacts, and opportunities that are associated with the sales team to which the user is assigned.

Depending on your business requirements, you can define different customization packages and assign them to different users through responsibilities. To meet individual user requirements, you can define different access control and synchronization filters for each customization package.

About the Customization Package

A *customization package* is a collection of XML metadata files and JavaScript files that Siebel CRM Desktop associates with a particular responsibility. Siebel CRM Desktop deploys a customization package when the user synchronizes the metadata. This synchronization determines which customization package is available to the user, and then retrieves it from the Siebel Server. To customize your deployment, you can modify the metadata files. The following are some examples of customizations you can make:

- Add or remove fields that Siebel CRM Desktop synchronizes.
- Change the layout of a custom form in the client.
- Change a control that Siebel CRM Desktop deploys to Microsoft Outlook.
- Change a security rule.

For more information, see Chapter 8, “Customizing Siebel CRM Desktop.”

The customization package describes the following information:

- The extensions to the Siebel CRM Desktop user interface, which includes Microsoft Outlook views, forms, lookup controls, and toolbars.
- Translated text strings that Siebel CRM Desktop uses to create prompts and labels in Microsoft Outlook.
- Data validation and security logic.
- Descriptions of synchronization preset filters and view modes that Siebel CRM Desktop uses during synchronization.
- Criteria that Siebel CRM Desktop uses to detect duplicate objects that occur in Siebel CRM Desktop.
- Business logic that JavaScript provides.
- Data mapping between Siebel fields and Microsoft Outlook fields.

If you change the data model, then the customization package performs a complete resynchronization.

About Metadata Files

A *metadata file* is an XML or JavaScript file that describes the Siebel CRM Desktop application. Siebel CRM Desktop uses these files in the following ways:

- **XML files.** Describes the default synchronization objects, synchronization mapping, custom views and forms in Microsoft Outlook, and so forth.
- **JavaScript files.** Describes business logic that Siebel CRM Desktop uses for data validation, custom actions that Siebel CRM Desktop provides access to in toolbars, and other custom processing that Siebel CRM Desktop performs in Microsoft Outlook.

The following items describe how Siebel CRM Desktop uses metadata files with a customization package:

- A customization package contains a collection of metadata files. These files describe the entire Siebel CRM Desktop application that you deploy to Microsoft Outlook.
- A customization package consists of a set of metadata files.
- Siebel CRM Desktop requires that a customization package include a minimum number of files. These files are described in "About Files in the Customization Package" on page 144.
- You can use a single metadata file with more than one customization package. These metadata files can be part of another customization package. Figure 5 on page 31 illustrates this relationship where the same metadata file occurs in different packages. For example, packages 1 and 3 include metadata file 2.
- Siebel CRM Desktop associates a customization package with a single Siebel responsibility. It associates a user with the responsibility as a way to acquire access to the customization package and the Siebel CRM Desktop configuration that you deploy to Microsoft Outlook.

For more information, see "About Files in the Customization Package" on page 144.

How Siebel CRM Desktop Reuses, Modifies, and Updates Metadata Files

Siebel CRM Desktop associates each customization package with a collection of metadata files, and it can associate each metadata file with more than one customization package.

After Siebel CRM Desktop associates a metadata file with an active deployment package, you cannot modify the metadata file. To create a new metadata file, you can export, modify, and then reimport the metadata file. Even though multiple customization packages can reference the same metadata files, it is typical for one or more metadata files that different customization packages use to contain different information.

For example, the Microsoft Outlook Sales Representative responsibility is different and separate from the Microsoft Outlook Sales Manager responsibility. Although you can use an existing responsibility and associate it with a customization package, it is recommended that you create a new responsibility. This technique provides you with more control over which users Siebel CRM Desktop associates with which customization package.

For more information, see "Republishing a Customization Package" on page 127.

4 How Siebel CRM Desktop Handles Siebel CRM Data

This chapter describes how Siebel CRM Desktop handles certain types of Siebel CRM data. It includes the following topics:

- How Siebel CRM Desktop Handles an Activity on page 35
- How Siebel CRM Desktop Handles a Shared Activity on page 44
- How Siebel CRM Desktop Handles Microsoft Outlook Calendar on page 47
- How Siebel CRM Desktop Handles a Microsoft Outlook Task on page 56
- How Siebel CRM Desktop Handles a Microsoft Outlook Email Message on page 56
- How Siebel CRM Desktop Handles Items If the User Removes the Siebel CRM Desktop Add-In on page 57
- How a User Can Link a Siebel CRM Record to a Microsoft Outlook Record on page 58

How Siebel CRM Desktop Handles an Activity

This topic describes how Siebel CRM Desktop handles an activity. It includes the following topics:

- “Overview of How Siebel CRM Desktop Handles an Activity” on page 35
- “How an Activity Is Created or Modified” on page 37
- “How Siebel CRM Desktop Processes an Activity” on page 37
- “How Siebel CRM Desktop Resolves Participants and Email Recipients of an Activity” on page 39
- “How Siebel CRM Desktop Displays an Activity in Microsoft Outlook” on page 41
- “How Siebel CRM Desktop Sets the Primary Employee of an Activity” on page 41
- “How Siebel CRM Desktop Handles an Attachment” on page 44

Overview of How Siebel CRM Desktop Handles an Activity

In Siebel CRM, an *activity* is a work item that the user must track or display as an interaction. The following items are examples of activities:

- A To do item
- An email sent to a contact
- An appointment that includes a contact

In the client of a Siebel application, such as the Mobile Web Client, Siebel CRM can display an activity in the Activities screen or in the Calendar. The Display In field of the Activities list determines where in Microsoft Outlook an activity displays. The following values are included:

- Calendar and Activities
- To Do and Activities
- Activities only
- Communication and Activities

The Type field specifies the type of activity. It can contain a wide range of possible values. For example:

- Appointment
- Field Repair
- Email-Outbound
- Research

To support an activity in Microsoft Outlook, Siebel CRM Desktop uses one of the following custom Siebel CRM activity objects:

- Calendar item, which is a meeting or appointment.
- Task item, which is a To Do. For example:
  - Book a flight
  - Review new proposal
- Other item, which is a record of something that happened. For example:
  - Correspondence was sent
  - Demonstration was given

A Siebel CRM activity does not map to a single native object in Microsoft Outlook. Instead, Siebel CRM Desktop synchronizes an activity from the Siebel Server to the Siebel CRM Desktop Client as a custom activity record rather than as a Microsoft Outlook Task or Calendar item. After synchronization, Siebel CRM Desktop does the following:

- Creates Microsoft Outlook calendar item that matches the calendar item from Siebel CRM
- Creates a Microsoft Outlook Task item that matches the To Do item from Siebel CRM

If the activity is a Siebel CRM activity, then Siebel CRM Desktop creates a shared appointment in the Outlook Calendar.

Because Microsoft Outlook does not synchronize directly between native Microsoft Outlook items and records on the Siebel Server, Siebel CRM Desktop uses the Siebel CRM activity as an intermediary between a native Microsoft Outlook item that resides in the user mailbox and a Siebel CRM activity that resides on the Siebel Server. If the user creates a shared Microsoft Outlook appointment, email, or task, then Siebel CRM Desktop creates another item in Microsoft Outlook that represents the Siebel CRM activity record in addition to the native Microsoft Outlook item that is shared.

How an Activity Is Created or Modified

The user can use one of the following ways to perform read, update, create, and delete operations:

- Use the Add Activity button.
- Use a form for an item in Microsoft Outlook that includes a relationship with an activity, such as all the activities for an account. This form allows the user to link the activity with a Siebel CRM record in Microsoft Outlook, such as an account, opportunity, or contact, and to display the link to the corresponding activity.

Siebel CRM Desktop can create an activity for an item in Microsoft Outlook, such as an appointment in the calendar, a task, or an email.

How Siebel CRM Desktop Processes an Activity

To process an activity, Siebel CRM Desktop uses the following types of objects:

- A native Microsoft Outlook item, such as an appointment in the calendar, an email, or a task
- A Siebel CRM activity record in Microsoft Outlook that Siebel CRM Desktop synchronizes from the Siebel Server
- A Siebel CRM activity record on the Siebel Server

Figure 6 illustrates the relationships between Microsoft Outlook items in Microsoft Outlook, Siebel CRM records in Microsoft Outlook, and Siebel CRM records on the Siebel Server. Multiple activity types map to the Display In value of the Activities Only list. For example, demos, and so forth. For brevity, Figure 6 does not include these types.

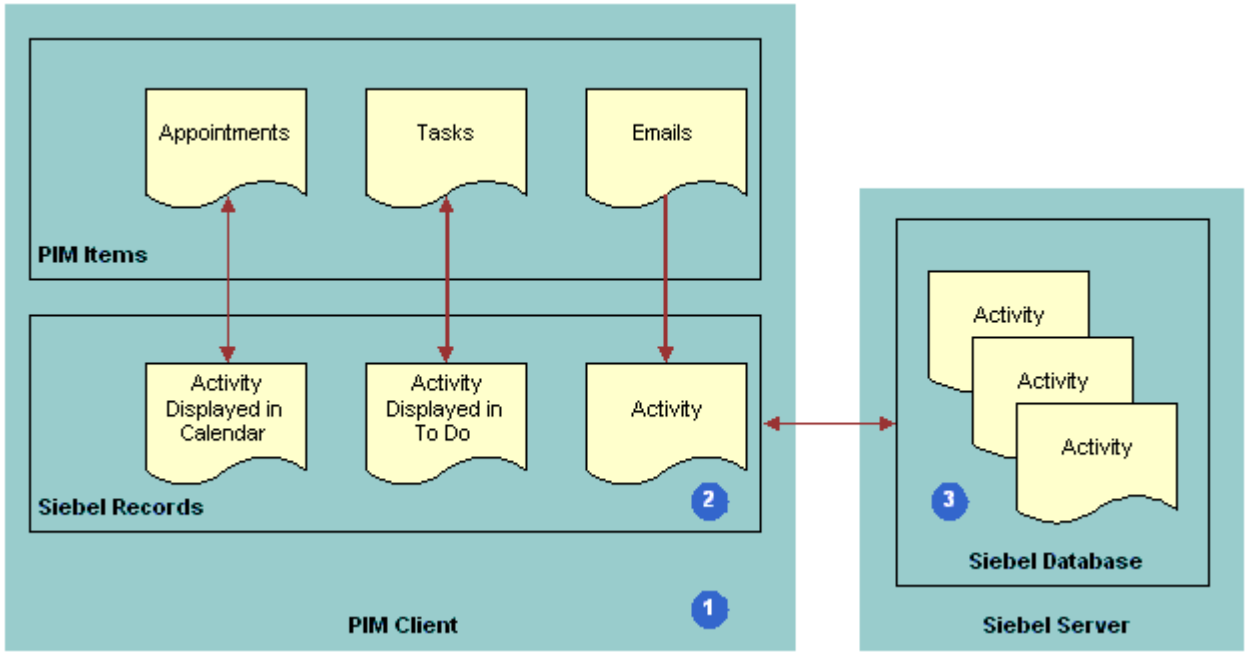


Figure 6. Relationship Between Microsoft Outlook Items, Siebel Activities in Microsoft Outlook, and Siebel Activities on the Siebel Server

To process an activity, the following work occurs:

1. An activity is created in Microsoft Outlook. For more information, see "How an Activity Is Created or Modified" on page 37.
2. Siebel CRM Desktop adds a record as a Siebel CRM activity in Microsoft Outlook.

   If the user marks as shared a Microsoft Outlook appointment, email, or task, and then saves and closes this item, then Siebel CRM Desktop immediately creates the Siebel CRM activity in Microsoft Outlook.
3. During synchronization, Siebel CRM Desktop maps the Siebel CRM activity in Microsoft Outlook one-to-one with the corresponding activity in the Siebel database on the Siebel Server. If the user shares a new Microsoft Outlook item or creates an activity in Microsoft Outlook, then Siebel CRM Desktop uploads this activity during synchronization to the Siebel Server and inserts it in the Siebel database. For more information, see "How Siebel CRM Desktop Creates a Corresponding Native Microsoft Outlook item" on page 39.

How Siebel CRM Desktop Creates a Corresponding Native Microsoft Outlook item

When synchronizing data, if the user possesses the rights to view the activity, and if the activity meets the requirements that the filter settings for the user defines, then Siebel CRM Desktop downloads to Microsoft Outlook any new activities that reside on the Siebel Server.

Table 3 describes how Siebel CRM Desktop creates a corresponding native Microsoft Outlook item depending on the settings of the Display In field.

Table 3. How Siebel CRM Desktop Creates a Corresponding Native Microsoft Outlook item

Display In Value for the Siebel CRM Activity	Description
Calendar and Activities	Siebel CRM Desktop creates a calendar item as an appointment only. If the Siebel Start date is not set for the activity, then Siebel CRM Desktop does not create a calendar item in Microsoft Outlook Calendar. The calendar item is linked to this activity.
To Do and Activities	Siebel CRM Desktop creates a native Microsoft Outlook task that is linked with this activity.
Not Calendar and Activities or To Do and Activities	Siebel CRM Desktop synchronizes the activity as visible under the appropriate parent object, such as an Account Activity, Contact Activity, and so forth.
Activities Only	If an activity is Activities Only, and if this activity is not associated with another item to which the user possesses visibility, such as an account or contact, then Siebel CRM Desktop synchronizes the activity but does not display it in the Siebel CRM Desktop client. Instead, it stores it in a hidden Activities folder that is not visible to the user.

How Siebel CRM Desktop Resolves Participants and Email Recipients of an Activity

This topic describes how Siebel CRM Desktop resolves participant lists and email recipients in Microsoft Outlook calendar.

How Siebel CRM Desktop Resolves Attendees of a Meeting

Siebel CRM Desktop does the following work:

- If the meeting organizer adds an email in the To line, then Siebel CRM Desktop creates an association with an employee or contact.
- If the meeting organizer uses an MVG (multi-value group) in the meeting form to add an association, then Siebel CRM Desktop adds the email address to the To line.

- If the meeting organizer uses an MVG in the activity form to add an association, then Siebel CRM Desktop does not update the To line. This allows the user to associate the Siebel CRM activity with a contact but not invite the contact to the meeting.

How Siebel CRM Desktop Resolves a Task Owner and Assignees of a Task

Siebel CRM Desktop does the following work:

- Resolves assignees in the To field into employees and contacts.
- Does not add the email addresses to the To field if associations to employees or contacts are made through the Employee or the Contact MVG dialog box for the activity that is linked to a shared task. Creating an association with an employee or a contact does not assign the task to this employee or contact.

How Siebel CRM Desktop Resolves Recipients of an Email

Siebel CRM Desktop does the following work:

- If the user shares an email that a user manually sent to or received from a contact, then Siebel CRM Desktop does the following:
  - Resolves the recipients and sender of the email into contacts.
  - Suggests associations for resolved contacts for the email activity.
  - Suggests a list of accounts and opportunities that are related to the resolved contacts.
  - If the user chooses an account or opportunity, then Siebel CRM Desktop associates the account or opportunity to the activity that the user creates from the email.
- If an email activity is created automatically, then Siebel CRM Desktop does the following:
  - Resolves the email recipients and sender into contacts.
  - Associates these contacts with the email activity.
  - Of these contacts, Siebel CRM Desktop creates the following associations for the first contact it encounters that contains a check mark in the Save Correspondence option:
    - Associates this contact as the primary for the activity
    - Associates the primary account for this contact with the activity
  - If the user sends a shared email, then Siebel CRM Desktop does the same processing except it does not resolve the sender as a contact.
  - If the user manually associates the shared email with a Siebel CRM record before the user sends the email, then the automail processing feature does the following work:
    - Preserves the associations that the user makes
    - Updates the email activity with contact associations that Siebel CRM Desktop resolves from the email addresses of the recipients.

- If, to associate a contact, the user uses the Contact MVG dialog box for an activity that is linked to a shared email, then Siebel CRM Desktop does not add email addresses to the To field. An association that is created to a contact does not update the recipients list for the email message.

How Siebel CRM Desktop Displays an Activity in Microsoft Outlook

Siebel CRM Desktop displays data for a Siebel CRM activity in Microsoft Outlook in the following ways:

- For a shared appointment, email, or task, Siebel CRM Desktop displays details of the related activity in the native Microsoft Outlook form. For example, the native Microsoft Outlook appointment form displays the following information:
  - The description of the Siebel CRM activity in the Subject field of the native Outlook Calendar item
  - The account that is associated with this Siebel CRM activity in the Account field in the extended area of the form

  For example, to review and change the account, opportunity, contacts, and employees for the Siebel CRM activity that is related to the shared item, the user can use the native Microsoft Outlook form.
- As a list of Siebel CRM activity records that are related to a parent sales record. For example, a list of activities that are related to an account or opportunity.

By default, Siebel CRM Desktop does not display a folder in the user mailbox for an activity. To make this folder visible, you can configure the metadata. For more information, see "Type Tag of the siebel_basic_mapping.xml File" on page 247.

How Siebel CRM Desktop Sets the Primary Employee of an Activity

This topic describes how Siebel CRM Desktop sets the primary employee of an activity for an appointment or a task.

How Siebel CRM Desktop Sets the Primary Employee of an Appointment

Siebel CRM Desktop sets the Primary Owned By field of an appointment according to the following priority:

1. Resolves the email address of the native Microsoft Outlook appointment to a Siebel CRM employee. If Siebel CRM Desktop finds an employee record that contains this address, then it sets the meeting organizer of the Outlook Calendar event as the primary.

2 If Siebel CRM Desktop does not find an employee that contains this address, then it compares this address with addresses from email accounts in the Microsoft Outlook profile. If it finds a match, then it returns the employee from the employee object. This situation can occur if the email address that is set for the current employee is not the same as the account address in the native Microsoft Outlook record for this employee.

3 If Siebel CRM Desktop does not find a match among the email accounts in the Microsoft Outlook profile, then no employee is found. In this situation, Siebel CRM Desktop sets the primary employee to the value in the Generic Siebel Owner system preference. For more information, see "How Siebel CRM Assigns the Meeting Organizer" on page 42.

For more information, see "Administering an Appointment That a Non-Siebel User Creates" on page 133.

How Siebel CRM Assigns the Meeting Organizer

A *Siebel user* is a user who is registered to use Siebel CRM Desktop or a Siebel application, such as Siebel Call Center. The *meeting organizer* is the user who creates the meeting. If a user creates a meeting, then Siebel CRM does the following:

- If the meeting organizer is a Siebel user, then Siebel CRM sets the value in the Owner field of the activity to the following value:

  Meeting Organizer

- If the meeting organizer is not a Siebel user, then Siebel CRM sets the value in the Owner field of the activity to the value that you specify in the Generic Siebel Owner system preference. For more information, see "Administering an Appointment That a Non-Siebel User Creates" on page 133.

How Siebel CRM Assigns the Meeting Organizer if This Organizer is Not a Siebel User

Siebel CRM requires the following:

- Every activity must include an owner.
- A Siebel employee record must exist for this owner.

Assume a Siebel user creates an appointment in Microsoft Outlook. In this situation, an employee record exists for this user, so Siebel CRM Desktop sets this user as the owner, and then synchronizes this appointment to the Siebel Server.

An employee record does not exist in the following situations:

- Assume employee A in your organization is not a Siebel user. This employee creates a meeting and then invites another employee in your organization who is a Siebel user to this meeting. A Siebel employee record does not exist for Employee A, and this employee cannot own a Siebel CRM record.
- A contact who is external to your company creates a meeting. A Siebel contact cannot own a meeting.

To create the meeting in this situation, Siebel CRM must first determine the owner for this activity. To avoid duplication errors and access conflicts between users for this meeting, Siebel CRM does the following:

1. Creates a meeting.
2. Assigns the value that you specify in the Generic Siebel Owner system preference as the owner of this meeting.

How Siebel CRM Desktop Sets the Primary Employee of a Task

Siebel CRM Desktop sets the primary employee for a task according to the following logic:

- If a user creates a shared task that is shared only with the user, then Siebel CRM Desktop resolves the user as the owner of the Siebel CRM activity.
- If a user creates a task that is shared and delegated, and if the user keeps a copy of the tasks in the user mailbox, then Siebel CRM Desktop creates an activity and sets the owner according to the following rules:
  - If the user delegates the task only to another employee, then Siebel CRM Desktop does the following work:
    - Creates an activity in Microsoft Outlook for the user
    - Sets this employee as an association to the employees team
    - Sets the first employee in the To line as the owner
  - If the user delegates the task of a shared contact to a mixture of shared, unshared, or native Microsoft Outlook contacts, then Siebel CRM Desktop does the following:
    - Creates the activity
    - Sets the user as the owner
    - Associates all shared contacts that it resolved from email addresses in the task To line. It makes these associations in the Contacts list.
  - If the user delegates the task to shared contacts and employees, then Siebel CRM Desktop does the following work:
    - Sets the first employee in the To line as the owner
    - Associates the creator with the Employee team
    - Associates contacts with the Contacts list
    - Does not create associations with other employees

    This configuration helps to avoid having two similar activities for the same employee:
    - For each assigned employee who accepts the task, Siebel CRM Desktop creates an activity with this employee, sets the owner, and associates the task creator to the Employees team. Siebel CRM Desktop does not create any other associations.
    - The activity that Siebel CRM Desktop creates in Microsoft Outlook for the first employee in the task To line is the same as the activity that it creates in Microsoft Outlook for the task creator.
  - If the user delegates the task to an external contact, then Siebel CRM Desktop creates an activity and sets the creator as the owner.

- If a user receives and shares a task, then Siebel CRM Desktop creates the activity, sets the employee who received the task as the owner, and adds the employee who sent the task to the Employees team as a nonprimary member.

How Siebel CRM Desktop Handles an Attachment

To perform upload, download and delete operations on attachments, Siebel CRM Desktop uses the EAI Siebel Adapter business service. Siebel CRM Desktop does this work in a way that is similar to that of a normal query, update, add, or delete operation.

How Siebel CRM Desktop Handles a Shared Activity

Microsoft Outlook supports the concept where more than one user can be associated with the same meeting or task. In this situation, Siebel CRM Desktop prevents the creation of duplicate Siebel activities. It makes sure that it associates only one Siebel CRM activity with a single Microsoft Outlook item that more than one user synchronizes. The first user who synchronizes the Microsoft Outlook item creates the Siebel CRM activity on the Siebel Server. Siebel CRM Desktop links Microsoft Outlook items with the Siebel CRM activity for any subsequent user who synchronizes.

To prevent duplicate records, Siebel CRM Desktop does the following:

1. Uses the unique identifier for the meeting and task that Microsoft Outlook provides.
2. Enters information in the CRMD Integration Id field on the Siebel CRM activity with the unique identifier from Step 1.
3. During synchronization, Siebel CRM Desktop validates that no other Siebel CRM activity includes this same unique identifier.
4. If Siebel CRM Desktop finds that there is no other activity, then it creates a new activity.
5. If Siebel CRM Desktop finds that there is another activity, then it downloads the existing activity with the same unique identifier, and then links it with the Microsoft Outlook item.

For more information, see "How Siebel CRM Desktop Prevents Duplicate Records" on page 73.

How the Origin of an Activity Affects Handling

This topic describes how the origin of an activity affects handling.

How the Origin of an Activity Affects Handling if the Item Originates in Siebel CRM

In this situation, Siebel CRM creates the activity on the Siebel Server. When Siebel CRM Desktop downloads this activity from the Siebel Server, it creates a native Microsoft Outlook item if the current user is the owner of the task, or if the current user is in the meeting participant list. Siebel CRM Desktop creates the activity as a simple appointment. No additional handling occurs in Microsoft Outlook.

How the Origin of an Activity Affects Handling if the Item Originates in Microsoft Outlook

In this situation, the activity is created in Microsoft Outlook, uploaded to the Siebel Server during synchronization, and then downloaded to another user during an incremental synchronization. When Siebel CRM Desktop downloads this activity from the server, it does not add an item in the Microsoft Outlook calendar. Instead, Siebel CRM Desktop expects Microsoft Outlook to create the necessary item in the user mailbox. To create the item, Microsoft Outlook runs the native process it uses to send a meeting invitation or to assign a task.

If Siebel CRM Desktop does this work before it synchronizes the Siebel CRM activity with the Siebel Server, then Siebel CRM Desktop links the Siebel CRM activity with the meeting or task, and then displays the item in shared mode. In shared mode, the share bar is active and any related details of the Siebel CRM activity display in the extended area of the native Microsoft Outlook form. If the user shares the item, then the item might not include all the details that the meeting organizer or task owner specified. These details only arrive after Siebel CRM Desktop synchronizes the Siebel CRM activity from the Siebel Server.

Example of How Siebel CRM Desktop Handles a Microsoft Outlook Meeting That Contains Multiple Attendees

Figure 7 illustrates how user 1, who is a meeting organizer, creates a native calendar item in Microsoft Outlook and shares it. User 2, the invitee, accepts the invitation and also shares it.

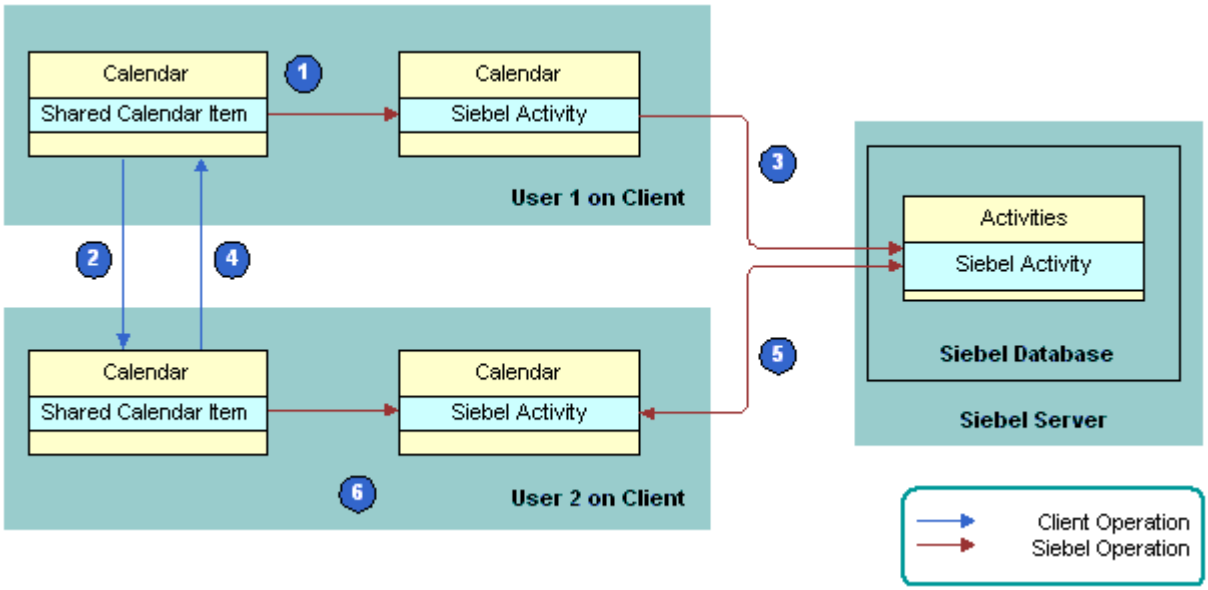


Figure 7. How Siebel CRM Desktop Handles a Microsoft Outlook Meeting That Contains Multiple Attendees

The following process is performed:

1. User 1, the meeting organizer, creates a shared meeting with user 2, a meeting attendee who is an employee.
2. User 1 saves the meeting and sends an invitation to user 2. The global ID is the same for the organizer and other meeting attendees.
3. User 1 synchronizes, and then Siebel CRM Desktop creates the activity on the Siebel Server.
4. User 2 receives and accepts the invitation.
5. User 2 shares the meeting, saves it, and then synchronizes. Siebel CRM Desktop determines user 1 already synchronized this Microsoft Outlook meeting with the Siebel Server because these items use the same global ID and include the same meeting organizer. In this situation, Siebel CRM Desktop identifies a duplicate during synchronization.
6. Siebel CRM Desktop detects that the activities are equivalent, and then proceeds to perform deduplication without displaying the collision dialog box.

Example of How Siebel CRM Desktop Handles a Shared Microsoft Outlook Appointment That Is Declined

This topic describes an example of how Siebel CRM Desktop handles a Microsoft Outlook appointment that the meeting organizer shares and that the meeting attendee declines. In this situation, the user receives an invitation from another user through Microsoft Outlook, and then declines the invitation:

1. User 1, the meeting organizer, creates an appointment, and then shares it and invites user 2, a meeting attendee.
2. User 1 sends an invitation to user 2.
3. User 1 synchronizes. Siebel CRM Desktop uploads the activity to the Siebel Server.
4. User 2 receives the meeting invitation. Siebel CRM Desktop creates a shared meeting in the Microsoft Outlook client of User 2. Siebel CRM Desktop shares the meeting because the preference for User 2 is to create new native Microsoft Outlook items as shared. Siebel CRM Desktop creates the Siebel CRM activity with User 2 in the employee team.
5. User 2 declines the meeting, with the decline set to send notification to the organizer.
6. Siebel CRM Desktop deletes the appointment from the calendar for user 2.
7. If user 2 declines the shared meeting, and if the activity is not synchronized with the Siebel Server, then Siebel CRM Desktop deletes the activity in Microsoft Outlook for user 2. The same situation applies for any meeting participant who unshares or deletes the shared meeting request.
8. User 1 receives the decline notification, Siebel CRM Desktop updates Microsoft Outlook, and removes user 2 from the employee team.
9. User 1 synchronizes. Siebel CRM Desktop synchronizes changes with the Siebel Server.
10. User 2 synchronizes.

How Siebel CRM Desktop Handles Microsoft Outlook Calendar

This topic describes how Siebel CRM Desktop handles data for data in the Microsoft Outlook calendar. It includes the following topics:

- "How Siebel CRM Desktop Handles a Microsoft Outlook Calendar Item That the User Saves, Changes or Deletes" on page 48
- "How Siebel CRM Desktop Handles a Siebel CRM Activity That the User Saves, Changes, or Deletes" on page 48
- "How Siebel CRM Desktop Handles an Appointment" on page 49
- "How Siebel CRM Desktop Handles a Recurring Appointment" on page 55

How Siebel CRM Desktop Handles a Microsoft Outlook Calendar Item That the User Saves, Changes or Deletes

The following behavior applies if the user saves, changes, or deletes an item in the Microsoft Outlook calendar:

- If the user saves a new Microsoft Outlook calendar item that is shared, then Siebel CRM Desktop creates a new Siebel CRM activity in Microsoft Outlook.
- If the user changes an appointment in the Microsoft Outlook calendar, and if the user is the owner of the activity, then Siebel CRM changes the activity.
- If the user changes an appointment in the Microsoft Outlook calendar, and if the user is not the owner of the activity, then Siebel CRM does not change the organizer appointment in Microsoft Outlook. It is not necessary to synchronize the appointment. Siebel CRM Desktop does not update the Siebel CRM activity.
- If the user deletes an appointment from the Microsoft Outlook calendar, and if the user is not the owner of the activity, then Siebel CRM Desktop removes the user from the participant list. If the activity is not synchronized with the Siebel Sever, then Siebel CRM Desktop deletes the activity in Microsoft Outlook for each participant.
- If the user deletes an appointment from the Microsoft Outlook calendar, and if the user is the owner of the activity, then Siebel CRM Desktop removes the activity.
- If the user synchronizes an All Day event from Microsoft Outlook with the Siebel Calendar, then the synchronization does the following:
  - Saves the event with a start time and end time of 12:00 A.M.
  - Uses start and end date values that the user specifies in Microsoft Outlook.

  For a single All Day event, this behavior results in a Siebel Calendar entry that includes a start time of 12:00 A.M and an end time of 12:00 A.M.

For more information, see "How Siebel CRM Desktop Maps Fields Between Siebel Activities and Outlook Calendar" on page 229.

How Siebel CRM Desktop Handles a Siebel CRM Activity That the User Saves, Changes, or Deletes

Microsoft Outlook internally associates a Siebel CRM activity with an appointment or task. Siebel CRM Desktop applies the following logic if a user saves, changes, or deletes a Siebel CRM activity:

- If the activity does not include an associated item, then Siebel CRM Desktop attempts to find the related Microsoft Outlook item, and then associates it with the activity.
- If the activity exists in Microsoft Outlook, then Siebel CRM Desktop links it to the corresponding Microsoft Outlook item.

- If the activity originates as Siebel CRM data, and if Siebel CRM Desktop cannot find a correlation, then it creates a new Microsoft Outlook item and associates it with the activity. The type of Microsoft Outlook item that Siebel CRM Desktop creates depends on the following value in the Display In field of the activity:
  - If the value in the Display In field is Calendar and Activities, then Siebel CRM Desktop creates an appointment.
  - If the value in the Display In field is To Do and Activities, then Siebel CRM Desktop creates a task.

The mapping that Siebel CRM Desktop creates between the Microsoft Outlook calendar item and the first Siebel CRM activity is the same as that described in "How Siebel CRM Desktop Maps Fields Between Siebel Activities and Outlook Calendar" on page 229, with Siebel CRM Desktop performing the following additions:

- Sets the value in the Show Time As field of the native Microsoft Outlook appointment to Busy
- Sets the appointment label to None

How Siebel CRM Desktop Handles an Appointment

This topic describes how Siebel CRM Desktop handles an appointment.

How Siebel CRM Desktop Correlates a Siebel CRM Activity with PIM Data in Microsoft Outlook

When Siebel CRM Desktop synchronizes a Siebel CRM activity to Microsoft Outlook, it attempts to find the PIM data that resides in Microsoft Outlook that corresponds the activity. PIM data is a calendar event, appointment, task or email. If it finds this item, then it shares it and correlates it with the Siebel CRM activity. Siebel CRM Desktop performs this correlation for the following items:

- An Outlook calendar item for a Siebel CRM record where the Display In value of the Siebel CRM activity is Calendar and Activities.
- An Outlook task item where the Display In value of the Siebel CRM activity is To Do and Activities.
- Each Outlook activity where the Display In value of the Siebel CRM activity is Activities Only.

For example, to perform a correlation for an Outlook calendar event, it uses the following keys:

1. Key 1:
   a. The CRMD Integration Id equals the GlobalObjectId of the calendar event.
   b. The owner is the meeting organizer of the calendar event.
2. Key 2:
   a. The owner is the meeting organizer of the calendar event.
   b. The description is the subject of the calendar event subject.
   c. The value in the Planned field in the Siebel database equals the start time of the calendar event. This Planned value maps to the value the Start field in Microsoft Outlook.

If the user creates an activity in Siebel CRM Desktop from a Microsoft Outlook calendar event, and if the user shares this activity with Siebel CRM, then the CRMD Integration Id field in the activity record in the Siebel database contains a value.

How Siebel CRM Desktop Correlates Data If Siebel CRM Desktop Is Installed

The following sequence describes how Siebel CRM Desktop uses key 1:

1. User 1 does the following work:
   a. Creates a meeting in Microsoft Outlook
   b. Shares this meeting with Siebel CRM
   c. Sends the meeting request to User 2

   In this situation, Siebel CRM Desktop creates a Siebel CRM activity. To populate the value in the CRMD Integration Id field in this activity record, it uses the value from the GlobalObjectId field of the meeting. This value is a unique for this meeting. Siebel CRM Desktop uses the corresponding values in the Outlook meeting to identify the meeting organizer and the meeting participants.
2. User 1 synchronizes this activity and Siebel CRM adds it to the Siebel database.
3. Assume user 2 sets the user preference to not automatically create a PIM item that Siebel CRM Desktop shares with Siebel CRM. If user 2 receives the meeting invitation from user 1, then Siebel CRM Desktop does not share the meeting for this user in the calendar and does not create a Siebel CRM activity.
4. When User 2 synchronizes, Siebel CRM Desktop synchronizes the activity that it added in Step 2 to Microsoft Outlook. It uses the find_ol_item function to find the Microsoft Outlook item that corresponds to this activity. It finds the unshared meeting because the following situations are true:
   - This meeting contains the same GlobalObjectId field that the CRMD Integration Id field of the Siebel CRM activity contains
   - This meeting contains the same meeting organizer that the Activity Owner field of the Siebel CRM activity contains

If the meeting attendee synchronized the activity from the Siebel Server before this attendee receives an invitation, and if this attendee sets the preference in the Options dialog box to not share new PIM items, then Siebel CRM Desktop uses the find_proxy_item function to find the Siebel CRM activity. If Siebel CRM Desktop finds this activity, then it shares the meeting with Siebel CRM.

How Siebel CRM Desktop Correlates Data If Siebel CRM Desktop Is Not Installed

Assume the following situation is true:

- To track activities, a user uses a Siebel CRM application, such as Siebel Call Center, and Microsoft Outlook.
- This user has not installed Siebel CRM Desktop.
- This user enters activities in Siebel CRM and Microsoft Outlook.

- The user has an activity in Siebel CRM. The user also has an appointment in the Microsoft Outlook calendar that matches this activity. This activity and this appointment each include the same subject, start date, and activity owner.
- Assume this user installs Siebel CRM Desktop and then synchronizes.

In this situation, Siebel CRM Desktop cannot use Key 1 because the Siebel CRM activity does not include a value in the CRMD Integration Id field. If this field does not contain a value, then Siebel CRM Desktop uses key 2. The following sequence describes how Siebel CRM Desktop uses key 2:

1. Because the CRMD Integration Id field is empty, it skips key 1.
2. Correlates the activity:
   a. If the activity is a task, then it uses Key 2 with the following differences:
      - The owner is the meeting organizer of the task.
      - The description is the subject of the task subject.

      For more information, see Step 2 on page 49.

   b. If the activity is an email message, then it uses the following keys:
   - Key 2.2:
     - CRMD Integration Id is the first twenty-two bytes of the PR_CONVERSATION_INDEX email message
     - Owner is the current user
   - Key 2.3
     - CRMD Integration Id is the first twenty-two bytes of the PR_CONVERSATION_INDEX email message
     - Owner is the current user
     - Planned is the date the email is sent or received. The value in the Planned field in the Siebel database equals the start time of the calendar event. This Planned value maps to the value the Start field in Microsoft Outlook.

Note the following:

- If the Display In value of the Siebel CRM activity is To Do and Activities, then the activity is a task.
- If the Display In value of the Siebel CRM activity is Activities Only, then the activity is an email message.

How Siebel CRM Desktop Uses Natural Keys to Identify a Duplicate Activity

Siebel CRM Desktop uses natural keys to detect a duplicate between Microsoft Outlook data and Siebel CRM data. Siebel CRM Desktop defines the following natural keys for an activity:

- Activity Owner and CRMD Integration Id, which matches the GlobalId of an appointment
- Activity Owner and Description, which matches the subject of the appointment and Start Date

Siebel CRM Desktop uses these keys to query the Siebel database. This query determines if a duplicate exists for this activity in the Siebel database. The following is an example of the natural keys that Siebel CRM Desktop might define in the connector_configuration.xml file:

```
<natural_keys>
  <natural_key>
     <field>CRMD Integration Id</field>
     <field>Primary Owner Id</field>
  </natural_key>
  <natural_key>
     <field>Description</field>
     <field>Planned</field>
     <field>Primary Owner Id</field>
  </natural_key>
</natural_keys>
```

For more information, see "About Files in the Customization Package" on page 144.

How Siebel CRM Desktop Handles Repeat Appointments

Microsoft Outlook includes more recurrence patterns than the Siebel calendar. Siebel CRM Desktop establishes a correlation between Microsoft Outlook and a Siebel CRM recurrent pattern in the following way:

- If the Microsoft Outlook pattern matches an existing Siebel CRM pattern, then Siebel CRM Desktop uses the corresponding recurrence pattern to create a Siebel CRM activity.
- If the Microsoft Outlook pattern does not match an existing Siebel CRM pattern, then Siebel CRM Desktop uses a Siebel CRM pattern that occurs more frequently. It also uses more exceptions for an excess occurrence.

For example, assume the user creates a meeting in Microsoft Outlook that occurs every two weeks for two months for a total of four meeting instances. If Siebel CRM Desktop attempts to synchronize this meeting with the Siebel Server, then it cannot directly support the recurrence patterns that are available in Siebel CRM. Instead, it does the following work:

- Creates a weekly meeting that lasts for two months for a total of eight meeting instances.
- Creates four exceptions that cancel the intervening weeks.

To remain compatible with Siebel CRM data, Siebel CRM Desktop represents each occurrence that has changed as a separate appointment in Microsoft Outlook data.

To handle a repeating appointment, Siebel CRM Desktop does the following work:

- Maps a recurrent Microsoft Outlook appointment to a recurrent Siebel appointment

- Maps a recurrent Siebel appointment to a recurrent Microsoft Outlook appointment

Table 4 describes the Siebel fields that Siebel CRM Desktop uses to create a repeating appointment.

Table 4. Siebel Fields That Siebel CRM Desktop Uses to Create a Repeating Appointment

Siebel Field	Description
RepeatingType	The frequency of the appointment.
RepeatingExpires	The date of occurrence of the last instance in the series.
Repeating	The flag that indicates an appointment is repeating.

How Siebel CRM Desktop Handles a Single Instance of a Repeating Appointment

Siebel CRM Desktop handles an exception to a recurrent appointment as separate records in Siebel CRM data. For example, to change the time of an instance of a recurrent meeting, Siebel CRM Desktop creates a separate appointment. Siebel CRM Desktop follows standard handling practices for the Siebel calendar so that it handles the appointment series and exceptions in Siebel CRM appropriately.

About Invitee List Handling for an Appointment

The list of invitees in an appointment can contain contacts and employees. To process the email addresses that are specified in the list, Siebel CRM Desktop intercepts the call to the EAI Siebel Adapter business service. If the user creates a shared appointment in Microsoft Outlook, then the client attempts to resolve the meeting attendees and categorize the attendees as related contacts or employees when the user saves the appointment item.

Because the user might not possess all the contact or employee data that Siebel CRM Desktop requires to parse all attendees, Siebel CRM Desktop repeats this process during synchronization. If Siebel CRM Desktop makes an insert or update request for an appointment record, then the Siebel Server validates the meeting attendees. If the server detects any changes in the attendee list, then the server returns the updated list to the client and the server updates the contact and employee lists that are associated with the appointment.

How Siebel CRM Desktop Handles an Invitee List for the Update Operation

To handle an invitee list for the update operation, Siebel CRM Desktop does the following work:

1. Updates the input to the EAI Siebel Adapter business service to reflect changes in the invitee list. This update occurs when Siebel CRM Desktop passes the List of Invitees in the Siebel message in the Email To Line field.
2. Marks the contacts and employees that Siebel CRM Desktop removes from the updated appointment in Microsoft Outlook, and then updates these records in the message.
3. Marks and updates the contacts and employees that Siebel CRM Desktop newly added as insert records, and then updates these records in the message.

How Siebel CRM Desktop Handles an Invitee List for the Query Operation

To handle an invitee list for the query operation, Siebel CRM Desktop does the following work:

1. To return the invitee list in the To line of the email, it queries the appointment. To handle this query, Siebel CRM Desktop processes the output Siebel messages that the EAI Siebel Adapter business service returns.
2. Processes the output from a call to the Query method or the QueryByTemplate method of the EAI Siebel Adapter business service.
3. If the returned record is an activity record with the type as Appointment, then Siebel CRM Desktop modifies it in order to add the email addresses of the employees and contacts in the activity to the email To line. Siebel CRM Desktop uses a semicolon to separate each email address. Siebel CRM Desktop retains the employee list and the contact list.

How Siebel CRM Desktop Handles a Recurring Appointment

A *recurring appointment* is an appointment that repeats at a specified interval. Siebel CRM Desktop supports synchronizing a recurrent appointment between Microsoft Outlook and the Siebel Server, but Siebel CRM Desktop handles a recurrence pattern differently for a Microsoft Outlook appointment than it does for a Siebel calendar activity.

Figure 8 illustrates how Siebel CRM Desktop handles a recurrent appointment.

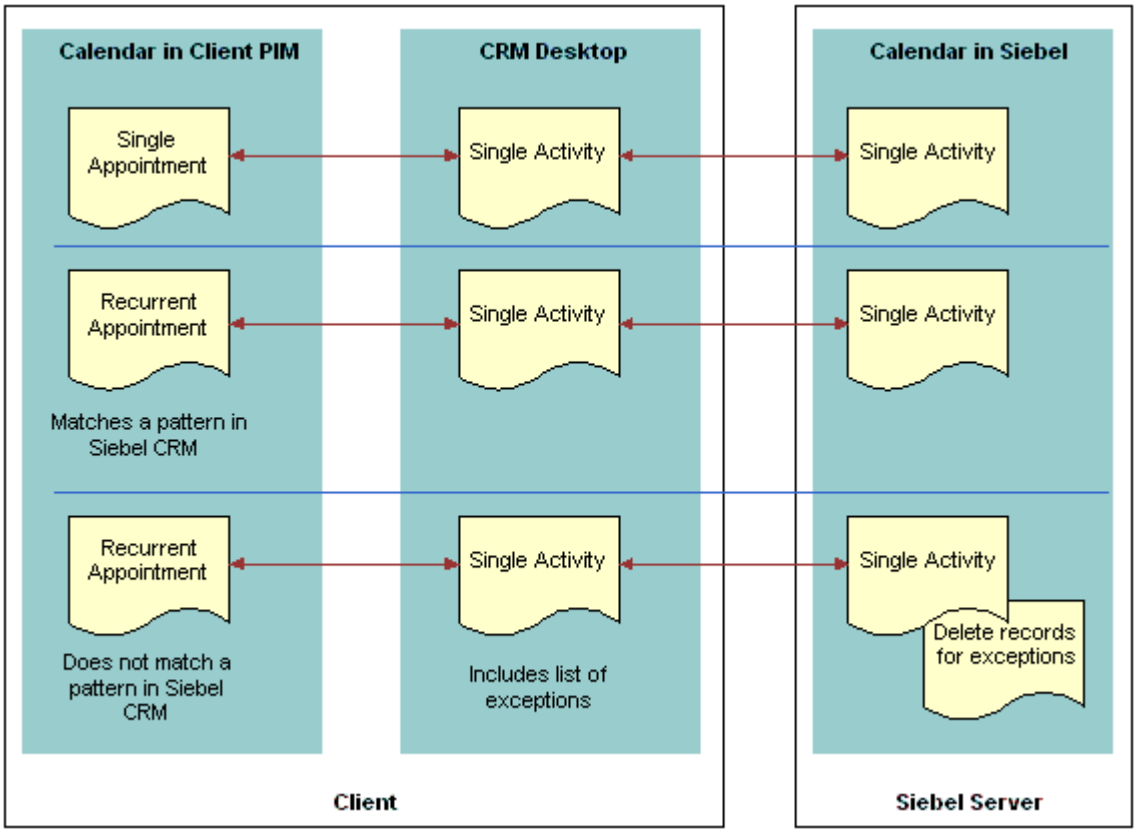


Figure 8. How Siebel CRM Desktop Handles a Recurrent Appointment

For more information, see "How Siebel CRM Desktop Maps Fields Between Siebel CRM Data and Microsoft Outlook Data" on page 229.

How Siebel CRM Desktop Handles a Microsoft Outlook Task

This topic describes how Siebel CRM Desktop handles data for a native Microsoft Outlook task. For more information, see "How Siebel CRM Desktop Maps Fields Between Siebel CRM Data and Microsoft Outlook Data" on page 229.

If the user changes or saves a native Microsoft Outlook task, then Siebel CRM Desktop does the following work:

- If the user saves a new Microsoft Outlook task that is shared, and if the record with the same global ID does not exist in the Siebel activities, then Siebel CRM Desktop creates a new Siebel CRM activity.
- If the user changes a native Microsoft Outlook task, then Siebel CRM Desktop changes the corresponding Siebel CRM activity.
- If the user deletes a native Microsoft Outlook task, and if the user:
  - Is the owner of the activity, then Siebel CRM Desktop deletes the corresponding Siebel CRM activity.
  - Is not the owner of the activity, then Siebel CRM Desktop removes the user from the employee team. Siebel CRM Desktop does not delete the corresponding Siebel CRM activity.

How Siebel CRM Desktop Handles a Microsoft Outlook Email Message

Siebel CRM Desktop handles a Microsoft Outlook email message in the following ways:

- Saves the email message as a Siebel CRM activity and sets the activity type to Email - Inbound or Email - Outbound. The customization package specifies the type of activity.
- Sets the Display In value to Communications and Activities.
- Creates one Siebel CRM activity for each Microsoft Outlook email message that is created.
- Allows the user to link the Siebel CRM activity to a Siebel CRM record. For more information, see "How a User Can Link a Siebel CRM Record to a Microsoft Outlook Record" on page 58.
- Saves the first email message as an .msg attachment along with the activity that is created.

If the user deletes the source email message or moves it to a new folder, then Siebel CRM Desktop does not change the activity. Deleting or modifying the activity does not affect the source email.

For more information, see "How Siebel CRM Desktop Maps Fields Between Siebel CRM Data and Microsoft Outlook Data" on page 229.

How Siebel CRM Desktop Handles Siebel CRM Data with Automatic Email Processing

The user can choose the Save Correspondence option for a shared contact with Siebel CRM data. When an email message is received, Siebel CRM Desktop automatically checks the recipients list. If Siebel CRM Desktop finds an email address that matches one or more contacts with the Save correspondence option chosen, then Siebel CRM Desktop automatically creates the corresponding Siebel CRM activity that is then associated with these contacts.

How Siebel CRM Desktop Handles Siebel CRM Data with Manual Email Processing

The user starts manual email processing from the email form. For more information, see "How a User Can Link a Siebel CRM Record to a Microsoft Outlook Record" on page 58.

How Siebel CRM Desktop Handles Items If the User Removes the Siebel CRM Desktop Add-In

If the user removes the Siebel CRM Desktop add-in, then Siebel CRM Desktop completely removes all Siebel CRM data. How Siebel CRM Desktop handles a shared Microsoft Outlook item when the user removes the Siebel CRM Desktop add-in depends on if the item is Microsoft Outlook data or Siebel CRM data, and on the type of object. Siebel CRM Desktop handles objects in the following ways:

- **Shared appointments.** Siebel CRM Desktop removes appointments that originate in Siebel CRM from the Microsoft Outlook calendar. For a Microsoft Outlook appointment, it removes any Siebel activities that are related to the Microsoft Outlook appointment. The appointment no longer displays as shared and no contextual Siebel CRM data is related to the appointment.
- **Shared contacts.** Because Siebel CRM Desktop cannot determine if a contact is Microsoft Outlook data or Siebel CRM data, it removes all shared contacts from the user mailbox. It is recommended that you backup or unshare contacts that the user must preserve before removing Siebel CRM Desktop.
- **Shared emails.** Siebel CRM Desktop removes Siebel activities that are associated with shared emails so they no longer display as shared in Microsoft Outlook, and so that Microsoft Outlook does not display any contextual data.
- **Shared tasks.** Siebel CRM Desktop handles tasks in the same way that it handles appointments. Siebel CRM Desktop removes tasks that originate in Siebel CRM from Microsoft Outlook. Siebel CRM Desktop does not remove a native Microsoft Outlook task. Microsoft Outlook does not display the task as a shared task, and it does not display any Siebel CRM data that is related to the task.

An Unshared Item Is Not Affected If the User Removes Siebel CRM Desktop

If the user removes Siebel CRM Desktop, then an unshared item is not affected. If the user shares an item in Microsoft Outlook, unshares it, and then synchronizes with the Siebel Server before the user removes Siebel CRM Desktop, then the item is not shared. This item is not affected if the user subsequently removes Siebel CRM Desktop. This situation occurs because Siebel CRM Desktop only deletes Siebel CRM data and extensions to Microsoft Outlook that you deploy through Siebel CRM Desktop.

How a User Can Link a Siebel CRM Record to a Microsoft Outlook Record

Siebel CRM Desktop allows the user to change linked values. For example, to choose Siebel CRM records to link with the email, the user can use the email form, and then perform the following work:

- Use the autocomplete feature when the user types in the field.
- Use the autosuggest feature when the user clicks Contact, Account, or Opportunity on the Extension Bar of the email form.
- Choose an item from a Siebel control on any shared Microsoft Outlook item:
  - The Siebel control calls the appropriate dialog box that allows the user to choose one or more records.
  - The dialog box supports creating a new record so long as the permissions on the source dialog box allow that operation.

In another example, Siebel CRM Desktop allows the user to link a Siebel CRM activity to one of the following Siebel CRM records:

- One account
- One opportunity
- Multiple contacts

5 How Siebel CRM Desktop Synchronizes Data

This chapter describes how Siebel CRM Desktop synchronizes data. It includes the following topics:

- How Siebel CRM Desktop Synchronizes Data Between the Client and the Siebel Server on page 59
- Factors That Determine Which Data Siebel CRM Desktop Synchronizes on page 66
- How Siebel CRM Desktop Handles a Conflict During Synchronization on page 71

For more information, see:

- "Overview of How Siebel CRM Desktop Synchronizes Data" on page 22.
- "Regulating the Number of Records That Siebel CRM Desktop Synchronizes" on page 131.

How Siebel CRM Desktop Synchronizes Data Between the Client and the Siebel Server

This topic describes how Siebel CRM Desktop synchronizes data between the client and the Siebel Server. It includes the following topics:

- "How Siebel CRM Desktop Synchronizes Data During the Initial Synchronization" on page 59
- "How Siebel CRM Desktop Synchronizes Data During an Incremental Synchronization" on page 60
- "How Siebel CRM Desktop Synchronizes Siebel CRM Data" on page 62
- "How Siebel CRM Desktop Manages Synchronization Duration" on page 63
- "Situations Where Siebel CRM Desktop Reinstalls the Data Structure" on page 64

How Siebel CRM Desktop Synchronizes Data During the Initial Synchronization

An *initial synchronization* is a type of synchronization that occurs in the following situations:

- Immediately after the user installs Siebel CRM Desktop.
- If you deploy a metadata change to the user that includes a change to the data schema.
- If the options in the login dialog box change. For example, the user name changes or the URL of the Siebel Server changes.

The purpose of the initial synchronization is to initialize the Microsoft Outlook data storage with the Siebel CRM data that is available to the user. Siebel CRM Desktop downloads files in the customization package to Microsoft Outlook the first time the user synchronizes metadata with the Siebel Server. This metadata includes the following information:

- Definition data for Siebel CRM Desktop, such as synchronization rules, object definitions, and so forth
- Siebel CRM data for Siebel CRM Desktop, such as accounts, opportunities, and so forth

Siebel CRM Desktop does the following work during the initial synchronization:

1. Establishes a synchronization session with the Siebel Server through the Web service interface.
2. Directs the Web Service Connector in Microsoft Outlook to call the DownloadMetadataFiles method of the PIM Client Metadata Service business service that resides on the Siebel Server.

   To broker requests through the EAI Siebel Adapter, Siebel CRM Desktop uses the business services that the Web services references. It uses the EAI Siebel Adapter business service to process the SiebelMessage payload in the requests.
3. Directs the PIM Client Metadata Service on the Siebel Server to obtain the login ID of the user from the session, and then determines what responsibility is associated with the user, what customization package is associated with the responsibility, and if the package is active. For more information, see "Relationships Between Users, Responsibilities, Customization Packages, and Metadata Files" on page 30.
4. If the customization package is published and valid, then Siebel CRM Desktop queries all metadata files of that package and constructs a Siebel message. Siebel CRM Desktop does the following work:
   - Sets the containsFiles argument to true.
   - Enters the relevant data in the packageId, responsibilityId, and hashValue arguments.
5. Applies the downloaded package for Microsoft Outlook.
6. Downloads Siebel CRM data.

   For more information, see "How Siebel CRM Desktop Synchronizes Siebel CRM Data" on page 62.
7. Logs out of the synchronization session that it established in Step 1.

For more information, see "How Siebel CRM Desktop Handles Errors During Synchronization" on page 133.

How Siebel CRM Desktop Synchronizes Data During an Incremental Synchronization

An *incremental synchronization* is a synchronization session that occurs any time after the initial synchronization. To determine the differences that exist in the data that is available to the user, Siebel CRM Desktop compares data in the Microsoft Outlook data storage to data in the Siebel database. Siebel CRM Desktop then does the following work:

- Inserts, updates, or deletes data on the Siebel Server according to changes that occurred in Microsoft Outlook since the prior synchronization
- Inserts, updates, or deletes data in Microsoft Outlook according to changes that occurred on the Siebel Server since the prior synchronization

Siebel CRM Desktop does this work for each difference until all data in the Microsoft Outlook data storage is synchronized with data in the Siebel database. In all situations, the user works with data locally in Microsoft Outlook and Siebel CRM Desktop sends those changes to the Siebel Server during an incremental synchronization, but not at the same time that it makes the change in Microsoft Outlook. Depending on the frequency of the process, a change might not appear on the server immediately.

To complete an incremental synchronization, Siebel CRM Desktop does the following work:

1. Connects to the Siebel Server to establish a synchronization session.
2. Authenticates the user.
3. Passes the values of the packageId and responsibilityId arguments that it caches in Microsoft Outlook to the Siebel Server. Siebel CRM Desktop cached these values during the prior synchronization. This is done to avoid expensive iterative operations through all responsibilities and customization packages every time Siebel CRM Desktop calls the Web service.
4. Receives a reply from the Siebel Server. This reply indicates whether new metadata is available for the user.
5. If new metadata is available, then it does the following work:
   - Determines if the customization package changed.
   - If the package changed, then it downloads the new package to a temporary folder in Microsoft Outlook.
   - If the package is not changed, then it proceeds to Step 9.
6. If the downloaded package is not compatible with the current version of Siebel CRM Desktop, then Siebel CRM Desktop does the following:
   a. Determines if the currently applied package is compatible with the downloaded package, and then does one of the following:
      - If it is compatible, then Siebel CRM Desktop synchronizes the current data.
      - If it is not compatible, then Siebel CRM Desktop stops the synchronization.
   b. Displays a product incompatibility error message, and does not apply the downloaded package.
   c. Exits this process.
7. If the downloaded package is compatible with the current version of Siebel CRM Desktop, then Siebel CRM Desktop does the following work:
   a. Synchronizes the data to the Siebel Server.
   b. Determines if the data schema in the downloaded package changed, and then does the following:
      - If the data schema has not changed, then it applies the new package.

- If the data schema has changed, then it prompts the user to remove the current customization and the current data. If the user agrees, then it removes the old customization package and the Siebel data, and then applies the new package. If the user does not agree, then it does not apply the new package.

8 If the customization packages are identical, then Siebel CRM Desktop does the following work:

   a Sends a reply which indicates that it is not necessary to download the customization package because the package that is on the Siebel Server is the same as the package that is in Microsoft Outlook.

   b Exits this process.

9 Downloads Siebel CRM data.

   For more information, see "How Siebel CRM Desktop Synchronizes Siebel CRM Data" on page 62.

   The user can now view the newly downloaded data.

10 Logs out of the synchronization session that it established in Step 1.

How Siebel CRM Desktop Handles Changes to Login Credentials

If the user name or the server URL changes, then Siebel CRM Desktop reinitializes the data structure. It does this to remove any personal user data that might exist, and to provide the user with an opportunity to synchronize data. Before Siebel CRM Desktop begins the reinitialization, it displays a warning to the user that any data that is not synchronized might be lost. If the user agrees to proceed, then the following occurs:

1 Siebel CRM Desktop removes the current customization.

2 The user logs in with new credentials.

3 Siebel CRM Desktop downloads the package from the Siebel Server and then starts the First Run Assistant.

How Siebel CRM Desktop Synchronizes Siebel CRM Data

To synchronize Siebel CRM data, such as opportunities and accounts, Siebel CRM Desktop does the following work:

1 Determines the number of records of each type of record, such as opportunities or accounts.

2 For all synchronization objects that are enabled, such as opportunities or accounts, gets the values of the change keys. For example, the record ID and the last updated date in the Siebel database.

3 Compares the set of IDs and timestamps in Microsoft Outlook to the set of IDs and timestamps on the Siebel Server to do the following:

- Determine differences that exist between the data sets for inserts, updates, and deletes.
- Identify conflicts and create a log entry in the synchronization conflict list for any conflict.

4 For each difference, Siebel CRM Desktop performs one of the following operations in Microsoft Outlook or on the Siebel Server:

- **Siebel insert.** Query the Siebel database to get the details of the new record, and then insert the appropriate item in Microsoft Outlook.
- **Siebel update.** Query the Siebel database to get the details of the updated record, and then update the appropriate item in Microsoft Outlook. Note that a Siebel update overwrites all fields in the corresponding Microsoft Outlook item, not just the updated fields.
- **Siebel delete.** Delete the appropriate item in Microsoft Outlook.
- Microsoft Outlook **insert.** Use the user key that is defined in the metadata to query the Siebel database, and then do one of the following:
  - If it does not find a match, then it inserts the appropriate record in the Siebel database, and then queries the Siebel database to get the record ID and timestamp.
  - If it does find a match, then it returns a *synchronization issue,* which is an error that occurs during synchronization.
- Microsoft Outlook **update.** Use the user key that is defined in the metadata to query the Siebel database, and then do one of the following:
  - If it does not find an update for the modification number of the record, then it updates the appropriate record in the Siebel database, and then queries the Siebel database to get the record Id and updated timestamp.
  - If it does find an update for the modification number, then it returns a synchronization issue. Note this handling is different than in the situation where Siebel CRM Desktop changes the same record in the Siebel database and when it compares IDs and timestamps. In this situation, Siebel CRM Desktop makes the change in the Siebel database during the actual update operation.
- Microsoft Outlook **delete.** Delete the appropriate record in the Siebel database.

5 If a conflict occurs, then Siebel CRM Desktop does the following work:

a Updates the synchronization issues and conflicts log on the client.

b Prompts the user to choose which changes to keep in each of the following situations:

- Update the record in Microsoft Outlook and on the Siebel Server.
- Update the record in one data set and delete the record in the other data set.

6 Repeats Step 4 for each additional Siebel CRM data object that requires synchronization.

How Siebel CRM Desktop Manages Synchronization Duration

Several factors determine the duration of a synchronization, such as the amount of data that is available to the user, network bandwidth, server performance, client performance, and so forth. To shorten this duration, you or the user can do the following:

- You can modify the application configuration. For more information, see Regulating the Number of Records That Siebel CRM Desktop Synchronizes on page 131.
- The user can adjust settings through the synchronization filter dialog box. For more information, see "Filters Reduce the Number of Siebel CRM Records That Are Available in Siebel CRM Desktop" on page 67.

The duration of an incremental synchronization session is typically shorter than for an initial synchronization because Siebel CRM Desktop downloads all objects during an initial synchronization while it only downloads those objects that are changed during an incremental synchronization.

Situations Where Siebel CRM Desktop Reinstalls the Data Structure

Siebel CRM Desktop reinstalls the data structure in any of the following situations:

- The customization package has changed and this involves a change to the data schema.
- The package update for the user involves a data schema change.
- The user logs in as a different user.
- There is a problem with the data structure. For example, assume the user deletes the Opportunities folder, and then removes this deletion from the Deleted Items folder. If the user restarts Microsoft Outlook, then Siebel CRM Desktop does the following:
  - Informs the user that there is a problem with the data structure.
  - Removes the data structure.
  - Installs a new data structure.

If Siebel CRM Desktop must reinstall the data structure, then it does the following work:

1. Removes all Siebel CRM data, such as accounts, opportunities, shared contacts, and activities.
2. Removes all shared appointments and tasks that the user created for a Siebel CRM activity.
3. Removes the custom data structure that it previously deployed to Microsoft Outlook data storage. For example, to remove all custom folders in the user mailbox.
4. Installs the new data structure.
5. To reenter the appropriate Siebel CRM data in the Microsoft Outlook data storage, the user must manually start a new, initial synchronization session.

The Customization Package Has Changed

During synchronization, Siebel CRM Desktop determines if the customization package for the user who is currently logged in has changed in such a way that it must reinstall the data structure. The following changes in the data structure of the customization package can cause this situation:

- An object is added to or deleted from the mapping scheme.
- A field is added to an existing object or an existing field is modified.

If the customization package has changed, then Siebel CRM Desktop displays a prompt that is similar to the following:

```
Data structure has changed. Continue synchronization? Selecting "Yes" will remove
your current data, re-install the data structure, and download the data anew.
```

The Customization Package Has Changed but the Data Structure Has Not Changed

If, during synchronization, Siebel CRM Desktop determines that the customization package for the user who is currently logged in has changed in such a way that there is no change to the data structure, then Siebel CRM Desktop downloads and installs the new package, and informs the user about this download. The synchronization continues. A modification to a security rule is an example of where the package has changed, but the data structure has not changed. In this situation, Siebel CRM Desktop does not start a new, initial synchronization.

Situations Where Local Data Might Be Lost

To prevent losing data due to a reinstallation of the data structure, Siebel CRM Desktop must use the current customization package to upload local data to the Siebel Server. The exception to this requirement occurs if the package is changed locally. In this situation, Siebel CRM Desktop cannot use the current, flawed package to synchronize data. Data for the user can be lost in the following situations:

- If data exists that Siebel CRM Desktop cannot upload to the Siebel Server because the permissions for the user have changed.
- If data exists for a user that is associated with synchronized items but is stored in fields that are not synchronized. Siebel CRM Desktop completely replaces the record in Microsoft Outlook with the record that it downloads from the Siebel Server. It erases any local data.

How Siebel CRM Desktop Prevents Data Loss if the User Deletes Customization Package Files

If Microsoft Outlook is open, then the user cannot delete any customization package files. For example, if the user attempts to use Windows Explorer to delete files from the following directory, then Windows Explorer does not allow the deletion:

```
C:\Documents and Settings\user\Application Data\Oracle\CRM Desktop\Profile\data
```

If Microsoft Outlook is not running, then the user can use Windows Explorer to delete customization package files. However, if the user subsequently starts Microsoft Outlook, then Siebel CRM Desktop restores the customization package files from local storage. This local storage is a pst folder or an Exchange mailbox.

Affect of a Connectivity Failure

An internet or network connectivity failure that occurs during synchronization can interrupt the synchronization. An interruption does not cause data loss or corruption. Synchronization can proceed from the last step that Siebel CRM Desktop ran successfully before the interruption.

Factors That Determine Which Data Siebel CRM Desktop Synchronizes

This topic describes the factors that determine which data Siebel CRM Desktop synchronizes. It includes the following topics:

- "Factors That Determine Which Data a Siebel CRM Desktop User Can Access" on page 66
- "How Differences in Data Between Microsoft Outlook and the Siebel Server Affect Synchronization" on page 69
- "How You or the User Can Modify Synchronization Frequency" on page 70
- "How Siebel CRM Desktop Avoids Duplicate Data" on page 70

Factors That Determine Which Data a Siebel CRM Desktop User Can Access

A Siebel user can typically access only a subset of data that is available in the Siebel database. This topic describes that factors that determine which data a Siebel CRM Desktop user can access. How you configure Siebel CRM Desktop determines many aspects of which data Siebel CRM Desktop synchronizes. For example:

- Synchronization objects that are configured
- Internal filters that are applied
- View modes that are configured on each object
- Security and other configuration that exists on the Siebel Server

You define this configuration before you deploy Siebel CRM Desktop to your users. In the First Run Assistant, the user can choose presets for a predefined filter and define personal filters. The internal filters and server application metadata configuration always restricts access to some data, and the user filters apply a second layer of filtering. Siebel CRM Desktop applies these filters during initial synchronization and incremental synchronization.

Filters Reduce the Number of Siebel CRM Records That Are Available in Siebel CRM Desktop

Figure 9 illustrates how the number of Siebel CRM records that are available to the Siebel CRM Desktop Client reduces as these records encounter each set of filters.

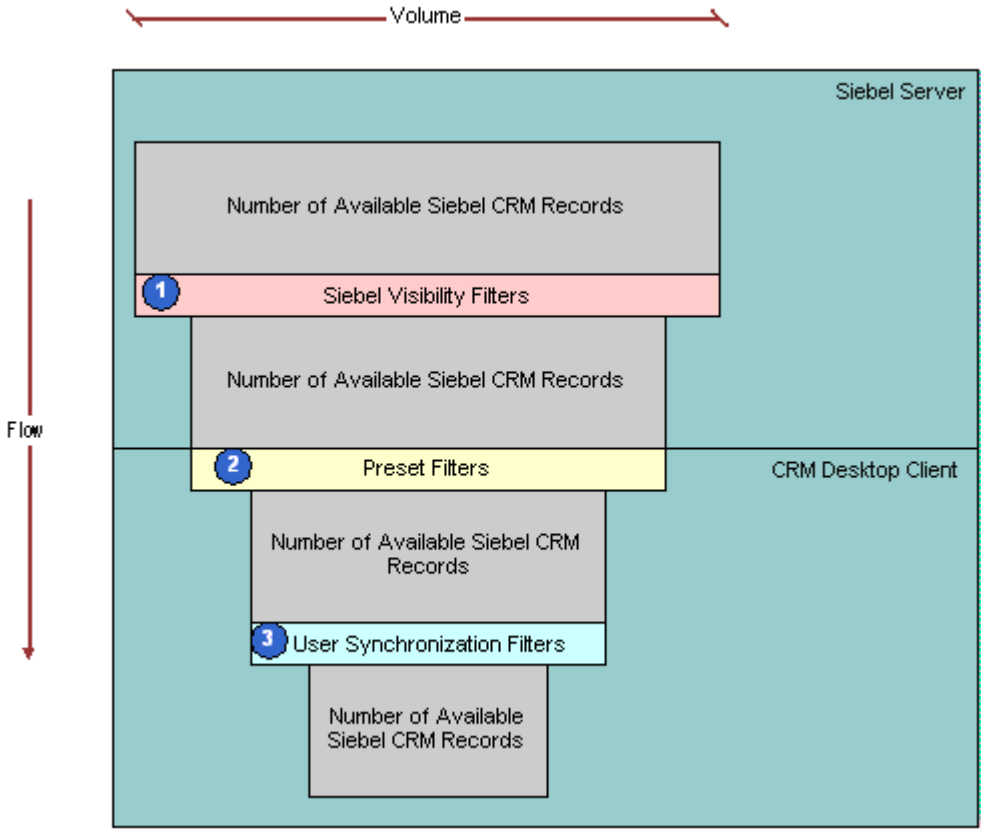


Figure 9. How Filters Reduce the Number of Siebel CRM Records That Are Available in Siebel CRM Desktop

The following filters reduce the number of Siebel CRM records that are available in Siebel CRM Desktop:

1 **Siebel visibility filters.** Visibility rules that are configured in the Siebel repository and that the Siebel Server applies affects data access. Because Siebel CRM Desktop integrates with the Siebel Server through the Web service interface, security, search specifications, and other logic that is configured at the integration or business object layer limits the data that Siebel CRM Desktop synchronizes to the client.

   The user interface configuration does not affect the results of queries or other operations that Siebel CRM Desktop performs.

2 **Preset filters.** Internal synchronization filters that are configured in the client application metadata determine which Siebel CRM data Siebel CRM Desktop synchronizes to the client. Although search specifications on the Siebel Server and security settings in the Siebel repository establish the first level of filtering, a set of filters that reside on the client can also restrict data that Siebel CRM Desktop downloads to the client.

3 **User synchronization filters.** The user configures synchronization filters when the user runs First Run Assistant. These filters affect which objects Siebel CRM Desktop enables for synchronization. They are determined by preset filters the user chooses or by filters that the user defines in the Filter Records Tab of the Synchronization Control Panel. To determine which data to synchronize, Siebel CRM Desktop uses this information in addition to the application configuration.

   Depending on relationships in the data, Siebel CRM Desktop might synchronize an object that is disabled for synchronization through the Filter Records Tab. For example, if the opportunity object is enabled but the account object is not enabled, then Siebel CRM Desktop still downloads any account data that the opportunity references. This download is required to make sure the data is complete. Also, Siebel CRM Desktop might still upload changes that the user makes in the client to the Siebel Server even if an object or synchronization filter is disabled. For example, if the user disables the account object and then creates an account in Microsoft Outlook, then Siebel CRM Desktop uploads the account to the Siebel Server. For more information, see the following topics:

   - Customizing How First Run Assistant Performs the Initial Synchronization on page 97
   - Controlling the Object Types That Siebel CRM Desktop Displays in the Filter Records Tab on page 114.
   - Controlling the Fields That Are Available in a Filter on page 115.

Objects That Are Enabled for Synchronization

Depending on the configuration that Siebel CRM Desktop downloads for the user, a set of objects that are enabled for synchronization determine the data that Siebel CRM Desktop can synchronize. These objects are defined in the application metadata that you deploy through the customization package that is available to the user. If an object is not defined in the application metadata, then Siebel CRM Desktop does not synchronize it. Application metadata also defines the field mappings that Siebel CRM Desktop uses in the synchronization. These mappings specify how Siebel CRM Desktop synchronizes objects in Microsoft Outlook and the Siebel Server. For more information, see "Customizing How Siebel CRM Desktop Maps Fields" on page 148.

View Modes That Are Configured in the Client

The view modes that are configured in the client application metadata affects data access. You can configure each synchronization object with a different level of data access. This configuration is implemented as a view mode argument that Siebel CRM Desktop passes to the EAI Siebel Adapter business service during synchronization. This configuration establishes basic access control in the client. For example, data about opportunities is available to the sales representatives who are on the team for the opportunity. To implement this functionality, the default configuration for Siebel CRM Desktop specifies that the opportunity synchronization object use the sales representative view mode. Several view mode arguments are available. For example, All, Organization, Sales Rep, or Personal. For more information, see "About the EAI Siebel Adapter Business Service" on page 27.

How Differences in Data Between Microsoft Outlook and the Siebel Server Affect Synchronization

For the initial synchronization, Siebel CRM Desktop downloads to Microsoft Outlook all data that resides on the Siebel Server that is available to the user. For an incremental synchronization, the changes that occur to data in Microsoft Outlook and on the Siebel Server play a large role in determining the data that Siebel CRM Desktop synchronizes. The following changes can occur:

- Data is created, updated, or deleted in Microsoft Outlook.
- Data is created, updated, or deleted on the Siebel Server.

In general, to determine which data is available, Siebel CRM Desktop does the following work during an incremental synchronization:

1. Uses the set of application configuration filters and user filters that are defined.
2. Identifies the differences between the data in Microsoft Outlook and the data on the Siebel Server. The difference is determined by a comparison of the change key values for all records that are available to the user in Microsoft Outlook and the server. The change key by default includes the record Id and the time the record was last updated in the Siebel database. The value for the record Id resides in the ROW_ID column of the data table. The value for the time resides in the DB_LAST_UPD column of the data table. Depending on the differences, Siebel CRM Desktop changes the values in a data set to make sure the data between Microsoft Outlook and the server is synchronized. For example:
   - If Siebel CRM Desktop detects a new record on the Siebel Server during synchronization, then it creates a corresponding record in Microsoft Outlook.
   - If Siebel CRM Desktop detects a new record in Microsoft Outlook during synchronization, then it creates a corresponding record on the Siebel Server.

For more information, see "How Siebel CRM Desktop Synchronizes Data During an Incremental Synchronization" on page 60.

How the Remove Local Records Synchronization Preference Affects Synchronization

To enable the Remove Local Records feature, the user can make sure the Remove Local Records Not Matching Filtering Criteria check box contains a check mark. This check box displays on the Filter Records tab of the Synchronization Control Panel. This synchronization preference allows the user to remove data that does not match a synchronization filter. If the user changes synchronization filters when this preference is enabled, even if data is not removed from the Siebel Server, then during synchronization Siebel CRM Desktop removes Siebel CRM data from Microsoft Outlook that falls outside of the filters. If this preference is not enabled, and if the user changes a synchronization filter, then data that was synchronized as a result of using a previous filter remains in Microsoft Outlook.

Differences in Data Access Rules

Differences in data access rules that occur from one synchronization to the next can occur for the following reasons:

- The user downloaded a different customization package with a different configuration of synchronization objects, view modes, or internal synchronization filters.
- The configuration of the Siebel repository changed. This can include security logic, search specifications, or other logic in the integration or business object layers.

How You or the User Can Modify Synchronization Frequency

Application settings that are related to synchronization frequency determine how often and what kinds of data Siebel CRM Desktop considers for synchronization. The user can specify frequency, or you can configure it:

- To specify the interval that Siebel CRM Desktop uses to automatically start a synchronization, the user can use the Synchronization tab of the Options dialog box in Siebel CRM Desktop. The user can also manually start a synchronization to override this configuration. The user can double-click the Siebel CRM Desktop icon in the system tray or choose the Synchronize Now option from the options menu.
- You can configure the application metadata to affect the frequency of the synchronization. This configuration determines how often Siebel CRM Desktop synchronizes each object. You can configure certain data that changes less often on the Siebel Server to synchronize less frequently. Example data includes List of Value and other reference data, such as employees or positions. For more information, see “Object Tag of the siebel_meta_info.xml File” on page 281.

How Siebel CRM Desktop Avoids Duplicate Data

Siebel CRM Desktop includes metadata for the client and configuration in the Siebel repository that prevents it from creating duplicate data. This configuration is in addition to the following items:

- The standard user keys included in the Siebel database

- The option to implement data deduplication that you can deploy to prevent duplicate data

To define the data structures that are available to synchronize with Microsoft Outlook, Siebel CRM Desktop uses Siebel integration objects. The integration objects support the definition of a user key that is the first additional layer of duplicate prevention. For information about how to configure integration object user keys and how the EAI Siebel Adapter uses these keys, see *Overview: Siebel Enterprise Application Integration*.

Siebel CRM Desktop also supports configuration of user keys in the metadata for the client. If Siebel CRM Desktop detects a Microsoft Outlook insert during synchronization, then it does the following work:

- Queries the synchronization object in the Siebel database with the user key to determine if there are any existing records that match the record being inserted.
- If Siebel CRM Desktop does not find a match, then it proceeds with the insert operation.
- If Siebel CRM Desktop finds a match, then it raises a synchronization issue that prevents the insert.

How Siebel CRM Desktop Handles a Conflict During Synchronization

This topic describes how Siebel CRM Desktop handles conflicts and potential duplication of Siebel CRM data that might occur during synchronization. It includes the following topics:

- "How Siebel CRM Desktop Prevents Cyclical Synchronization" on page 71
- "How Siebel CRM Desktop Prevents Duplicate Records" on page 73
- "How Siebel CRM Desktop Handles a Synchronization Error" on page 74

How Siebel CRM Desktop Prevents Cyclical Synchronization

This topic describes how Siebel CRM Desktop handles cyclical synchronization.

About Cyclical Synchronization

Siebel CRM Desktop synchronizes PIM data between Microsoft Outlook and Siebel CRM data on the Siebel Server. This situation creates a potential for duplicate records and cyclical synchronization. A potential synchronization problem occurs when two or more synchronizations form a circular loop. *Cyclical synchronization* is a phenomenon that occurs when a single transaction repeatedly loops between Microsoft Outlook and the Siebel Server. For example, consider two separate processes that synchronize appointment data between the PIM system and the Siebel Server, such as between Siebel CRM Desktop and Siebel Server Synchronization for Exchange (SSSE). If a change in a timestamp in the source data store starts these processes and updates the timestamp in the target data store, then a single transaction can repeatedly synchronize between A and B.

How Siebel CRM Desktop Handles Cyclical Synchronization

Siebel CRM Desktop must use the correct order when it synchronizes application objects that contain fields which, in turn, reference other fields. To determine the sequence that Siebel CRM Desktop uses to process the objects, the Synchronization Engine uses the references that exist between these fields. The algorithm that the Synchronization Engine uses to define the sequence also determines if a chain of cyclical links exists. The Synchronization Engine cannot synchronize together any objects that belong to a chain of cyclic links, because Siebel CRM Desktop cannot resolve the reference field for at least one of the objects until it adds the object that this field references to the data file.

To define such an object, the Synchronization Engine uses a required attribute of the field that is referenced. Siebel CRM Desktop defines the required attribute for the link tag in the connector_configuration.xml file. To break the chain of cyclic links, Siebel CRM Desktop chooses an object that does not contain a reference field that is required, and that is part of the chain of cyclic links. In this situation, the Synchronization Engine does the following work:

1. Determines the sequence in which objects are processed. This order depends on the references that exist between these objects.
2. Identifies the chain of cyclical links.
3. For each chain of cyclical links, chooses an object that breaks the chain.
4. Creates jobs to process the objects in an appropriate sequence.
5. Creates more jobs to process the objects that it chose to break each chain.
6. (Conditional). Assume Siebel CRM Desktop does not run the jobs that are described in Step 5. Examples that might cause this situation include an error, a terminated synchronization, and so forth. The field that is referenced that was not removed contains a null value. Therefore, the Synchronization Engine processes the objects that are not removed during the next synchronization.
7. During the next synchronization, Siebel CRM Desktop does the following work:
   a. Processes the objects that it removed during the previous synchronization.
   b. Creates corresponding jobs for these objects.
   c. Assume there is a match between the object on the Siebel Server and the object in Microsoft Outlook, and that the object that Siebel CRM Desktop did not remove from the chain of cyclical links is modified or removed. In this situation, Siebel CRM Desktop does the following:
      - Creates a duplicate to process the object
      - Does not create a job

Example of How Siebel CRM Desktop Handles Cyclical Synchronization

The example in this topic describes how Siebel CRM Desktop chooses the account object to break the chain of cyclic links between the account and business addresses of the account objects. Siebel CRM Desktop does not require the Primary Address Id field of the account object. Siebel CRM Desktop does the following work:

- Adds the account object to the data file
- Resolves the Primary Address Id field to the reference of the account

- Adds the business address of the account to the record

The following code illustrates this example:

```
<type id="Account.Business_Address">
  <synchronizer name_format=":[:(Street Address):]">
     <links>
        <link required="true">BogusParentId</link>
           </links>
     </synchronizer>
     </type>
type id=<"Account">
  <synchronizer name_format=":[:(Name):]">
     <links>
        <link>Primary Address Id</link>
        <link>Primary Position Id</link>
        <link>Primary Territory Id</link>
        <link>Primary Industry Id</link>
           </links>
        </synchronizer>
     </type>
```

How Siebel CRM Desktop Prevents Duplicate Records

Deduplication is a feature that Siebel CRM Desktop uses to make sure that it only synchronizes a single instance of a record between Microsoft Outlook and the Siebel Server. Assume Siebel CRM Desktop downloads a contact to Microsoft Outlook, and then the user attempts to create another contact that contains the same name. In this situation, Siebel CRM Desktop informs the user that a contact with the same name already exists.

Assume a contact that resides on the Siebel Server does not also exist in Microsoft Outlook due to synchronization filters, and the user creates a contact in Microsoft Outlook that contains the same name as the contact on the Siebel Server. In this situation, Siebel CRM Desktop does not detect a duplicate in Microsoft Outlook and the user saves a local copy of the contact. During the next synchronization, Siebel CRM Desktop does the following work:

1. Detects the duplicate conflict.
2. If all contact fields in both data files are the same, then Siebel CRM Desktop resolves the duplicate without interaction with the user.

3 If some of the fields are different, then Siebel CRM Desktop presents the duplicate records to the user, and then prompts the user to choose which values to keep.

The connector_configuration.xml file includes the set of objects that are configured for synchronization and the definition of the natural keys that Siebel CRM Desktop uses for deduplication. For more information, see "How the Origin of an Activity Affects Handling" on page 44.

How Siebel CRM Desktop Uses a Natural Key to Identify Duplicates

This topic describes how Siebel CRM Desktop uses a natural key to identify duplicates.

Insert a Record from Microsoft Outlook to the Siebel Server

If Siebel CRM Desktop detects a potential insert operation for a record from the data store in Microsoft Outlook to the Siebel database on the Siebel Server, then it does the following work:

1 Queries the Siebel database, using natural keys in the query to identify any record in the Siebel database that is a duplicate of the potential insert operation in Microsoft Outlook.

2 If it finds a duplicate, then Siebel CRM Desktop treats the transaction as a duplicate conflict. It returns the record ID of the Siebel CRM record to Microsoft Outlook so that the association between the record in the Siebel database and the record in Microsoft Outlook is defined for use in incremental synchronizations. To determine if the record exists in the Siebel database before it requests an insert, Siebel CRM Desktop performs this work from Microsoft Outlook as a query to the Siebel Server.

Insert a Record from the Siebel Server to Microsoft Outlook

If Siebel CRM Desktop detects a potential insert operation for a record from the Siebel database on the Siebel Server to the data store in Microsoft Outlook, then it does the following work:

1 Queries the data store in Microsoft Outlook, using natural keys in the query to identify any record in the data store that is a duplicate of the potential insert operation from the Siebel database.

2 If Siebel CRM Desktop finds a duplicate, then it treats the transaction as a duplicate conflict. For more information, see Step 2 on page 74 in "Insert a Record from Microsoft Outlook to the Siebel Server" on page 74.

How Siebel CRM Desktop Handles a Synchronization Error

If a top-level error occurs, then the Synchronization Engine stops any further processing and displays a message to the user that describes the error. The engine also logs the message to a log file.

Table 5 describes where the Synchronization Engine stores the error message log file.

Table 5. Where the Synchronization Engine Stores the Error Message Log

Operating System	Directory
Windows XP	C:\Documents and Settings*username*\Application Data\Oracle\CRM Desktop\Profile\Logs\GeneralLog
Windows Vista	C:\Users\user\AppData\Roaming\Oracle\CRM Desktop\Profile\Logs\GeneralLog
Windows 7	C:\Users\user\AppData\Roaming\Oracle\CRM Desktop\Profile\Logs\GeneralLog

The following types of errors can occur in this situation:

- System error
- Resource allocation error
- General storage problem
- Application state malfunction
- Logic failure
- Connectivity problem
- Missing xml or js files in the customization package

If an operation failure occurs in the Synchronization Engine, then Siebel CRM Desktop creates a synchronization issue, and then attempts to perform this operation again during the next synchronization session. The following types of errors can occur in this situation:

- Unexpected failure during an add, update, or delete operation.
- If the data of an object changed since Siebel CRM Desktop queried the object during the current synchronization session, then Siebel CRM Desktop cannot perform the update and delete operations until the next synchronization cycle. In this situation, it creates an issue that it handles in the subsequent synchronization. This synchronization creates a collision because the object was changed since the last synchronization. A *collision* is a situation that occurs when the same record is modified on the Siebel Server and in Microsoft Outlook between synchronization sessions.

6 Installing Siebel CRM Desktop

This chapter describes how to install Siebel CRM Desktop. It includes the following topics:

- Roadmap for Installing Siebel CRM Desktop on page 77
- Process of Preparing the Siebel Server on page 77
- Overview of Installing the Siebel CRM Desktop Add-In on page 85
- Process of Installing the Siebel CRM Desktop Add-In on page 89
- Options for Installing the Siebel CRM Desktop Add-In on page 92

Roadmap for Installing Siebel CRM Desktop

To install Siebel CRM Desktop, you do the following:

1. "Process of Preparing the Siebel Server" on page 77
2. "Process of Customizing Objects in Siebel CRM Desktop" on page 166
3. "Process of Installing the Siebel CRM Desktop Add-In" on page 89

Process of Preparing the Siebel Server

This process is a step in "Roadmap for Installing Siebel CRM Desktop" on page 77.

To prepare the Siebel Server for Siebel CRM Desktop, you do the following:

1. "Preparing the Implementation Environment for Siebel CRM Desktop" on page 77
2. "Administering Metadata Files" on page 78
3. "Creating and Publishing the Customization Package" on page 82
4. "Administering Server Variables" on page 84

Preparing the Implementation Environment for Siebel CRM Desktop

This task is a step in "Process of Preparing the Siebel Server" on page 77.

To prepare the implementation environment for Siebel CRM Desktop

- Make sure the environment in which you intend to implement Siebel CRM Desktop supports Siebel CRM Desktop.

  Before you install Siebel CRM Desktop, you must verify that the supported environments are installed. For more information, see *Siebel System Requirements and Supported Platforms* on Oracle Technology Network.

Administering Metadata Files

This task is a step in "Process of Preparing the Siebel Server" on page 77.

This topic describes how to administer the predefined metadata files that the Siebel CRM Desktop client uses to determine which data to synchronize and the validation rules to apply.

To administer metadata files

1. With administrator privileges, log in to Siebel Sales Enterprise through a Siebel Web Client that is connected to the Siebel Server.
2. Navigate to the Administration - CRM Desktop screen, and then the Metadata Files view.
3. In the Metadata Files list, add a separate record for each of the metadata types:
   - To define the Type field, choose it from the list.
   - To define the File Name field, type the file name in the File Name field.

   For more information, see "Metadata File Types" on page 78.

   Instead of adding a separate record for each metadata type, you can add them all at one time. For more information, see "Uploading All Metadata Files at One Time" on page 81.

Metadata File Types

Table 6 lists each of the metadata file types that you must add in the Metadata Files view of the Administration - CRM Desktop screen. If you must support a language other than English, then see "Metadata File Types That Support Languages" on page 80.

For more information, see "XML Files in the Customization Package" on page 145.

Table 6. Metadata File Types

Metadata Type	Metadata File Name
OL Actions	actions
OL Application Script	application_script
OL SBL Basic Mapping	siebel_basic_mapping
OL Business Logic	business_logic

Table 6. Metadata File Types

Metadata Type	Metadata File Name
OL Connector Configuration	connector_configuration
OL Dialog Definitions	dialogs
OL Resource Bundle ENU	package_res
OL Helpers	helpers
OL Information	info
OL Raw Item Functions	raw_item_functions
OL Lookup View Definitions	lookup_view_defs
OL SBL Metadata Configuration	siebel_meta_info
OL 2003 Form Definitions	forms_11
OL 2007 Form Definitions	forms_12
OL Platform Configuration	platform_configuration
OL Recurrence Processing	recurrence_processing
OL Security Utilities	security_utils
OL View Definitions	views
OL Action Support	actions_support
OL Data Model	data_model
OL Form Helpers	form_helpers
OL Idle	idle
OL SBL Helpers	sb_helpers
OL Security Manager	security_manager
OL 2003 Toolbar Definitions	toolbars_11
OL 2007 Toolbar Definitions	toolbars_12
OL 2010 Toolbar Definitions	toolbars_14
OL Autoresolver	autoresolver
OL Form Script	forms
OL MD5 Scripts	md5
OL MVG Dialog	mvg_dialogs

Metadata File Types That Support Languages

Table 7 lists each of the metadata file types that you must add in the Metadata Files view of the Administration - CRM Desktop screen for specific languages. For all languages except Japanese, you add only one file for each language that you must support. For example, if you must support only German, then add only the package_res.de_DE file. You do not need to add a file to support English. If you must support Japanese, then you must add the following files:

- package_res.ja_JP
- forms_11.jp_JP
- forms_12.jp_JP

Table 7. Metadata File Types That Support Languages

Metadata Type	Metadata File Name
OL Resource Bundle ARA	package_res.ar_EG
OL Resource Bundle CHS	package_res.zh_CN
OL Resource Bundle CHT	package_res.zh_TW
OL Resource Bundle CSY	package_res.cs_CZ
OL Resource Bundle DAN	package_res.da_DK
OL Resource Bundle DEU	package_res.de_DE
OL Resource Bundle ESN	package_res.es_ES
OL Resource Bundle FIN	package_res.fi_FI
OL Resource Bundle FRA	package_res.fr_FR
OL Resource Bundle HEB	package_res.he_IL
OL Resource Bundle ITA	package_res.it_IT
OL Resource Bundle JPN	package_res.ja_JP
OL Resource Bundle KOR	package_res.ko_KR
OL Resource Bundle NLD	package_res.nl_NL
OL Resource Bundle PLK	package_res.pl_PL
OL Resource Bundle PTB	package_res.pt_BR
OL Resource Bundle PTG	package_res.pt_PT
OL Resource Bundle RUS	package_res.ru_RU
OL Resource Bundle SVE	package_res.sv_SE
OL Resource Bundle THA	package_res.th_TH
OL Resource Bundle TRK	package_res.tr_TR

Table 7. Metadata File Types That Support Languages

Metadata Type	Metadata File Name
OL 2003 Forms JPN	forms_11.jp_JP
OL 2007 Forms JPN	forms_12.jp_JP

Uploading All Metadata Files at One Time

You can upload all metadata files at one time.

To upload all metadata files at one time

1. Use a compression application to compress all the client metadata files you use into a single compressed file.

   You can save this compressed file on the computer where you installed the Siebel Server or on the computer where you installed the Siebel CRM Desktop client. This example uses the 7-Zip compression application. You can use a different compression application, such as WinZip or gzip.

2. (Conditional) If the computer where you installed the Siebel Server does not include the executable for this compression application, then install it:

   a. On the computer where you installed the Siebel Server, navigate to a directory where you can install a compression utility.

   b. Install the 7z.exe executable file.

3. Configure Siebel CRM to unzip a batch file:

   a. Open Siebel Tools, and then display the Business Service User Property object type.

      For more information, see "Displaying Object Types in Siebel Tools" on page 156.

   b. In the Object Explorer, click Business Service.

   c. In the Business Services list, query the Name property for PIM Zip Util Service.

   d. In the Object Explorer, expand the Business Service tree, and then click Business Service User Prop.

      The records that Siebel Tools displays in the Business Service User Props list are predefined unzip commands that you can use to unzip a batch file. Siebel Tools only compiles the record with Unzip Command in the name property. Each record supports a different compression application. The default Unzip Command works for Unix. Do one of the following:

      - If you use Unix, then you can use the default record. Proceed to Step i.
      - If you do not use Unix, then proceed to Step e.

   e. In the Business Service User Props list, locate the command described in the following table.

Name	Value
Unzip Command	Unzip -o ${IN} -d ${OUT}

f Change the value in the Name property to Unzip Command Bak4.

g In the Business Service User Props list, locate the command to unzip a 7z.exe file using values from the following table.

Name	Value
Unzip Command 1	"C:\Program Files\7-zip\7z.exe" x ${IN} -o${OUT} -y

This example uses the 7-Zip compression application. Note the following:

- To locate the command for a different compression application, examine all commands in the Business Service User Props list.
- If you cannot locate a command for the compression application you use, then you must create a new business service user property specifically for the application you use.
- Do not change any values in the command, such as IN or OUT. Siebel CRM automatically enters information in these placeholders.
- Do not delete any other command that Siebel Tools displays in this list. Siebel CRM only uses the command that includes Unzip Command in the Name property.

h Set the Name property of the command you located in Step g to Unzip Command.

i If necessary, modify the path in the Value property.

This path must identify the location of the executable for the compression application. If this path includes a space character ( ), then you must enclose the entire path with double quotation marks.

j Compile your changes.

4 Import the metadata files:

a With administrator privileges, log in to Siebel Sales Enterprise through a Siebel Web Client that is connected to the Siebel Server.

b Navigate to the Administration - CRM Desktop screen, and then the Metadata Files view.

c Click Import.

d In the dialog box, click Browse, locate the compressed file that you created in Step 1, and then click OK.

Siebel CRM extracts the metadata files. If you ever upgrade Siebel CRM Desktop, then repeat Step c and Step d.

Creating and Publishing the Customization Package

This task is a step in "Process of Preparing the Siebel Server" on page 77.

To determine the information that is available to the user, you associate a responsibility with a customization package. You can publish a package when the updates are finished and the package is ready to download. Publishing makes a package *read only* so that no more modifications can be performed on the package. For more information, see "Relationships Between Users, Responsibilities, Customization Packages, and Metadata Files" on page 30.

To create and publish the customization package

1 Navigate to the Administration - CRM Desktop screen, and then the Packages view.

2 Create a new customization package, using values from the following table.

Field	Value
Package Name	Enter any value. It is recommended that you define a name that helps describe the purpose of the package configuration. For example, EMEA Sales Rep, or Field Sales Rep.
Responsibility	Choose the responsibility that is appropriate for the target group of users for the package. Do not assign a user to more than one package. To prevent associating a user to more than one responsibility and more than one package, it is recommended that you maintain a separate set of Siebel CRM Desktop responsibilities where you can control the user assignment. If necessary, before you perform this step, you can create a new responsibility, and then assign specific users to this responsibility. For more information, see "Guidelines for Assigning a Responsibility to a Customization Package" on page 84.

3 In the Metadata Files list, add a record for each of the metadata file types.

For more information, see "Metadata File Types" on page 78.

TIP: To simplify adding the files, query the list for the metadata files you must add, position your cursor in the list area of the Add Metadata Files dialog box, and then press and hold down the CTRL key on your keyboard. With CTRL still depressed, press the A key, then release these keys. The Siebel application chooses all records. Note that the Siebel application displays the Add Metadata Files dialog box when you click Add in the Metadata Files list. If you use this technique, then make sure only those records you defined for this customization package displays in the list.

4 In the Packages list, click the link in the Package Name field for the customization package you created in Step 2.

5 In the Package Details form, click Publish.

The Siebel application changes the Status field of the Package Details form to Published.

For more information about:

- How a customization package fits in Siebel CRM Desktop, see "Relationships Between Users, Responsibilities, Customization Packages, and Metadata Files" on page 30.

- How to create a customization package, see "Creating and Publishing the Customization Package" on page 82.

Guidelines for Assigning a Responsibility to a Customization Package

When you develop a customization package you must make sure that you assign the user to only one customization package. Note the following guidelines:

- You must assign one responsibility to a customization package.
- You must not assign more than one responsibility to a customization package.
- You can assign multiple responsibilities to a user, but you can associate only one of those responsibilities with an active customization package.

Administering Server Variables

This task is a step in "Process of Preparing the Siebel Server" on page 77.

To administer server variables

1. Set the maximum page size:
   a. Log in to a Siebel client that is connected to the Siebel Server.
   b. Navigate to the Administration - Server Configuration screen, and then Servers view.
   c. In the Components list, query the Component field for EAI Object Manager.
   d. In the bottom applet, click the Parameters tab, and then query the Parameter field for Maximum Page Size.
   e. In the Component Parameters list, configure the Maximum Page Size parameter using values from the following table.

Field	Value
Default Value	1000
Value on Restart	1000

   f. In the Components list, click Manual Start.
2. Set the DSMaxFetchArraySize parameter:
   a. Navigate to the Administration - Server Configuration screen, and then the Enterprises view.
   b. Query the Profile field of the Profile Configuration view for Server Datasource.

c In the Profile Parameters list, click Advanced Profile Parameters, query the Alias field for DSMaxFetchArraySize, and then make sure the Value is -1 (negative 1).

If DSMaxFetchArraySize is not -1, then Siebel CRM Desktop might not return all records. Certain cautions apply if you set DSMaxFetchArraySize to -1. For more information, see *Configuring Siebel Business Applications*.

3 (Optional) Administer the Generic Siebel Owner system preference.

For more information, see "Administering an Appointment That a Non-Siebel User Creates" on page 133.

4 (Optional) Change the value for the session timeout:

a Navigate to the following directory on the Siebel Server:

siebel_installation_folder/SWEApp

where:

- *siebel_installation_folder* is the location where you installed the Siebel Server

b Open the eapps.cfg file.

c In the default section of the eapps.cfg file, change the SessionTimeout parameter, as required.

The SessionTimeout parameter determines the number of seconds a connection can remain idle before it times out. The default value is 900, which is 15 minutes. You can use the default value or you can adjust this value to meet the specific requirements for your deployment.

5 Stop and then restart the Siebel Server.

Overview of Installing the Siebel CRM Desktop Add-In

An *installation package* is a package that is composed of a Windows Installer (msi) file. The Siebel CRM Desktop application provides you with this file, and you can use it to install the Siebel CRM Desktop add-in on the client computer. It includes the following data:

- The installation information for the Siebel CRM Desktop add-in
- The core resource files and images for all languages that Siebel CRM Desktop supports

You can deploy Siebel CRM Desktop through third-party deployment software that you choose. You can use the distribution criteria in these products to distribute software to any group of users, operating systems, domains, workgroups, and so forth. Systems Management Server (SMS) from Microsoft is an example of deployment software. To deploy the Siebel CRM Desktop add-in to multiple users, you can use deployment software to create a collection, and then distribute the distribution package. A *collection* is a list of users, computers, workgroups, or domains to which you must distribute the software.

You can use third-party deployment software to perform an installation in the background or to perform a removal that uses the default installation parameters. With some deployment software, you can specify various installation parameters.

After you complete the installation and the user starts Microsoft Outlook, the First Run Assistant displays to help the user configure Siebel CRM Desktop. For more information, see "Customizing the First Run Assistant" on page 93.

About Files, File Locations, and Profiles

Siebel CRM Desktop handles files and profiles according to the following conditions:

- To store Siebel CRM data, Siebel CRM Desktop uses a specific profile and a data folder as the default location for email delivery in the chosen profile. You can use a Microsoft Exchange Mailbox or a PST file as the location of the default email delivery for Microsoft Outlook. For more information, see "Caution About Changing the Default Mail Delivery Location" on page 86.
- During installation, you choose the installation directory and the Microsoft Outlook profile where the installer installs the Siebel CRM Desktop add-in. For more information, see "Using the DOS Command Line to Set Optional Parameters" on page 101.
- If Siebel CRM Desktop cannot detect the Microsoft Outlook profile or data folder, or if the profile or folder is invalid, then Siebel CRM Desktop aborts the installation process. For more information, see "Resolving an Invalid Microsoft Outlook Profile" on page 90 and "Setting the Microsoft Outlook Profile for the Siebel CRM Desktop Add-In" on page 104.
- If you configure multiple Microsoft Outlook profiles in Microsoft Outlook, then First Run Assistant installs the Siebel CRM Desktop add-in only to one of the profiles. You cannot install the Siebel CRM Desktop add-in with any other Microsoft Outlook profile.
- If you install and then remove the Microsoft Outlook profile, then the Siebel CRM Desktop add-in remains in an installed state and continues to display in the Add and Remove Programs dialog box. In this situation, the user can remove the Siebel CRM Desktop add-in.
- If the user uses a Microsoft Exchange Server account in cached mode, then Siebel CRM Desktop uses the Microsoft Outlook .ost file to store Siebel CRM Desktop data. Microsoft Outlook synchronizes the data in the .ost file with the Microsoft Exchange Server according to a schedule.
- If you remove the .pst file from the Microsoft Outlook profile where you install Siebel CRM Desktop, then you can still remove the Siebel CRM Desktop add-in. For more information, see "Caution About Changing the Default Mail Delivery Location" on page 86.

Caution About Changing the Default Mail Delivery Location

You must not change the default email delivery location in the Microsoft Outlook profile where you install Siebel CRM Desktop.

CAUTION: If you change the default email delivery location then Siebel CRM Desktop will not function correctly. Changing this location is not supported.

Changes That Siebel CRM Desktop Makes During Installation

This topic describes changes that Siebel CRM Desktop makes during installation to the file system, Windows Registry, and settings in Microsoft Outlook.

Where Siebel CRM Desktop Stores Data in the File System

Siebel CRM Desktop places most files that it requires in the following folder:

APPDATA\Oracle\CRM Desktop\Profile\

where:

APPDATA is an environment variable that the operating system automatically sets.

For example, in Windows XP, Siebel CRM Desktop places most files that it requires in the following folder:

\Documents and Settings*username*\Application Data\Oracle\CRM Desktop\Profile

You can change this directory. For more information, see "Setting the Installation Directory of the Siebel CRM Desktop Add-In" on page 102.

Table 8 describes where Siebel CRM Desktop stores data in the file system when the Siebel CRM Desktop client runs on Windows XP.

Table 8. Example of Where Siebel CRM Desktop Stores Data in the File System

Windows XP Folder on Client Computer	Description
\Documents and Settings*username*\Application Data\Oracle\CRM Desktop\bin	Siebel CRM Desktop saves the following information: ■ Add-in dll files. ■ Resources that binary files use. For example, Microsoft Word helper for Outlook 2003 email processing. ■ Microsoft Visual Studio run-time libraries. ■ Help files.
\Documents and Settings*username*\Application Data\Oracle\CRM Desktop\Profile	Siebel CRM Desktop saves the following information: ■ The Data folder, which includes package files ■ Siebel CRM Desktop logs ■ Database files for the Synchronization Engine

Table 8. Example of Where Siebel CRM Desktop Stores Data in the File System

Windows XP Folder on Client Computer	Description
\Documents and Settings*username*\Application Data\Oracle\CRM Desktop\Profile\Data	XML files and JavaScript files of the customization package. For more information, see "About Files in the Customization Package" on page 144.
\Documents and Settings*username*\Application Data\Oracle\CRM Desktop\Profile\Logs	The Logs folder includes the following items: ■ The CRMDesktop.log file. For more information, see "How Siebel CRM Desktop Handles Errors During Synchronization" on page 133. ■ The following subfolders: ■ **GeneralLog.** Includes log files that are related to general issues. Multiple log files can exist. Siebel CRM periodically deletes older log files. ■ **SoapDump.** Includes log files that are related to SOAP requests. ■ **SyncDump.** Includes log files that are related to synchronization.
Temp\	When you download the customization package Siebel CRM Desktop places some files in a temp directory.

Changes That Siebel CRM Desktop Makes in the Windows Registry

Siebel CRM Desktop stores Siebel CRM Desktop settings in the following registry key:

```
HKEY_CURRENT_USER\Software\Oracle\CRM Desktop
```

These settings include the following information:

- General settings for Siebel CRM Desktop, such as login information.
- In the Logging subkey, logging settings that it uses to tune logging behavior.
- In the StructureBackup subkey, backup information from the Microsoft Exchange folder file or personal folders file. For more information, see "How Siebel CRM Desktop Displays Data in Microsoft Outlook" on page 24.

When you install Siebel CRM Desktop it registers COM classes in the Windows Registry. For more information about Windows Registry settings that Microsoft Windows requires to register COM classes, see the topic about Registering COM Applications at the Microsoft Developer Network Web site.

For more information, see "Modifying the Windows Registry to Change Siebel CRM Desktop Behavior" on page 106.

Changes that Siebel CRM Desktop Makes to Settings in Microsoft Outlook

Because Siebel CRM Desktop runs as a Microsoft Outlook add-in, it must register with Microsoft Outlook. For more information about Windows Registry settings that Microsoft Windows requires to register an add-in, see the topic about Registry Settings for COM Add-Ins at the Microsoft Developer Network Web site.

Siebel CRM Desktop adds the following items to the Microsoft Exchange folder file or personal folders file:

- Custom folders
- Custom objects
- Custom forms

For more information, see "How Siebel CRM Desktop Displays Data in Microsoft Outlook" on page 24.

Process of Installing the Siebel CRM Desktop Add-In

This process is a step in "Roadmap for Installing Siebel CRM Desktop" on page 77. To install the Siebel CRM Desktop add-in, you do the following:

1. "Verifying the Network and Infrastructure" on page 89
2. "Installing the Siebel CRM Desktop Add-In for a Single User" on page 91

For more information, see "Options for Installing the Siebel CRM Desktop Add-In" on page 92.

Verifying the Network and Infrastructure

This task is a step in "Process of Installing the Siebel CRM Desktop Add-In" on page 89.

This topic describes how to make sure you configure the network and infrastructure to successfully install and start the Siebel CRM Desktop add-in.

To verify the network and infrastructure

1. Choose your deployment software and review the conditions that apply for the installation.

   For more information, see "Overview of Installing the Siebel CRM Desktop Add-In" on page 85

2. Make sure a direct connection to the Siebel Server is available.

   To connect to EAI, Siebel CRM Desktop uses information from one of the following sources:

   - **From parameters during installation.** This configuration is appropriate if you install Siebel CRM Desktop in the background. For more information, see "Setting the URL for the Siebel Server" on page 103.

- **From the connection settings dialog box.** This configuration is appropriate if you install Siebel CRM Desktop manually. For more information, see "Customizing How First Run Assistant Uses the Customization Package" on page 94.

3 Make sure the EAI object manager on the Siebel Server is enabled and online.

4 Make sure only a single Position is defined for the user account.

Because the user cannot use Microsoft Outlook to change the position, it is recommended that you define only a single Position for a given user account.

5 Make sure the customization package for the user position is published for only one of the user responsibilities.

6 Verify that the email address you use in the Microsoft Outlook account in the profile where you install Siebel CRM Desktop is the same as the email address for this employee record on the Siebel Server.

7 Make sure the Microsoft Outlook profile is valid.

For more information, see "Resolving an Invalid Microsoft Outlook Profile" on page 90.

8 Make sure you uploaded and published the customization package on the Siebel Server.

For more information, see "About the Customization Package" on page 32.

9 Make sure the number of records for each type of Siebel object, such as accounts, is limited to an amount that the local email folder and environment can handle.

This amount depends on the size of the Microsoft Outlook folder, capabilities of the client computer, the Microsoft Outlook version, connectivity, and so forth. It is recommended that you test these amounts in a test environment before you deploy Siebel CRM Desktop to all users. For more information, see "Regulating the Number of Records That Siebel CRM Desktop Synchronizes" on page 131.

Resolving an Invalid Microsoft Outlook Profile

If Siebel CRM Desktop cannot detect the Microsoft Outlook profile or data folder or finds that the profile or folder is invalid, then Siebel CRM Desktop aborts the installation process. An invalid profile is a profile that does not contain a valid configuration for storing data. In this situation, Siebel CRM Desktop displays an error message that is similar to the following message:

Specified profile "$PREFERRED" is missing, invalid, or not configured

To resolve an invalid profile

1 Verify that the profile you specify for installation exists on the user computer.

2 Verify that the user started Microsoft Outlook with this profile at least one time.

For more information, see "A User Must Start Microsoft Outlook Before You Install Siebel CRM Desktop for Multiple Users" on page 101.

3 Verify that the email account in this profile is one of the following types:

- Exchange Server email account

- Internet email account, which is a POP3 account.

Siebel CRM Desktop does not support IMAP.

Installing the Siebel CRM Desktop Add-In for a Single User

This topic describes how to install the Siebel CRM Desktop add-in for a single user.

To install the Siebel CRM Desktop add-in for a single user

1. Make sure requirements for the operating system are met.

   The CRMDesktop.msi installation package validates the operating system version and the Microsoft Outlook version that is currently installed on the client computer. For more information, see "Preparing the Implementation Environment for Siebel CRM Desktop" on page 77.

2. Make sure Microsoft Outlook is installed on the client computer and configured for use.

   If it is not, then an error occurs and Siebel CRM Desktop aborts the installation.

3. Make sure you possess rights on the client computer so that you can run the executable file that Siebel CRM Desktop provides in the installation package.

4. If Microsoft Outlook is open, then close it.

   CRMDesktop.msi checks if Microsoft Outlook is closed and if the Microsoft Outlook.exe process is stopped. If Microsoft Outlook is open, then CRMDesktop.msi prompts you to close Microsoft Outlook.

5. Copy the CRMDesktop.msi file from the release location to the client computer.

   You can copy the CRMDesktop.msi file to the client computer in one of the following ways:

   - Manually copy the CRMDesktop.msi file to the client computer. This topic describes how to run the CRMDesktop.msi file manually for a single client computer.
   - Use third-party deployment software to deploy the CRMDesktop.msi file to multiple users. For more information, see "Guidelines for Installing the Siebel CRM Desktop Add-In for Multiple Users" on page 99.

6. Locate the CRMDesktop.msi installation package.

   The CRMDesktop.msi installation package is located on the client computer, depending on where you save the CRMDesktop.msi file. The following directory is a typical location:

   `C:\Documents And Settings\`*`username`*`\Desktop`

7. Run the CRMDesktop.msi installation package:

   a. In the Welcome dialog box, click Next.

b In the Destination Folder dialog box, specify the folder in which Siebel CRM Desktop must be installed.

You can specify any directory. For more information, see "Setting the Installation Directory of the Siebel CRM Desktop Add-In" on page 102.

c In the Related Settings dialog box, choose the Microsoft Outlook profile.

The Microsoft Outlook profile determines where Siebel CRM Desktop installs the Siebel CRM Desktop add-in. The list for the Microsoft Outlook profile displays the profiles that are available and configured in Microsoft Outlook for the user who is currently logged in. For more information, see "Setting the Microsoft Outlook Profile for the Siebel CRM Desktop Add-In" on page 104.

d In the Ready to Install the Program dialog box, click Install.

Because you can install Siebel CRM Desktop for multiple users, the user who is currently logged in can view application files that Siebel CRM Desktop stores in the following default directory:

```
c:\Documents and Settings\username1\Application Data\Oracle
```

Siebel CRM Desktop stores the files for another user on this computer in the following directory:

```
c:\Documents and Settings\username2\Application Data\Oracle
```

e In the InstallShield Wizard dialog box, click Finish.

Siebel CRM Desktop installs the Siebel CRM Desktop add-in the background. The next time the user opens Microsoft Outlook, Siebel CRM Desktop starts the First Run Assistant in the client computer.

8 Notify the user that Siebel CRM Desktop is installed and that the user can configure it.

For more information, see "Customizing the First Run Assistant" on page 93.

Options for Installing the Siebel CRM Desktop Add-In

This topic describes options that are available for installing the Siebel CRM Desktop add-in. It includes the following topics:

- "Customizing the First Run Assistant" on page 93
- "Guidelines for Installing the Siebel CRM Desktop Add-In for Multiple Users" on page 99
- "Using the DOS Command Line to Set Optional Parameters" on page 101

Also, you can administer various files to change how the Siebel CRM Desktop application behaves. For more information, see "Changing the Behavior of Siebel CRM Desktop" on page 105.

Customizing the First Run Assistant

The *First Run Assistant* is a wizard that guides the user through the initial setup of the Siebel CRM Desktop add-in. The first time the user starts Microsoft Outlook after you install the Siebel CRM Desktop add-in, Siebel CRM Desktop displays the Siebel CRM Desktop icon in the system tray. The first time the user starts Microsoft Outlook after you install the Siebel CRM Desktop add-in, Siebel CRM Desktop starts the First Run Assistant. After the user finishes using the assistant, the user can begin using Microsoft Outlook.

At each step, the First Run Assistant displays a dialog box where the user can specify certain settings. This topic describes how you can customize the behavior of some of these dialog boxes. For more information, see "Overview of How Siebel CRM Desktop Synchronizes Data" on page 22.

Customizing How First Run Assistant Uses the Customization Package

Table 9 describes work you can perform to customize how First Run Assistant registers and obtains the customization package. It lists work items in the order in which the user performs them while the user uses the assistant. In order for the assistant to start, the user must first install the Siebel CRM Desktop add-in. For more information, see "Installing the Siebel CRM Desktop Add-In for a Single User" on page 91.

Table 9. Steps That First Run Assistant Performs and Customizations You Can Make

Step	Description	Possible Customization
1	The user opens Microsoft Outlook the first time after the user installs the Siebel CRM Desktop add-in. Because it is the first time that Microsoft Outlook is open after the add-in is installed, First Run Assistant displays the welcome screen, and then the user clicks it.	Not applicable
2	First Run Assistant checks the connection settings and does one of the following: ■ If Siebel CRM Desktop establishes a connection, then the assistant continues. ■ If Siebel CRM Desktop does not establish a connection, then the assistant displays the CRM Desktop - Options dialog box with the Connection tab active. Siebel CRM Desktop chooses the Use Internet Explorer Settings for Proxy-Server option by default. The Manual Proxy-Server Configuration option allows the user to specify a proxy server. If your organization uses a proxy server, then you must provide the user with the following information: ■ The host name for the proxy server in the Server window ■ The port number in the window that displays immediately to the right of the Server window The proxy server requires a separate host name and a port number.	For more information, see "Customizing How Siebel CRM Desktop Connects to the Internet" on page 95.

Table 9. Steps That First Run Assistant Performs and Customizations You Can Make

Step	Description	Possible Customization
3	After Siebel CRM Desktop establishes a network connection, First Run Assistant displays the CRM Desktop-Login dialog box. The user enters the user name and password. The user name must be the First Name and Last Name or the User ID of the user record in the Siebel database. The user can enter the First and Last name in any order. The USERID is the same user ID that the user uses for the Siebel Web Client. For example, Wasaka Takuda, or WTAKUDA. The password is the same password as the password that the user uses for the Siebel Web Client.	For more information, see the following topics: ■ "Changing Behavior of the Login Dialog Box" on page 96 ■ "Modifying the Windows Registry to Change Siebel CRM Desktop Behavior" on page 106
4	First Run Assistant automatically enters the URL that the Siebel application uses to connect to the Siebel Server. It enters this URL in the Server URL window. For example: http://*server_name*/eai_enu	You can specify the URL. For more information, see "Setting the URL for the Siebel Server" on page 103.

Customizing How Siebel CRM Desktop Connects to the Internet

You can customize how Siebel CRM Desktop connects to the Internet.

To customize how Siebel CRM Desktop connects to the Internet

1. Use an XML editor to open the platform_configuration.xml file.
2. In the initialization_script section, locate the initialization section.
3. Add the following code:

   ```
   application.settings.set("ProxyUsage", value);
   ```

   where:

   ■ *value* is an integer. Use values from the following table:

Value	Description
0	Use the proxy server setting that is set in Internet Explorer.
1	Use a direct connection to the Internet. This option does not use a proxy server.
2	Use a manual proxy server configuration.

4. Save and then close the platform_configuration.xml file.

5 Test your work.

Changing Behavior of the Login Dialog Box

You can change the behavior of the CRM Desktop-Login dialog box.

To change behavior of the CRM Desktop-Login dialog box

1 To hide the Save Password check box that Siebel CRM Desktop displays in the CRM Desktop-Login dialog box, set the following Windows Registry setting to 1:

Siebel:HideSavePasswordOption

If the user clicks Save Password in the CRM Desktop-Login dialog box, then Siebel CRM Desktop saves an encrypted copy of the password locally in the client computer. If you suppress display of the Save Password check box, then the user must enter the password every time the user logs into Siebel CRM Desktop.

2 To prevent Siebel CRM Desktop from displaying the CRM Desktop-Login dialog box, you do the following:

a Set the following Windows Registry setting to 1:

SuppressLoginDialog

b Define the save_password parameter and the Login externally.

If you do not define the save_password parameter, then Siebel CRM Desktop requires the user to enter the password every time the user opens Microsoft Outlook and then synchronizes.

For more information, see "How Siebel CRM Desktop Suppresses the Desktop-Login Dialog Box" on page 96.

How Siebel CRM Desktop Suppresses the Desktop-Login Dialog Box

If you suppress display of the Desktop-Login dialog box, then Siebel CRM Desktop does the following:

- If the login, password, and URL connection parameters are present in the Windows Registry, and if save_password is present in the Windows Registry and set to 1, then Siebel CRM Desktop attempts to validate the user credentials on the Siebel Server.
- If the Siebel Server returns an error for this login, then Siebel CRM Desktop displays the Desktop-Login dialog box and allows the user to attempt to login or to cancel the login. If the Siebel Server cannot validate the login credentials, then it returns an error.
- If a connection parameter is not present in the Windows Registry, or if save_password is not present in the Windows Registry or is set to 0, then the Siebel Server returns a Credentials Verification Failed error.

Customizing How First Run Assistant Performs the Initial Synchronization

After Siebel CRM Desktop installs the data structure, as described in Table 10 on page 97, the second part of First Run Assistant displays, which prompts the user to set preferences and run the first synchronization session that downloads Siebel CRM records to Microsoft Outlook. Table 10 describes the work that you can perform to customize how the assistant performs this initial synchronization. It lists work items in the order in which the user performs them while the user runs the assistant.

Table 10. Customizing How Siebel CRM Desktop Performs the Initial Synchronization

Step	Description	Administrative Work
1	After First Run Assistant installs the folder structure, it presents the following choices in the Filter Records tab of the Synchronization Control Panel dialog box: ■ Leave the filters at their default settings. ■ Choose a filter from the predefined filter that Siebel CRM Desktop deploys with the Siebel CRM Desktop add-in. ■ Specify filter settings. The user can also specify the synchronization frequency and other settings that Siebel CRM Desktop uses.	For more information, see the following topics: ■ "Controlling the Object Types That Siebel CRM Desktop Displays in the Filter Records Tab" on page 114 ■ "Regulating the Size and Type of Synchronized Records" on page 130
2	The First Run Assistant displays the Synchronization tab of the CRM Desktop - Options dialog box where the user can set the synchronization schedule. By default, Siebel CRM Desktop does the following: ■ Enters a check mark in the Schedule for the Automatic Synchronization Interval check box ■ Enters a check mark in the Show Progress During Automatic Synchronization check box ■ Sets the frequency slide bar to Once an Hour	For more information, see "Controlling the Synchronization Intervals That Siebel CRM Desktop Displays in the Synchronization Tab" on page 114.
3	The First Run Assistant displays the Advanced tab of the CRM Desktop - Options dialog box, where the user can share with Siebel CRM Desktop any new native Microsoft Outlook appointments, contacts, or tasks that the user creates in Microsoft Outlook. By default, Siebel CRM Desktop includes a check mark in the Appointments, Contacts, Tasks check box.	For more information, see "Customizing How Siebel CRM Desktop Shares Native Microsoft Outlook Items" on page 98
4	The First Run Assistant displays the Siebel CRM Desktop dialog box. For more information, see "How Siebel CRM Desktop Converts Contacts" on page 98.	For more information, see, "Controlling How Siebel CRM Desktop Converts Contacts" on page 116.

After the user finishes specifying the configuration settings, Siebel CRM Desktop automatically starts the synchronization and adds content to the CRM folders. This content depends on choices the user specifies in the First Run Assistant. After the synchronization finishes, the user can find the Siebel CRM data that Siebel CRM Desktop downloads in the corresponding CRM folders. The user can view Siebel contacts that Siebel CRM Desktop downloads to the Microsoft Outlook Contacts folders. Contacts that existed in Microsoft Outlook before you installed Siebel CRM Desktop are not automatically shared with Siebel CRM Desktop. The user can use icons or group contacts to separate them from the Siebel CRM contacts according to the Shared and Not Shared attribute.

Customizing How Siebel CRM Desktop Shares Native Microsoft Outlook Items

You can customize Siebel CRM Desktop to share or not share any new native Microsoft Outlook items that the user creates in Microsoft Outlook. These items include Microsoft Outlook appointments, contacts, or tasks.

To customize how Siebel CRM Desktop shares native Microsoft Outlook items

1. Use an XML editor to open the platform_configuration.xml.
2. In the initialization_script section, locate the initialization section.
3. Add the following code:

   ```
   application.settings.set("SharedByDefault:NewItems", <value>);
   ```

   where:

   - *value* is an integer. Use values from the following table:

Value	Description
0	Do not share Outlook item.
1	Share Outlook item.

4. Save and then close the platform_configuration.xml file.
5. Test your work.

How Siebel CRM Desktop Converts Contacts

The Appointments, Contacts, Tasks check box in the Advanced tab of the CRM Desktop - Options dialog box of the First Run Assistant can include one of the following items:

- **Includes a check mark.** Siebel CRM Desktop converts native Microsoft Outlook contacts to Siebel CRM contacts. The user can share these Microsoft Outlook contacts with Siebel CRM. Siebel CRM Desktop does not automatically share contacts. The user must manually choose to share each contact.

- **Does not include a check mark.** The contacts remain as native Microsoft Outlook contacts. After the First Run Assistant finishes, the user can use the Actions menu to convert native contacts to Siebel CRM contacts in an unshared state, and then manually share the contacts with Siebel CRM. If the user does not convert these contacts, then the user cannot share the existing Microsoft Outlook contacts with Siebel CRM.

Guidelines for Installing the Siebel CRM Desktop Add-In for Multiple Users

This topic describes guidelines for installing the Siebel CRM Desktop Add-In for multiple users who share a single computer. For multiple users who use multiple computers you must install Siebel CRM Desktop on each computer for each user.

To install Siebel CRM Desktop silently on the client computer, you can use Microsoft SMS Package or the Microsoft Group Policy Object. You can configure the installation to retry on each user log-on attempt until it is successful. A *silent installation* is a type of installation that runs in the background while the user continues to use the computer. It does not display messages or windows during the installation.

Siebel CRM Desktop includes a separate set of the following items for each user who uses the client computer:

- Binaries for the Siebel CRM Desktop application
- Configuration files
- Settings in the Microsoft Registry

Guidelines for Using a Microsoft SMS Package to Install the Siebel CRM Desktop Add-In for Multiple Users

This topic describes guidelines for using a Microsoft SMS Package to install the CRM Desktop add-in for multiple users who use the same computer. For details on how to use Microsoft SMS Package, see the Microsoft TechNet Web site:

- Make sure the user runs Microsoft Outlook on the client computer at least one time.

  For more information, see "A User Must Start Microsoft Outlook Before You Install Siebel CRM Desktop for Multiple Users" on page 101.

- Create a .mst transform file.

  You can use a third-party tool, such as Orca. This file must contain the custom parameters that Siebel CRM Desktop requires to complete an installation. For example, the name of the Siebel Server, and so forth.

- Create a .bat file that includes the following command:

```
msiexec.exe /qn /i "\\server\install\CRMDesktop\CRMDesktop.msi"
TRANSFORMS="\\server\install\CRMDesktop\transform.mst"
```

- Place the .bat file in the same network path that includes the .msi and .mst files.

- For the Microsoft SMS Package that contains the Siebel CRM Desktop client application, choose the Data Source tab.

  This package contains the source files.

- Make sure the following option contains a check mark:

  Always obtain files from source directory

- For the Program object of Siebel CRM Desktop within the package:
  - Click the General tab.
  - In the Command line window, specify the .bat file and set Run to Hidden.
  - Click the Environment tab, and then enter information using values from the following table.

Window	Value
Program can run	Only when a user is logged on
Run mode	Run with user's rights
Drive mode	Runs with UNC name

- Make sure the computer where you install Siebel CRM Desktop can access the network path to the following files that you specified:
  - .bat
  - .msi
  - .mst

Guidelines for Using the Microsoft Group Policy Object to Install the Siebel CRM Desktop Add-In for Multiple Users

This topic describes guidelines for using the Group Policy Object to install the CRM Desktop add-in for multiple users who use the same computer. For details on how to use the Microsoft Group Policy Object, see the Microsoft TechNet Web site:

- Make sure the user runs Microsoft Outlook on the client computer at least one time.

  For more information, see "A User Must Start Microsoft Outlook Before You Install Siebel CRM Desktop for Multiple Users" on page 101.

- Create an .mst transform file.

  You can use a third-party tool, such as Orca. This file must contain the custom parameters that Siebel CRM Desktop requires to complete an installation. For example, the name of the Siebel Server, and so forth.

- Create the Group Policy Object:
  - Navigate to the Microsoft Active Directory.
  - In the Group Policy Object Editor, create a package under the User Configuration section, not the Computer Configuration.

- Set the Deployment method of the package to Assigned.
- On the Deployment tab of the package Properties, make sure the following option contains a check mark:

  Install this application at logon
- On the Modifications tab, specify the network path that identifies the location of the .mst file and the .msi.

  For example, `\\server\install\CRMDesktop\transform.mst`.
- Make sure the computer where you install Siebel CRM Desktop can access the network path to the .mst file and the .msi file.

A User Must Start Microsoft Outlook Before You Install Siebel CRM Desktop for Multiple Users

If a user starts Microsoft Outlook for the first time on a new workstation, and if a configured email profile for Microsoft Outlook does not exist, and if you attempt a silent installation, then the installation silently fails. For the installation to run successfully, a configured email profile for Microsoft Outlook must exist. Microsoft Outlook creates this profile the first time a user starts Microsoft Outlook. The following sequence must occur:

1. User starts Microsoft Outlook.
2. User quits Microsoft Outlook.
3. You install Siebel CRM Desktop for multiple users.
4. User starts Microsoft Outlook again.
5. Siebel CRM Desktop completes the silent installation.

Using the DOS Command Line to Set Optional Parameters

You can use the DOS command line to set optional parameters that affect installation. This topic includes the following topics:

- "Abbreviating the Installation Procedure" on page 102
- "Setting the Installation Directory of the Siebel CRM Desktop Add-In" on page 102
- "Setting the URL for the Siebel Server" on page 103
- "Setting the Microsoft Outlook Profile for the Siebel CRM Desktop Add-In" on page 104

You can run the CRMDesktop.msi installation package from the DOS command line interface on the client computer. Siebel CRM Desktop supports all parameters that you can set in the Windows Installer msiexec command line. For more information, see the documentation about command line options for Windows Installer at the Microsoft TechNet Web site.

To use the DOS command line to set optional parameters

1. On the client computer, open the DOS command line interface.
2. Navigate to the directory that contains the CRMDesktop.msi file.

   For example, `c:\WINDOWS\system32`.
3. Enter the Windows Installer command using the following format:

   ```
   msiexec.exe /I CRMDesktop.msi optional_parameter_1 optional_parameter_n
   ```

   where:

   - *optional_parameter* is a parameter you can enter that Siebel CRM Desktop runs when you run the CRMDesktop.msi installation package. For example:

     ```
     msiexec.exe /I CRMDesktop.msi INSTALLDIR=c:\My_Custom_Directory
     OL_PROFILE="test PST"
     ```

   Note the following conditions:

   - You must specify each optional parameter in the same command line after the name of the CRMDesktop.msi file.
   - To separate each optional parameter, you must enter a space without a slash (/).
   - You can arrange optional parameters in any order.
4. Press Enter.

   The welcome dialog box of the Siebel CRM Desktop Setup wizard displays.

Abbreviating the Installation Procedure

To automatically run the windows that normally require user action, you can use the optional QR parameter. If you use the QR parameter, then the CRMDesktop.msi installation package does not display dialog boxes that require user action.

To use the QR parameter to abbreviate the installation procedure

- Append the QR parameter to the msiexec command.

  For example:

  ```
  msiexec.exe /I CRMDesktop.msi INSTALLDIR=c:\My_Custom_Directory OL_PROFILE="test
  PST" /QR
  ```

Setting the Installation Directory of the Siebel CRM Desktop Add-In

To change the default location of where the CRMDesktop.msi installation package saves files during installation, you can use the optional INSTALLDIR parameter. CRMDesktop.msi installs to the following directory, by default:

```
c:\Documents and Settings\username\Application Data\Oracle\CRM Desktop\
```

To set the installation directory of the Siebel CRM Desktop add-in

- Enter the following parameter on the msiexec command line anywhere after the mandatory CRMDesktop.msi name parameter:

  INSTALLDIR=*directory_path*

  For example:

  \Documents and Settings*username*\Desktop\CRMDesktop.msi

  where:

  - *user name* is the name of the user, such as WTAKUDA.

Setting the URL for the Siebel Server

To specify the URL of the Siebel Server to which the Synchronization Engine connects, you can set optional parameters.

To set the URL for the Siebel Server

- Enter the following parameters on the msiexec command line anywhere after the mandatory CRMDesktop.msi name parameter:

  SIEBEL_SERVER_PROTOCOL=*protocol* SIEBEL_SERVER_HOST=*host_name_or_address*
  SIEBEL_SERVER_PORT=*server_port* SIEBEL_SERVER_COMPONENT=*component_name*
  SIEBEL_SERVER_SUFFIX=*request_suffix*

  where:

  - *protocol* is http, which is the default value.
  - *host_name_or_address* is the computer name or IP address of the target server. This parameter is empty by default. To use a fully qualified domain name for the server_address variable, you must set the EnableFQDN parameter in the configuration (cfg) file. For more information, see *Siebel System Administration Guide*.
  - *server_port* is 80, which is the default value.
  - *component_name* is eai_enu, which is the default value.
  - *request_suffix* is the following default value:

    start.swe?SWEExtSource=WebService&SWEExtCmd=Execute&WSSOAP=1

  For example:

  ```
  msiexec.exe /I CRMDesktop.msi SIEBEL_SERVER_PROTOCOL=http
  SIEBEL_SERVER_HOST=sdcv440s133.siebel.com SIEBEL_SERVER_PORT=80
  SIEBEL_SERVER_COMPONENT=eai_enu SIEBEL_SERVER_SUFFIX=
  start.swe?SWEExtSource=WebService&SWEExtCmd=Execute&WSSOAP=1
  ```

  No parameters are required.

Because any information that you define in these parameters sets the parameter values in the Windows Registry, the user is not required to set them. For example, the protocol variable of the SIEBEL_SERVER_PROTOCOL parameter overrides the Siebel:Protocol entry in the Windows Registry. For more information, see "Modifying the Windows Registry to Change Siebel CRM Desktop Behavior" on page 106. To override the URL you can also use XML code. For more information, see "XML Code to Customize Forms" on page 258.

Setting the Microsoft Outlook Profile for the Siebel CRM Desktop Add-In

You can set the optional OL_PROFILE parameter in the msiexec command line. In the context of Siebel CRM Desktop, a Microsoft Outlook profile is the standard Microsoft Outlook concept that Siebel CRM Desktop uses to remember the email accounts and the settings that tell Microsoft Outlook where email is stored for the user. The OL_PROFILE parameter specifies the Microsoft Outlook profile where Siebel CRM Desktop is installed.

To set the Microsoft Outlook profile for the Siebel CRM Desktop add-in

- Enter the following parameter on the msiexec command line anywhere after the mandatory CRMDesktop.msi name parameter:

  ```
  OL_PROFILE="my_profile"
  ```

  where:

  my_profile is the name of the profile that is added in Microsoft Outlook.

  For example:

  ```
  msiexec.exe /I CRMDesktop.msi OL_PROFILE="test PST"
  ```

Siebel CRM Desktop supports the following values for the OL_PROFILE parameter:

- **$DEFAULT.** Siebel CRM Desktop uses the default profile or default folder file. The default profile is the profile that you specify in Windows by choosing the Start menu, navigating to Control Panel, Mail, Show Profiles, and then specifying a profile in the window of the Always Use This Profile option. The default folder is the folder that is specified for email delivery in the OL_PROFILE parameter.
- **$PREFERRED.** The following conditions apply:
  - If you do not specify a value for the OL_PROFILE parameter, then Siebel CRM Desktop uses the value in $PREFERRED as the default value.
  - If you specify $PREFERRED as the value for the OL_PROFILE parameter, then Siebel CRM Desktop uses the same algorithm that it uses to determine the value of $DEFAULT. The exception is that if Siebel CRM Desktop does not find a default profile or finds a profile that is not valid, then Siebel CRM Desktop chooses any other suitable profile.

7 Administering Siebel CRM Desktop

This chapter describes how to administer Siebel CRM Desktop. It includes the following topics:

- Changing the Behavior of Siebel CRM Desktop on page 105
- Removing and Upgrading the Siebel CRM Desktop Add-In on page 124
- Administering Metadata on page 127
- Administering Features That Affect Performance on page 128
- Troubleshooting Problems That Occur with Siebel CRM Desktop on page 134

Changing the Behavior of Siebel CRM Desktop

This topic describes how you can change the behavior of Siebel CRM Desktop. It includes the following topics:

- "Modifying the Windows Registry to Change Siebel CRM Desktop Behavior" on page 106
- "Overriding Windows Registry Parameters That Locate the Siebel Server" on page 110
- "Setting Behavioral Limits for Siebel CRM Desktop" on page 111
- "Defining the URL That Siebel CRM Desktop Uses to Open the Siebel Web Client" on page 113
- "Controlling the Object Types That Siebel CRM Desktop Displays in the Filter Records Tab" on page 114
- "Controlling the Synchronization Intervals That Siebel CRM Desktop Displays in the Synchronization Tab" on page 114
- "Controlling the Fields That Are Available in a Filter" on page 115
- "Controlling How Siebel CRM Desktop Converts Contacts" on page 116
- "Controlling How Siebel CRM Desktop Deletes Records During Synchronization" on page 116
- "Configuring the Exclusions List" on page 122
- "Controlling How Siebel CRM Desktop Handles an Archived Item" on page 123

Modifying the Windows Registry to Change Siebel CRM Desktop Behavior

There are several Windows Registry parameters you can modify that affect Siebel CRM Desktop behavior. For example, you can specify visibility of certain dialog boxes, the path where certain files are stored, or if the live update feature is allowed.

CAUTION: Modifying the Windows Registry can cause serious and permanent problems that you might not be able to resolve. You must be very careful to make only the modifications you require, and that the modifications you make do not negatively affect functionality or performance.

For more information, see "Changes That Siebel CRM Desktop Makes in the Windows Registry" on page 88.

To modify Windows Registry parameters to change the behavior of Siebel CRM Desktop

1. To automate changes to Windows Registry parameters, use an administrative tool, such as Systems Management Server or Marimba.
2. Add or modify Windows Registry parameters, as necessary.

   For more information, see "Windows Registry Parameters That Affect Siebel CRM Desktop Behavior" on page 106.

Windows Registry Parameters That Affect Siebel CRM Desktop Behavior

Table 11 describes the Windows Registry parameters that you can modify to change Siebel CRM Desktop behavior. In the Registry Editor (regedit), you can modify these parameters in the following path:

```
HKEY_CURRENT_USER\Software\Oracle\CRM Desktop
```

Table 11. Windows Registry Parameters That Affect Behavior of the Siebel CRM Desktop Add-In

Windows Registry Parameter	Description
AppLanguageID	ID of the current installation package of Microsoft Outlook.
ConnectorCfgPath	Path to the connector_configuration.xml file in the customization package.
DestinationProfile	Name of the Microsoft Outlook profile for which the add-in is installed.
DestinationStore	Name of the Microsoft Outlook profile for which the add-in is installed.

Table 11. Windows Registry Parameters That Affect Behavior of the Siebel CRM Desktop Add-In

Windows Registry Parameter	Description
DisableLiveUpdate	Specifies the live update feature. The following values are valid: ■ **0.** Live update is allowed. ■ **1.** Live update is not allowed.
DisableSyncConfirmation	Suppresses confirmation for deleting an object. The following values are valid: ■ **1.** Suppress confirmation for deleting an object. The corresponding attribute in the connector_configuration.xml file of the customization package automatically overwrites the DisableSyncConfirmation key.
FiltersEstimateOnTimer	Sets the interval in milliseconds for an automatic estimation of records after the filters are changed in the control panel. The following values are valid: ■ **0.** Do not estimate automatically. This entry is not accessible through the administrative interface.
HTTPClient:AcceptCompression	A flag that instructs the Web Service Connector to accept zipped HTTP content. This parameter is not accessible through the administrative interface.
HTTPClient:CompressOutgoing	A flag that instructs the Web Service Connector to send zipped HTTP content. This parameter is not accessible through the administrative interface.
HTTPClient:ConnectTimeout	The timeout for the connection in milliseconds.
HTTPClient:ReceiveTimeout	The timeout for the receiving requests in milliseconds.
HTTPClient:SendRetryCount	The count for the connection retries in milliseconds.
HTTPClient:SendTimeout	The timeout for the sending requests in milliseconds.
MaximumSyncPassthrough	Sets the threshold for the number of objects that Siebel CRM Desktop sends from the Siebel Server to Microsoft Outlook during one synchronization cycle. If the number of objects exceeds the value specified for the MaximumSyncPassthrough parameter, then Siebel CRM Desktop prompts the user to apply a more restrictive filter. The default value is 5000.

Table 11. Windows Registry Parameters That Affect Behavior of the Siebel CRM Desktop Add-In

Windows Registry Parameter	Description
Page:Feedback:AttachLog	Specifies a log for the feedback form. The following values are valid: ■ **0.** Do not attach a log to the feedback form. ■ **1.** Attach a log to the feedback form.
ProxyLogin	The login for the proxy server.
ProxyPassword	The password for the proxy server.
ProxyServer	The host name for the proxy server.
ProxyServerPort	The port number for the proxy server.
ProxyUsage	Flag that specifies a proxy server. The following values are valid: ■ **0.** Do not use a proxy server. ■ **1.** Use a proxy server.
RunPeriodicalSyncAlways	Determines if Siebel CRM Desktop starts a scheduled synchronization at the scheduled time or waits until Microsoft Outlook is idle. The following values are available: ■ **0.** Wait until Microsoft Outlook is idle to start the scheduled synchronization. ■ **1.** Start the scheduled synchronization immediately when the synchronization is scheduled to occur. The value is 1.
SessionsKeepAliveAmount	Defines the number of synchronization sessions to store in the internal database as history. Siebel CRM Desktop stores statistical information for each synchronization session. To view information about synchronization issues, the user can use the list control in the Sync Issues tab of the Synchronization Control Panel.
SharedByDefault:NewItems	Determines how Siebel CRM Desktop shares newly created Microsoft Outlook items. The SharedByDefault:NewItems registry key controls the following option: Always share with Siebel new: Appointments, Contacts, Tasks To access this option, the user right-clicks the Siebel CRM Desktop icon in the system tray, chooses Options, and then clicks the Advanced tab in the CRM Desktop - Options dialog box.

Table 11. Windows Registry Parameters That Affect Behavior of the Siebel CRM Desktop Add-In

Windows Registry Parameter	Description
Siebel:ComponentName	Specifies the component name. The default value is eai_enu. The Siebel:ComponentName parameter is appended to the URL. For more information, see "Overriding Windows Registry Parameters That Locate the Siebel Server" on page 110.
Siebel:HideSavePasswordOption	Determines how Siebel CRM Desktop displays the Save Password check box on the login screen. The following values are valid: ■ **0.** Display the Save Password check box. Siebel CRM Desktop displays this check box by default. ■ **1.** Do not display the Save Password check box.
Siebel:MetaInfoFilePath	Describes the path to the siebel_meta_info.xml file in the customization package.
Siebel:Protocol	Defines the URL protocol. The default is http. The following values are valid: ■ http ■ https
Siebel:RequestSuffix	Defines the suffix of the URL to the Siebel Server. The following is the default value: `start.swe?SWEExtSource=WebService&SWEExtCmd=Execute&WSSOAP=1` For more information, see "Overriding Windows Registry Parameters That Locate the Siebel Server" on page 110.
Siebel:Server	Defines the host name of the URL. The default value is *empty*.
Siebel:ServerPort	Defines the port of the URL. The default value is 80.
SuppressSyncEstimating	Determines how, at the beginning of the synchronization, Siebel CRM Desktop gets an estimated count of all the object types that are visible to the current user. The following values are valid: ■ **0.** Do get the count of all the object types. ■ **1.** Do not get the count of all the object types.
SwitchToAutoCompleteChoiceLimit	Limits the number of items that Siebel CRM Desktop displays in the box in the Filters tab when the user creates rules.

Table 11. Windows Registry Parameters That Affect Behavior of the Siebel CRM Desktop Add-In

Windows Registry Parameter	Description
SyncDumpBaseFname	Defines the full path and file name for the synchronization dump file that Siebel CRM Desktop creates separately for each synchronization. Siebel CRM Desktop stores synchronization history information in this file. Siebel CRM Desktop uses it for investigation purposes.
Siebel:SOAPDumpBaseFname	Describes the path to where Siebel CRM Desktop stores information about SOAP activity. This information describes the requests and replies of the Web Service Connector. For more information, see "Enabling SOAP Data Dumps" on page 134.

Overriding Windows Registry Parameters That Locate the Siebel Server

To override the following registry settings, you can use certain values in the connector_configuration.xml file:

- Siebel:ComponentName
- Siebel:RequestSuffix

To locate the Siebel Server the Siebel CRM Desktop add-in uses these entries. For more information, see "Setting the URL for the Siebel Server" on page 103 and "Modifying the Windows Registry to Change Siebel CRM Desktop Behavior" on page 106.

CAUTION: To edit the name and suffix of the server component, you cannot use the Login dialog box of the Siebel CRM Desktop add-in. If the customization package includes values that cause the server connection to fail, then you must edit these values manually in the Windows Registry on the client computer.

To override Windows Registry parameters Siebel CRM Desktop uses to locate the Siebel Server

- Modify the following default_settings tag that resides in the platform tag of the connector_configuration.xml file:

  ```
  <setting name="Siebel:ComponentName" type="string"> new_value</setting>
  ```

  where:

  - *setting name* is the registry setting that Siebel CRM Desktop must override.
  - *type* is the parameter type. Siebel CRM Desktop supports the following types:
    - **string.** Specifies a string value.
    - **int**. Specifies an integer value.

- *new_value* is the value that overrides the registry setting.

The default_settings tag and all attributes in the default_settings tag are optional. For more information, see "XML Code to Customize Synchronization" on page 251.

Setting Behavioral Limits for Siebel CRM Desktop

You can set limits to affect certain Siebel CRM Desktop behavior. The files that this topic describes are part of the customization package. To modify one of these files, you can use any editor that supports editing in JavaScript or XML, such as Notepad.

To set behavioral limits for Siebel CRM Desktop

1 To set the maximum number of days, weeks, months, or years that Siebel CRM Desktop creates for a recurring event, you can modify the recurrence_processing.js file. Use values from the following table.

Variable With Default Value	Description
var daily_max_length = 12	Sets the maximum number of months after which the daily recurring activity stops occurring.
var weekly_max_length = 12	Sets the maximum number of months after which the weekly recurring activity stops occurring.
var monthly_max_length = 24	Sets the maximum number of months after which the monthly recurring activity stops occurring.
var yearly_max_length = 60	Sets the maximum number of months after which the yearly recurring activity stops occurring.

2 Modify the actions.js file to set certain limits. Use values from the following table.

Variable With Default Value	Description
var max_attach_file_size = 5	Sets the maximum size in megabytes of an attachment file in Microsoft Outlook.

Variable With Default Value	Description
var ask_delete_confirmation = true	Determines if Siebel CRM Desktop displays the Revert Deletions and Accept Deletions buttons in the Confirm Synchronization tab of the Synchronization Control Panel.
var action_selection_limit = 30	Sets the maximum number of items that Siebel CRM Desktop can process when the user runs a toolbar command. For example, if you set var action_selection_limit to 30, then the user can only choose 30 records in Microsoft Outlook, and then use the toolbar in Microsoft Outlook to run a command. An example command is Email to Contacts. To avoid undesirable performance, you can modify this limit.

3 Edit the siebel_meta_info.xml file, using values from the following table. For caution information, see "Customizing Meta Information" on page 152.

Variable	Description
max_commands_per_batch	Sets the maximum number of commands for each batch. For more information, see "Common_settings Tag of the siebel_meta_info.xml File" on page 280.
max_ids_per_command	Defines the maximum number of object IDs. For more information, see "Common_settings Tag of the siebel_meta_info.xml File" on page 280.
open_with_url_tmpl	Sets a template for the code that Siebel CRM Desktop uses to create a URL to open the Siebel Web Client. For more information, see "Defining the URL That Siebel CRM Desktop Uses to Open the Siebel Web Client" on page 113.
ViewMode	Sets the visibility of an object. Siebel CRM Desktop supports the following values: ■ Sales Rep ■ Personal ■ Organization ■ All For example, `ViewMode = Sales Rep`. The value you set is specific to each object. In general, you set ViewMode to Sales Rep for an object that a position determines. Examples of an object that a position determines include a contact, account, or opportunity.

Defining the URL That Siebel CRM Desktop Uses to Open the Siebel Web Client

You can define the URL that Siebel CRM Desktop uses to open the Siebel Web Client when the user clicks the Siebel icon on the CRM Desktop toolbar. To examine more details about the record, the user can navigate to the Siebel Web Client from the context of a record in Microsoft Outlook. This feature is useful when the user must perform work that requires data from the Siebel Web Client that is not available in Microsoft Outlook. Siebel CRM Desktop does the following work:

- Opens a new browser window for the Siebel Web Client when the user clicks the Siebel icon on the CRM Desktop toolbar, or when the user clicks the Siebel icon on the Extension Bar of the form. Siebel CRM Desktop displays the detail form of the record that is currently chosen in Microsoft Outlook. For example, a contact, an opportunity, or an account. A Siebel URL that you can define determines the browser window that Siebel CRM Desktop opens.
- If the user clicks the record but the record does not reside on the Siebel Server, or if the user does not possess visibility to this record, then Siebel CRM Desktop displays a message similar to This Record is Not Found in Siebel. This situation can occur if the user creates the record in Microsoft Outlook but the record is not synchronized to the server.
- Does not display the button if the user chooses a contact that is not shared in Microsoft Outlook.
- Does not display the button if the user chooses a native Microsoft Outlook contact in Microsoft Outlook.
- Displays an error message in the Siebel Web Client if the user does not possess direct visibility to the record. The user responsibility determines this visibility.

For more about:

- Caution information, see “Customizing Meta Information” on page 152
- Templates, see the information about the open_with_url_tmpl variable in “Setting Behavioral Limits for Siebel CRM Desktop” on page 111

To define the URL that Siebel CRM Desktop uses to open the Siebel Web Client

1. Use an XML editor to open the siebel_meta_info.xml file.

   For more information, see “About Files in the Customization Package” on page 144.

2. Locate the definition of the object type for the TypeId attribute of the object tag.

   Examples of these object types include account, contact, opportunity, activity, and so forth.

3. Modify and then insert the open_with_url_tmpl tag in the object element of the object type that Siebel CRM Desktop must open in the Siebel Web Client.

   For more information, see “Open_with_url_tmpl Tag of the siebel_meta_info.xml File” on page 284.

4. Repeat Step 1 and Step 3 for each type of object that Siebel CRM Desktop must open in the Siebel Web Client.

Controlling the Object Types That Siebel CRM Desktop Displays in the Filter Records Tab

You can specify the filter settings for every user and deploy them when you install the Siebel CRM Desktop add-in. This option allows you to customize which filters and other settings that Siebel CRM Desktop displays. By default, Siebel CRM Desktop displays the following objects in the Filter Records tab of the Synchronization Control Panel dialog box:

- Contacts
- Accounts
- Opportunities
- Activities

You can customize the Synchronization Control Panel dialog box to display or not display these object types.

To control the object types that Siebel CRM Desktop displays in the Filter Records tab

1. Use an XML editor to open the connector_configuration.xml file.

   For more information, see "About Files in the Customization Package" on page 144.

2. Locate the type tag for the object you must display or hide.

   For example, type id="Opportunity".

3. Locate the view tag of the type you located in Step 2.

4. Set the suppress_sync_ui attribute to true to hide the object, or false to display the object.

   For more information, see "View Tag of the connector_configuration.xml File" on page 253.

5. Save your changes and then republish the customization package.

   For more information, see "Republishing a Customization Package" on page 127.

6. (Optional) To add a new object to the list of objects that Siebel CRM Desktop displays on the Filter Records tab, you do the following:

   a. In the XML code, add a new type and view tag for the new object.

      Service Requests is an example of a new object.

   b. Set the suppress_sync_ui attribute for this new object to false.

Controlling the Synchronization Intervals That Siebel CRM Desktop Displays in the Synchronization Tab

This topic describes how to control the synchronization intervals that Siebel CRM Desktop displays in the Synchronization Tab of the of the CRM Desktop - Options dialog box.

To control the synchronization intervals that Siebel CRM Desktop displays in the Synchronization Tab

1 Open the platform_configuration.xml file in an XML editor.

For more information, see "About Files in the Customization Package" on page 144.

2 In the initialization_script-CDATA section, add the following code:

```
<initialization_script>

  <![CDATA[

application.settings.set("CustomSyncPeriods", "10 = page-sync-periods-10-
minutes; 20 = page-sync-periods-20-minutes; 30 = page-sync-periods-30-minutes; 60
= page-sync-periods-1-hour; 1440 = page-sync-periods-1-day");

]]>

 </initialization_script>
```

where:

10 = page-sync-periods-10-minutes describes an interval in minutes and a resource string that you add in Step 4.

3 Save and then close the platform_configuration.xml file.

4 Open the package_res.xml file in an XML editor, and then add the following code:

```
<str key="page-sync-periods-10-minutes">Every 10 minutes</str>
<str key="page-sync-periods-20-minutes">Every 20 minutes</str>
<str key="page-sync-periods-30-minutes">Every 30 minutes</str>
<str key="page-sync-periods-1-hour">Once an Hour</str>
<str key="page-sync-periods-1-day">Once a Day</str>
```

5 Republish the customization package on the Siebel Server.

Controlling the Fields That Are Available in a Filter

To control the fields that are available in a filter, you use the IsHidden property of the field in the siebel_meta_info.xml file. For more information, see "How Siebel CRM Desktop Converts Contacts" on page 98.

To control the fields that are available in a filter

1 Use an XML editor to open the siebel_meta_info.xml file.

For more information, see "About Files in the Customization Package" on page 144.

2 Locate the first instance of the tag that defines the field you must modify.

For example, in the Contact.Account object, the following tag defines the Account Status field:

```
<field Name='Account Status' Label='Account Status' DataType='DTYPE_TEXT'
HasPicklist='yes' PicklistIsStatic='yes' PicklistCollectionType='ACCOUNT_STATUS'
PicklistTypeId='PickList_Generic' IOElemName='AccountStatus' />
```

3 Do one of the following:

- To make the field not available in a filter criteria, add the following property to this tag:

  ```
  IsHidden='yes'
  ```

- To make the field available in a filter criteria, add the following property to this tag:

  ```
  IsHidden='no'
  ```

Note that the DataType property must not be DTYPE_ID.

4 Repeat Step 2 through Step 3 for each of the other objects you must modify.

For example, the Contact.Account object also includes the account status.

Controlling How Siebel CRM Desktop Converts Contacts

By default, the First Run Assistant prompts the user to convert Microsoft Outlook contacts to Siebel CRM contacts. You can disallow this feature.

To control how Siebel CRM Desktop converts contacts

1 Use an XML editor to open the platform_configuration.xml file.

For more information, see "About Files in the Customization Package" on page 144.

2 Add the following code to the platform tag:

```
<initialization_script>
  <![CDATA[
   application.settings.set("FRA:SuppressConvertItemsMsgBox", "true");
  ]]>
 </initialization_script>
```

Controlling How Siebel CRM Desktop Deletes Records During Synchronization

You can control how Siebel CRM Desktop deletes records during a synchronization. The *delete confirmation* feature allows the user to cancel, during synchronization, a deletion that the user made in Microsoft Outlook. If you enable this feature, then Siebel CRM Desktop does the following work:

- Displays the Confirm Synchronization tab on the Synchronization Control Panel dialog box.

- Uses the Confirm Synchronization tab to allow the user to confirm the delete operation. If the user deletes records in Microsoft Outlook, then Siebel CRM Desktop displays the Confirm Synchronization tab during synchronization. If the user confirms, then Siebel CRM Desktop removes the deleted records from the Siebel database on the Siebel Server. For more information, see "How the Number of Deleted Records Determines Delete Confirmation" on page 117.

To control how Siebel CRM Desktop deletes records during synchronization

1. Use an XML editor to open the connector_configuration.xml file.
2. To display the Confirm Synchronization tab on the Synchronization Control Panel dialog box, you do the following:
   a. Add the following code to the root tag of the connector_configuration.xml file:

      ```
      <features deletion_confirmation_mode="enable"/>
      ```

      For more information, see "Setting the Delete Confirmation Mode Attribute" on page 119.

   b. Specify the objects that Siebel CRM Desktop displays in the Delete on Siebel list in the Confirm Synchronization tab.

      For more information, see "Specifying the Type of Object the User Can Confirm for Deletion" on page 119.
3. To suppress display of the Confirm Synchronization tab on the Synchronization Control Panel dialog box, add the following code to the root tag of the connector_configuration.xml file:

   ```
   <features deletion_confirmation_mode="suppress"/>
   ```

How the Number of Deleted Records Determines Delete Confirmation

The number of records that the user has deleted determines whether or not Siebel CRM Desktop displays the Confirm Synchronization tab. For example:

- If the user deletes three or more accounts, ten or more contacts, or five or more opportunities, then Siebel CRM Desktop displays the Confirm Synchronization tab.
- If the user deletes only one or two accounts, then Siebel CRM Desktop does not display the Confirm Synchronization tab.

Table 12 lists the minimum number of records that the user must delete to cause Siebel CRM Desktop to display the Confirm Synchronization tab. For information on how to configure this behavior, see "Specifying the Type of Object the User Can Confirm for Deletion" on page 119.

Table 12. Threshold at Which Siebel CRM Desktop Displays the Confirm Synchronization Tab

Object	Number of Deleted Records
Account	3
Account.Account_Note	10
Account.Assignment_Group.Association	5

Table 12. Threshold at Which Siebel CRM Desktop Displays the Confirm Synchronization Tab

Object	Number of Deleted Records
Account.Business_Address (HOR)	10
Account.Business_Address.Association (SIA)	10
Account.Contact.Association	20
Account.Industry.Association	5
Account.Position.Association	20
Action	20
Action.Contact.Association	100
Action.Employee.Association	100
Action_Exception_Squeezzed	Not applicable
Assignment_Group	Not applicable
Attachment	10
Business_Address (SIA)	10
Contact	10
Contact.Business_Address.Association (SIA)	10
Contact.Contact_Note	10
Contact.Contact_Sync_Owner	Not applicable
Contact.Personal_Address (HOR)	10
Contact.Position.Association	20
Currency	Not applicable
Defaults	Not applicable
Employee	Not applicable
Employee.Position.Association	Not applicable
Industry	Not applicable
Internal_Division	Not applicable
Internal_Product	Not applicable
Opportunity	5
Opportunity.Assignment_Group.Association	5
Opportunity.Contact.Association	20
Opportunity.Internal_Division.Association	5
Opportunity.Opportunity_Note	10
Opportunity.Opportunity_Product	5

Table 12. Threshold at Which Siebel CRM Desktop Displays the Confirm Synchronization Tab

Object	Number of Deleted Records
Opportunity.Position.Association	20
Position	Not applicable
Sales_Method.Sales_Cycle_Def	Not applicable

Setting the Delete Confirmation Mode Attribute

To control the Confirm Synchronization Tab on the Synchronization Control Panel dialog box, you use the deletion_confirmation_mode attribute of the connector_configuration.xml file.

Table 13 describes the values you can use for the deletion_confirmation_mode attribute.

Table 13. Values for the Delete Confirmation Mode Attribute

Value	Description
suppress	Disables delete confirmation. Siebel CRM Desktop does not display the Confirm Synchronization tab in the Synchronization Control Panel.
enable	Enables delete confirmation. Siebel CRM Desktop displays the Confirm Synchronization tab in the Synchronization Control Panel. It displays the Revert Deletions button and the Accept Deletions button.
revert_only	Siebel CRM Desktop displays the Confirm Synchronization tab in the Synchronization Control Panel, but enables only the Revert Deletions button. It displays but does not enable the Accept Deletions button. This is the default setting.
user_confirm	Siebel CRM Desktop displays the Confirm Synchronization tab in the Synchronization Control Panel, but displays only the Accept Deletions button. It displays but does not enable the Revert Deletions button.

The following example sets the deletion_confirmation_mode attribute to revert_only. The ellipses (. . .) indicates code that this book omits from this example for brevity:

```
<root>
     <features deletion_confirmation_mode="revert_only"
       . . .
</features>
```

Specifying the Type of Object the User Can Confirm for Deletion

You can specify the type of object that Siebel CRM Desktop displays in the Delete on Siebel list in the Confirm Synchronization tab. For example, you can specify Siebel CRM Desktop to display only opportunity records.

To specify the type of object the user can confirm for deletion

1 Use an XML editor to open the connector_cinfiguration.xml file.

2 Locate the object type that Siebel CRM Desktop must display in the Delete on Siebel list in the Confirm Synchronization tab list.

For example, for opportunities, locate the following object type:

```
type id="Opportunity"
```

3 In the object you located in Step 2 add the synchronizer tag.

For example, add the following tag:

```
<synchronizer name_format=":[:(Name):]" threshold="5">
```

For more information, see "Setting the Synchronizer Tag" on page 120.

4 Repeat Step 2 through Step 3 for each type of object that Siebel CRM Desktop must display in the Delete on Siebel list.

Setting the Synchronizer Tag

The synchronizer tag in the connector_configuration.xml file controls the type of records that Siebel CRM Desktop displays in the Delete on Siebel list in the Confirm Synchronization tab list. It includes a threshold attribute.

In the following example, the threshold attribute set to 5 causes Siebel CRM Desktop to display the Confirm Synchronization tab only if the user deleted five or more opportunities since the last synchronization:

```
<type id="Opportunity" state_field="ObjectState">

  <view label="#obj_opportunity" label_plural="#obj_opportunity_plural"
small_icon="type_image:Opportunity:16" normal_icon="type_image:Opportunity:24"
large_icon="type_image:Opportunity:48"></view>

          <synchronizer name_format=":[:(Name):]" threshold="5">

               <links>

               </links>

               <natural_keys>

               </natural_keys>

          </synchronizer>

     </type>
```

Table 14 describes the values for the threshold attribute of the synchronizer tag.

Table 14. Values for the Threshold Attribute of the Synchronizer Tag

Value	Description
0	Siebel CRM Desktop does not display delete confirmation for the object type.
1	Siebel CRM Desktop displays delete confirmation for the object type.
Any value greater than 1	If you specify any value that is greater than one, then Siebel CRM Desktop does the following: ■ If the value you specify is greater than the number of deleted objects, then Siebel CRM Desktop does not display delete confirmation for the object. ■ If the value you specify is less than or equal to the number of deleted objects, then Siebel CRM Desktop displays delete confirmation for the object.

Configuring the Exclusions List

The Exclusions List allows the user to exclude an individual record from synchronization even if this record matches a defined filtering criteria. Siebel CRM Desktop does the following work:

1. To identify the records that it must synchronize from the Siebel Server, it uses the following filters:
   - Filters that the user creates
   - Master filters. A *master filter* is a type of predefined filter that the user cannot change. For example, a master filter can cause Siebel CRM Desktop to not synchronize a contact that includes an inactive status from Siebel CRM to Microsoft Outlook.
2. Excludes the records that are in the Exclusions List. It excludes each record in the list only if some other record does not reference this record.
3. If a record is listed in the Exclusions List, and if no other record references it, then Siebel CRM Desktop removes this record from Microsoft Outlook even if the Remove Local Records option contains a check mark.

To configure the exclusions list

- Display the Exclusions button on the Filter Records screen of the Synchronization Control Panel:
  a. Use an XML editor to open the connector_configuration.xml file.
  b. Locate the following features node:

     ```
     </features>
     ```

  c. Make sure the following attribute in this features node is set to true:

     ```
     enable_sync_exclusions
     ```

.

Examples of How Siebel CRM Desktop Uses the Exclusions List

Assume the following:

- Contact 1 references account 1.
- Contact 1 matches a filter but account 1 does not match a filter.

In this example, Siebel CRM Desktop synchronizes account 1 because contact 1 references it.

For another example, assume the following:

- Contact 1 references account 1.
- Contact 1 matches a filter and account 1 matches a filter.
- After the first synchronization, Microsoft Outlook displays contact 1 in the Contact folder of the CRM and Personal Contacts view and account 1 in the Accounts folder of the Siebel Accounts view.

- The user deletes account 1 and then Siebel CRM Desktop displays the following prompt:

  Click No to remove the items from filter

- The user clicks No, and then Siebel CRM Desktop moves account 1 to the Exclusions List. At the next synchronization, Siebel CRM Desktop synchronizes account 1 because contact 1 references it.

How Siebel CRM Desktop Automatically Adds Accounts, Contacts, and Opportunities to the Exclusions List

If the user deletes an account, contact, or opportunity in the Explorer view, then Siebel CRM Desktop displays the following prompt:

> Are you sure you would like to delete this item from Siebel and Outlook? Click Yes to delete the items from both applications. Click No to remove the items from filter.

The user can choose one of the following values:

- **Yes.** Siebel CRM Desktop deletes the record in Microsoft Outlook and then deletes it from the Siebel database on the Siebel Server during the next synchronization. If you enable delete confirmation, then Siebel CRM Desktop requests the user to confirm the deletion before it deletes the record from the Siebel database. For more information, see "Controlling How Siebel CRM Desktop Deletes Records During Synchronization" on page 116.
- **No.** Siebel CRM Desktop deletes the record from Microsoft Outlook and adds it to the Exclusions List. If this record is associated with another record in Microsoft Outlook through the lookup field, then Microsoft Outlook displays it the next time the user synchronizes. A lookup field is a field that Siebel CRM Desktop uses to look up an object and then associate it with the current record. The account field on the activity record is an example of a lookup field. For example, assume the following:
  - A contact references a primary account.
  - The user deletes the account from the Explorer view and then clicks No at the confirmation prompt.
  - Siebel CRM Desktop removes the account from the Accounts folder, the Accounts field on the Contact form, and from the Accounts MVG dialog box.
  - The user synchronizes, and then Microsoft Outlook displays the record in the Accounts Lookup dialog box and in the Account field on the Contact form.

Controlling How Siebel CRM Desktop Handles an Archived Item

Siebel CRM Desktop does not distinguish between items in Microsoft Outlook that Outlook archives and items that the user deletes. If Outlook archives an item, then Siebel CRM Desktop interprets this action as a deletion, and it deletes the corresponding Siebel CRM record in the Siebel database. To modify this behavior, you use the archive_activity_days variable in the business_logic.js file. If the deleted Microsoft Outlook event contains the following:

- An end date that occurs further in the past than the current system date minus the number of days that the archive_activity_days variable specifies, then Siebel CRM Desktop does not delete the corresponding Siebel CRM record.
- An end date that occurs more recently than the current system date minus the number of days that the archive_activity_days variable specifies, then Siebel CRM Desktop deletes the corresponding Siebel CRM record.

Siebel CRM Desktop sets the archive_activity_days variable to 7 by default. For example, assume you leave the archive_activity_days variable at the default value. Siebel CRM Desktop does the following:

- If a meeting occurred further in the past than the current system date minus 7 days, and if the meeting no longer exists in Microsoft Outlook, then Siebel CRM Desktop treats the meeting as an archived item. It does not delete the corresponding Siebel CRM meeting from the Siebel database.
- If a meeting was scheduled to occur less than 7 days before the current system date, then Siebel CRM Desktop assumes the user intentionally deleted the meeting, and it deletes the corresponding Siebel CRM meeting from the Siebel database.

To control how Siebel CRM Desktop handles archive items

1. Open the business_logic.js file with a program that can edit JavaScript code.

   For more information, see "About Files in the Customization Package" on page 144.

2. Change the value for the var archive_activity_days attribute.
3. Add your modification to a customization package.
4. Publish the customization package.

   For more information, see "Creating and Publishing the Customization Package" on page 82.

Removing and Upgrading the Siebel CRM Desktop Add-In

This topic describes how you can remove and upgrade the Siebel CRM Desktop add-in. It includes the following topics:

- "Removing the Siebel CRM Desktop Add-In for a Single User" on page 125
- "Removing the Siebel CRM Desktop Add-In for Multiple Users" on page 126
- "Upgrading the Siebel CRM Desktop Siebel CRM Desktop Add-In" on page 126

Removing the Siebel CRM Desktop Add-In for a Single User

To avoid a loss of data, it is recommended that the user synchronize and back up all personal data before removing the Siebel CRM Desktop add-in. For more information, see "How Siebel CRM Desktop Handles Items If the User Removes the Siebel CRM Desktop Add-In" on page 57.

To remove the Siebel CRM Desktop add-in for a single user

1. Make sure all personal data is synchronized and backed up:
   a. In Microsoft Outlook, perform a synchronization.
   b. Backup personal data.

   To backup personal data, it is recommended that the user use the standard Microsoft Outlook feature to export personal data to a file.
2. Remove the Siebel CRM Desktop add-in:
   a. Click the Start menu, choose Settings, and then open the Control Panel.
   b. In the Control Panel, open the Add or Remove Programs application.
   c. In the Currently Installed Programs window, click CRM Desktop, and then click Remove.

   Siebel CRM Desktop automatically does the following work:

   - Removes the data structure
   - Removes Siebel CRM data
   - Removes shared appointments and tasks that Siebel CRM Desktop created to support Siebel CRM activities
   - Removes shared contacts

   Siebel CRM Desktop treats the native Microsoft Outlook items in the following ways:

   - Converts all unshared appointments, tasks, and contacts to native Microsoft Outlook items.
   - Leaves shared appointments and tasks that originated in Microsoft Outlook as native Microsoft Outlook items in their corresponding Microsoft Outlook folders.
   - Leaves all native Outlook items and Outlook email messages in their corresponding Microsoft Outlook folders.

Removing the Siebel CRM Desktop Add-In for Multiple Users

This topic gives one example of using third party deployment software to remove the Siebel CRM Desktop add-in for multiple users. You might use this feature differently, depending on your business model. This topic describes how to use the Systems Management Server (SMS) from Microsoft. For more information, see the topic about how to uninstall a product in the Windows Installer (msiexec) command line options section of the Microsoft TechNet Web site. For more information, see "How Siebel CRM Desktop Handles Items If the User Removes the Siebel CRM Desktop Add-In" on page 57.

To remove the Siebel CRM Desktop add-in for multiple users

1. Open the SMS management console.
2. Open the distribution package you created.
3. Choose Programs.
4. In the programs list, double-click Siebel CRM Desktop.
5. In the command line, enter the following command:

   ```
   %SOE_file_name% /x /QR
   ```

6. Save the distribution package.
7. Run the advertisement for the Siebel CRM Desktop add-in.

Upgrading the Siebel CRM Desktop Siebel CRM Desktop Add-In

To upgrade to a newer version of the Siebel CRM Desktop add-in, you must first remove the old version of the add-in, and then run the CRMDesktop.msi installation package for the new version. For more information, see "How Siebel CRM Desktop Handles Items If the User Removes the Siebel CRM Desktop Add-In" on page 57.

To upgrade the Siebel CRM Desktop add-in

1. Remove the Siebel CRM Desktop add-in.

   For more information, see "Removing the Siebel CRM Desktop Add-In for a Single User" on page 125.

   Note that the removal process automatically deletes the appropriate directories, application files, and data files.

2. Install the Siebel CRM Desktop add-in.

   For more information, see "Installing the Siebel CRM Desktop Add-In for a Single User" on page 91.

Administering Metadata

This topic describes how to administer metadata.

Overview of Administering Metadata

You use metadata administration to create and manage a customization package. A customization package is associated with a set of metadata files that Siebel CRM Desktop treats as a Siebel file attachment. You can perform the following work in metadata administration:

- Upload a metadata file to the Siebel Server.
- Create, update, or delete a customization package.
- Display customization packages that are available, including information about a customization package, such as information about child metadata files.
- Create, update, or delete a metadata file.
- Display metadata files and the details about each metadata file.
- Associate a metadata file with a customization package.
- Download the necessary metadata files through Web services for the user.
- Track the expiration of a customization package and files.
- Control privileges for the user.
- Control access that the user possesses to a customization package.

For more information about:

- Work you perform to administer metadata during installation, see "Administering Metadata Files" on page 78 and "Creating and Publishing the Customization Package" on page 82.
- A description of metadata files that you can administer, see "About Metadata Files" on page 33.
- The process that Siebel CRM Desktop performs to synchronize metadata, see "How Siebel CRM Desktop Synchronizes Data Between the Client and the Siebel Server" on page 59.

Republishing a Customization Package

If you change a metadata file, then you must republish the customization package that references the file. Siebel CRM Desktop downloads the changed metadata files as new metadata file records in the package.

To republish a customization package, it is recommended that you unpublish the old package and then create a new package. This allows you to make sure the new package works as expected. If necessary, you can adjust the new package until it works correctly. To revert to the old package, you can unpublish the new package and publish the old package.

To republish a customization package

1. Unpublish the old customization package:
   a. With administrator privileges, log in to a Siebel client that is connected to the Siebel Server.
   b. Navigate to the Administration - CRM Desktop screen, and then the Packages view.
   c. Query the Package Name field of the Packages list for the package you must refresh.
   d. In the Packages list, click the link in the Package Name field.
   e. In the Package Details form, click Unpublish.
   f. Make sure the Siebel application changes the Status field of the Package Details form to Unpublished.
2. Create and publish a new customization package.

   For more information, see "Creating and Publishing the Customization Package" on page 82.

Administering Features That Affect Performance

This topic describes how to administer and monitor features that affect Siebel CRM Desktop performance. It includes the following topics:

- "Controlling the Time and Day to Perform Synchronizations" on page 128
- "Regulating the Size and Type of Synchronized Records" on page 130
- "Regulating the Number of Records That Siebel CRM Desktop Synchronizes" on page 131
- "Administering Batching Options" on page 132
- "Administering Tracing and Logging" on page 132
- "Administering an Appointment That a Non-Siebel User Creates" on page 133

Controlling the Time and Day to Perform Synchronizations

To manage the load that synchronizations put on the Siebel Server, you can control the day and time of day when Siebel CRM Desktop performs synchronizations. This technique can be useful to manage how, when, and the volume of data that Siebel CRM Desktop synchronizes. For example, to avoid an overloaded server, you can limit synchronizations that occur on a Monday morning after a weekend sales conference.

To control the time and day to perform synchronizations

1 Use an XML editor to open the siebel_meta_info.xml file.

For more information, see "About Files in the Customization Package" on page 144 and "Customizing Meta Information" on page 152.

2 Add a tag named delays_schedule.

3 In the delays_schedule tag, define a request_delay DayOfWeek attribute that meets your scheduling requirements.

For more information, see "Guidelines for Coding the Delays_Schedule Tag" on page 129 and "Example Code to Control the Time and Day to Perform Synchronizations" on page 129.

4 Repeat Step 3, as necessary.

Guidelines for Coding the Delays_Schedule Tag

If you code the delays_schedule tag, then apply the following guidelines:

- Use MON, TUE, WED, THU, FRI, SAT, or SUN to specify a day of the week.
- Use one of the following formats to specify a day:
  - yyyy/mm/dd
  - mm/dd
  - dd

  where:
  - *yyyy/mm/dd* is the year, month, and day.
- To specify a delay, use the DaySpec attribute instead of the DayOfWeek attribute. Siebel CRM Desktop interprets the number you enter for a delay in milliseconds.

For more information, see "Example Code to Control the Time and Day to Perform Synchronizations" on page 129.

Example Code to Control the Time and Day to Perform Synchronizations

The following code controls the delay for when synchronizations can occur:

```
<delays_schedule>

   <request_delay DaySpec="MON" StartTime="12:00:00" EndTime="13:00:00"
   DelayMsecs="12" />

   <request_delay DaySpec="MON" StartTime="12:00:00" EndTime="15:12:00"
   DelayMsecs="13" />

   <request_delay DaySpec="SUN" StartTime="10:22:17" EndTime="16:34:07"
   DelayMsecs="14" />
```

```
    <request_delay DaySpec="SUN" StartTime="10:22:19" EndTime="16:34:07"
    DelayMsecs="15" />

    <request_delay DaySpec="TUE" StartTime="12:00:00" EndTime="13:00:00"
    DelayMsecs="16" />

    <request_delay DaySpec="WED" StartTime="12:00:00" EndTime="13:00:00"
    DelayMsecs="17" />

    <request_delay DaySpec="THU" StartTime="12:00:00" EndTime="17:00:00"
    DelayMsecs="18" />

    <request_delay DaySpec="FRI" StartTime="12:00:00" EndTime="13:00:00"
    DelayMsecs="19" />

    <request_delay DaySpec="SAT" StartTime="12:00:00" EndTime="13:00:00"
    DelayMsecs="20" />

    <request_delay DaySpec="2009/09/01" StartTime="12:00:00" EndTime="13:00:00"
    DelayMsecs="20" />

    <request_delay DaySpec="2009/12/31" StartTime="12:00:00" EndTime="13:00:00"
    DelayMsecs="20" />

    <request_delay DaySpec="05/07" StartTime="12:00:00" EndTime="13:00:00"
    DelayMsecs="20" />

</delays_schedule>
```

Regulating the Size and Type of Synchronized Records

You can regulate the size and type of records that Siebel CRM Desktop synchronizes. By default, there is only one predefined filter, and it is named Default filter. If you use the default filter, then Siebel CRM Desktop downloads the following records from the Siebel Server to Microsoft Outlook:

- All accounts, contacts, and opportunities to which the user possesses visibility
- All activities for which the user is the owner
- All notes that are related to the downloaded records
- All attachments that are related to the downloaded records that are no larger than 5 MB in size and that include one of the following file extensions:
  - doc
  - docx
  - xsl
  - xslx
  - msg
  - txt
  - rtf

- html
- ppt
- pptx
- pdf
- mht
- mpp
- vsd

Siebel CRM Desktop downloads any child record that is related to a parent record for which the user possesses visibility. This logic allows the user to view the child in the appropriate window of the parent record. For example, the Contacts list displays the following information:

- All unshared contacts
- All shared contacts to which the user possesses direct visibility

If the user attempts to open the detail form for a record to which the user does not possess direct visibility, then Siebel CRM Desktop displays a read-only version of the detail form. This read-only form includes only some of the details.

To regulate the size and type of synchronized files

- Administer the Filter Presets tag of the connector_configuration.xml file.
- Notify the user to use the Filter Records tab of the Synchronization Control Panel.

  The Filter Records Tab allows the user to restrict the file type and maximum file size. Note that settings you administer in the Filter Presets tag of the connector_configuration.xml file override settings the user makes in the Filter Records tab. For example, assume you set a maximum file size of 5 MB, and the user sets the limit to 9 MB. In this situation, Siebel CRM Desktop restricts the file size to 5 MB.

  For more information, see "Filter Presets Tag of the connector_configuration.xml File" on page 255, and "Example Code to Set the Size and Type of Field" on page 256.

Regulating the Number of Records That Siebel CRM Desktop Synchronizes

It is recommended that no more than 10,000 records be available to a single user. This recommendation is necessary to produce the most desirable synchronization performance and scalability, and to maintain acceptable Microsoft Outlook performance when the user works with Siebel CRM data. If Siebel CRM Desktop synchronizes more than 10,000 records, then the following undesirable situations might occur:

- Longer synchronization times
- More server resources used to support user synchronization sessions
- Slower Microsoft Outlook performance when working with Siebel CRM data

In addition to the control that the default synchronization filters in Siebel CRM Desktop provide, the application configuration for the Siebel Server defines a maximum query size parameter. If the result is the same as or larger than the maximum query size, then Siebel CRM Desktop returns an error message to Microsoft Outlook that indicates that the number of records requested is too large.

To regulate the number of records that Siebel CRM Desktop synchronizes

1. With administrator privileges, log in to a Siebel client that is connected to the Siebel Server.
2. Navigate to the Administration - Server Configuration screen, and then the Servers view.
3. In the components tab, choose EAI Object Manager.
4. Set the Maximum Page Size parameter.

   For more information, see "Resolving an Exceeded Row Size Problem" on page 137.

Administering Batching Options

To control the overall server load, you can administer the number and size of batch requests.

To administer batching options

1. Use an XML editor to open the siebel_meta_info.xml file.

   For more information, see "About Files in the Customization Package" on page 144 and "Customizing Meta Information" on page 152.

2. Define or find the existing section under the root tag that contains the name common_settings.

   For more information, see "Common_settings Tag of the siebel_meta_info.xml File" on page 280.

3. Define the following subtags:
   - **max_commands_per_batch.** Defines the number of commands for each batch.
   - **max_ids_per_command.** Defines the number of IDs for each command.

Administering Tracing and Logging

The EAI Object Manager performs logging for the business service methods, uses component events to log information for the Siebel Adapter business service, and creates log files that capture the following information on the Siebel Server:

- Input messages
- Time
- Detailed error messages
- Trace

EAI stores these logs in the following file:

Siebel Server install directory\eaiobjmgr_*lang*.log

In UNIX, Siebel CRM Desktop stores these log files at the following location:

$SIEBEL_ROOT/enterprises/*ENTERPRISE NAME*/*SERVER NAME* log

For more information, see the topic that describes event logs in *Siebel System Monitoring and Diagnostics Guide*.

To administer tracing and logging

- Set the EnableLogging parameter to true.

  For more information, see the topic about common event types for Application Object Manager diagnostics in *Siebel System Monitoring and Diagnostics Guide*.

How Siebel CRM Desktop Handles Errors During Synchronization

If the current version of Siebel CRM Desktop is not compatible with the downloaded customization package, then Siebel CRM Desktop does not apply the package. Instead, it displays an error message to notify the user, and then records an entry in the CRMDesktopX.log file, where X is an incremental number that uniquely identifies the log file name. Siebel CRM stores the error message in the most recent log file. For information about where Siebel CRM Desktop stores this log file, see "Where Siebel CRM Desktop Stores Data in the File System" on page 87.

If an error occurs while the customization package is downloaded, then Siebel CRM Desktop displays an error message near the taskbar. The error notifies the user that the customization package changed but Siebel CRM Desktop cannot download it because of certain errors. The problem might be due to the fact that the user does not possess the required privilege on the Siebel Server. In this situation, the user must contact the system administrator to acquire the necessary privileges, and then restart the synchronization.

Administering an Appointment That a Non-Siebel User Creates

To administer an appointment that a non-Siebel user creates, such as an invitation from an external contact, you can use Siebel Multi-Org (multiple organization). For more information, see "How Siebel CRM Assigns the Meeting Organizer" on page 42.

To administer an appointment that a non-Siebel user creates

1. Log in to the Siebel client with administrator privileges.
2. Navigate to the Administration - Application screen, and then the System Preferences view.

3 In the System Preferences list, query the System Preference Name property for Generic Siebel Owner.

   If you do not specify the Generic Siebel Owner parameter, then Siebel CRM sets each user who shares this meeting as the Activity Owner. If more than one of these users synchronizes the same Microsoft Outlook meeting, then duplication error occurs.

4 In the System Preference Value field, enter a user name.

When you enter the value in the System Preference Value field, use the following guidelines:

- Make sure the user you specify as the Generic Siebel Owner is associated with all organizations. This allows any other user who creates a shared meeting to choose the user. If the user you specify as the Generic Siebel Owner is not associated with all organizations, then the selection that the user makes fails and Siebel CRM Desktop displays an error that indicates the user cannot choose the current record.
- Do not specify a real user as the Generic Siebel Owner. If you specify a real user as the Generic Siebel Owner, then this user receives all the appointments that match the criteria.
- It is recommended that you specify SADMIN as the Generic Siebel Owner for the following reasons:
  - SADMIN is not a real user.
  - Users are accustomed to viewing records that SADMIN creates.

Troubleshooting Problems That Occur with Siebel CRM Desktop

This topic describes how to troubleshoot problems that occur with Siebel CRM Desktop. It includes the following topics:

- "Enabling SOAP Data Dumps" on page 134
- "Troubleshooting Problems That Occur When Siebel CRM Desktop Connects to the Siebel Server" on page 136
- "Troubleshooting Problems That Occur During Synchronization" on page 137
- "Troubleshooting Problems That Occur When You Customize Siebel CRM Desktop" on page 139

For information about accessing log files, see "Where Siebel CRM Desktop Stores Data in the File System" on page 87.

Enabling SOAP Data Dumps

You can enable extended SOAP dumps and use them to troubleshoot problems in greater detail. A SOAP dump logs all requests to the Siebel Server and all responses from the Siebel Server.

To enable SOAP data dumps

1 Back up the Windows Registry.

2 On the client computer, add the following key to the Windows Registry:

Key	Value
[HKEY_CURRENT_USER\Software\Oracle\CRM Desktop]	"Siebel:SOAPDumpBaseFname"="x:*path*\\soap_dump.xml" where: *path* identifies an existing path where Siebel CRM stores the SOAP dump.

For more information, see "Windows Registry Parameters That Affect Siebel CRM Desktop Behavior" on page 106.

3 Restart Microsoft Outlook.

Siebel CRM will create SOAP dump files during each synchronization cycle.

Troubleshooting Problems That Occur When Siebel CRM Desktop Connects to the Siebel Server

To resolve a problem that occurs when Siebel CRM Desktop connects to the Siebel Server, look for it in the list of symptoms or error messages in Table 15.

Table 15. Problems That Occur When CRM Desktop Connects to the Siebel Server

Symptom or Error Message	Solution
Siebel CRM Desktop creates an entry in the CRMDesktop.log file that is similar to the following error: Synchronization canceled by error: siebel_service: failed attempting to connect to siebel This error occurs if Siebel CRM Desktop cannot connect to the EAI Object manager.	You can make sure the following items are up and running: ■ Siebel Server ■ Server component for the EAI Object manager
Siebel CRM Desktop creates an entry in the CRMDesktop.log file that is similar to the following error: Exception 'class siebel::siebel_cntr_exception' throwing: siebel_service: SOAP response has unexpected structure. This error typically occurs if the architecture component that Siebel CRM Desktop calls returns an HTTP error instead of an expected SOAP message, and Siebel CRM Desktop cannot parse the error. An Internal Server Error is an example of an HTTP error.	You can look for potential problems in the following logs: ■ SOAP log ■ Web Server log ■ SWE log ■ EAI log

Table 15. Problems That Occur When CRM Desktop Connects to the Siebel Server

Symptom or Error Message	Solution
Siebel CRM Desktop creates an entry in the CRMDesktop.log file that is similar to the following error: Exception 'struct win32_exceptions::inet_cannot_connect' throwing: WinInet: Cannot connect to Internet! !sd_synchronization::salesdesktop_sync_host::load_picklists Synchronization canceled by error: WinInet: Cannot connect to Internet This error occurs if Siebel CRM Desktop cannot connect to the Web server.	You can do the following work: ■ Make sure the Web server is running. ■ Make sure the computer that runs the Siebel CRM Desktop client can connect to the Web server.
When attempting to log in to the Siebel CRM Desktop client Siebel CRM Desktop displays a message that is similar to the following: siebel_service: SOAP response has unexpected structure Siebel CRM Desktop creates an entry in the CRMDesktop.log file that is similar to the following error: Exception 'class siebel::siebel_cntr_exception' throwing: siebel_service: error occurred while retrieving meta data package: This user does not have an active package available. This error occurs if the responsibility that is associated with the user is not associated with a customization package.	Make sure the responsibility that is associated with the user is associated with a customization package. For more information, see "Creating and Publishing the Customization Package" on page 82.

Troubleshooting Problems That Occur During Synchronization

This topic describes how to troubleshoot problems that occur during synchronization. For more information, see "How Siebel CRM Desktop Handles Errors During Synchronization" on page 133.

Resolving an Exceeded Row Size Problem

During the initial synchronization the user might encounter an error that is similar to the following:

There were more rows than could be returned. Please refine your query to bring back fewer rows.

Siebel CRM Desktop returns this message because the number of records that the synchronization query attempts to return is larger than the allowable maximum.

To resolve an exceeded row size problem

1. With administrator privileges, log in to a Siebel client that is connected to the Siebel Server.
2. Navigate to the Administration - Server Configuration screen, and then the Servers view.
3. In the Components tab, choose EAI Object Manager.
4. Change the value for the DSFetchSize parameter.

   The value you set for the DSFetchSize parameter depends on implementation and performance requirements that are specific to your deployment.
5. Notify the Siebel CRM Desktop users to synchronize data.
6. If the problem persists, then create a service request (SR) on My Oracle Support.

   Alternatively, you can phone Oracle Global Customer Support directly to create a service request or get a status update on your current SR. Support phone numbers are listed on My Oracle Support.

Troubleshooting Problems That Occur When You Customize Siebel CRM Desktop

To resolve a problem that occurs when you customize Siebel CRM Desktop, look for it in the list of symptoms or error messages in Table 16.

Table 16. Problems That Occur When You Customize Siebel CRM Desktop

Symptom or Error Message	Solution
After you add a new field to a Siebel CRM Desktop form, Siebel CRM Desktop displays an error that is similar to the following: Updating error of *object name* object on storage {CEDC3D49-DDBB-93A2-4992-E5B2CAE62932}: object_locked (Conversion of input Message to Property Set failed with the error : Cannot convert XML Hierarchy to Integration Object Hierarchy. (SBL-EAI-04111) This error occurs if the definition of an integration object in the SRF does not match the definition of the custom Siebel CRM Desktop form.	You can do the following work: ■ Recompile the relevant integration object. The name for this object uses the following format: CRMDesktopObjectIO where: Object indicates the name of an object type. For example, CRMDesktopContactIO. ■ Deploy the recompiled SRF to the Siebel Server.
You add a new validation rule to a form, but you cannot save or close the form when you add a new record. Siebel CRM Desktop displays an error that is similar to the following: Exception 'class scripting::execute_exception' throwing: Script run-time error. 'phonetest' is undefined (Microsoft JScript runtime error) Line: 6, Char 7 Consider the following custom validation rule: `<rule message="#msg_business_phone_validation"> <expression> <![CDATA[var phonetes = new RegExp(/^[+]?[0-9]{0,3}[\s]?[\(-.]?[0-9]{2,3}[\)-.]?[\s]?[0-9]{3}[-.]?[0-9]{2}[-.]?[0-9]{2}$/); item["Work Phone #"] != '' ? (item["Work Phone #"].match(phonetest) != null ? true : false) : true;]]> </expression> </rule>` This example includes a typographical error in the declaration of the phonetest variable. The variable name is missing the last t character.	You can do the following work: ■ Fix the variable name. ■ Republish the package.

Table 16. Problems That Occur When You Customize Siebel CRM Desktop

Symptom or Error Message	Solution
After you add a new field to a form, Siebel CRM Desktop renders the form with the native Outlook form, and then logs the following error: Exception (struct ml::rethrowable_exception<struct resource_manager::resource_not_found>) encountered: "resource id 'lbl_location' not found" This error occurs if you fail to add a localized value to the package_res.xml file.	You can do the following work: ■ Add the appropriate localized value to the package_res.xml file. For example: <str key="lbl_location">Site:</str> ■ Republish the package.
After you add a new field to a form, Siebel CRM Desktop renders the form with the native Outlook form, and then logs the following error: Exception 'struct xml::parse_error' throwing: Errors occurred during parsing the document! Code: 0xc00ce504 Line: 4 Column: 4 Public ID: unknown System ID: unknown. A name was started with an invalid character. The following message in this log indicates that the XML is not valid: 'struct xml::parse_error'	You can do the following work: ■ Make sure the custom XML you created uses the correct XML format and does not contain errors. ■ To help locate the invalid syntax, identify all changes. You can use a diff tool that compares the XML to a previously working package and then highlights the changes. ■ To identify an XML format error, use an XML schema validation tool.
You remove an existing field from a form. Siebel CRM Desktop renders the form momentarily, but then the form goes blank. Siebel CRM Desktop logs the following errors: 14:46:21:262-2628: Exception 'class scripting::execute_exception' throwing: Script run-time error. 'undefined' is null or not an object (Microsoft JScript runtime error) Line: 77, Char 6 14:46:21:262-2628: Exception (struct ml::rethrowable_exception<class scripting::execute_exception>) encountered: "Script run-time error. 'undefined' is null or not an object (Microsoft JScript runtime error) Line: 77, Char 6" These errors occur if JavaScript references a control that does not exist or if the control is not spelled properly.	You can do the following work: ■ To identify the control Id values, review the deleted elements. ■ Perform a global search for code that references these values. ■ Modify or remove all JavaScript code that references these control Ids. ■ Republish the customization package.

8 Customizing Siebel CRM Desktop

This chapter describes how to customize Siebel CRM Desktop. It includes the following topics:

- Customizing the Siebel CRM Desktop Application on page 143
- Performing Typical Customization Work on page 155
- Process of Customizing Objects in Siebel CRM Desktop on page 166
- Process of Adding a Predefined Siebel Picklist to Siebel CRM Desktop on page 185
- Process of Adding an MVG Field on page 203

Customizing the Siebel CRM Desktop Application

This topic describes options for modifying the customization package. It includes the following topics:

- "Overview of Customizing Siebel CRM Desktop"
- "About Files in the Customization Package" on page 144
- "Customizing How Siebel CRM Desktop Maps Fields" on page 148
- "Customizing Which CRM Data Siebel CRM Desktop Removes if the User Removes Siebel CRM Desktop" on page 149
- "Customizing Synchronization" on page 149
- "Customizing a Form" on page 150
- "Customizing a Toolbar" on page 151
- "Customizing a Dialog Box" on page 151
- "Customizing a View" on page 152
- "Customizing the SalesBook Control" on page 152
- "Customizing Meta Information" on page 152
- "Customizing Localization" on page 153
- "Customizing the URL Protocol to Use HTTPS" on page 154
- "Customizing the Email Address of the Support Team" on page 154

CAUTION: XML files and JavaScript files contain predefined configuration information that is critical to Siebel CRM Desktop operations. If you modify any XML or JavaScript file, then you must be very careful. Only make the minimal modifications that you require. It is recommended that you unit test each change you make.

Related Topics

For more information, see the following topics:

- "Metadata That Describes the Siebel CRM Desktop Application" on page 30.
- "Troubleshooting Problems That Occur When You Customize Siebel CRM Desktop" on page 139

Overview of Customizing Siebel CRM Desktop

You can customize Siebel CRM Desktop to achieve the following:

- Extend the set of data that is available to the user.
- Add custom business logic to support the work that the user performs.

Siebel CRM Desktop can synchronize this information with Siebel CRM data in Microsoft Outlook. You can also make the data available to support offline usage when the user possesses limited or no connection to the Internet. You can create an interface where Siebel CRM Desktop stores the information the user requires to complete a business process. This technique allows the user to work in a single application rather than having to navigate between multiple applications, which simplifies the work process.

You Can Customize How Siebel CRM Desktop Processes Objects

To add custom logic for object processing in Microsoft Outlook, you can use Siebel CRM Desktop functionality, such as deduplication or data validation. You can also use JavaScript to create your own custom logic. Because you can use JavaScript to create, delete, modify, and convert objects, you can migrate your Siebel CRM data to Microsoft Outlook, including logic that supports a business process.

You can use a standard Siebel CRM Desktop form to display Siebel CRM objects, or you can create a completely new and custom form. A custom form can contain native Microsoft Outlook controls, such as a text box, or new controls that you develop for Siebel CRM Desktop, such as a lookup or a multiselect list. You can implement these forms through a custom toolbar, which can contain custom actions that you use JavaScript to implement. Some of the basic functionality that allows the user to use Siebel CRM Desktop functionality comes predefined with the product. To display relationships between Siebel CRM objects, the user can use Microsoft Outlook view controls on custom forms. These views are fully configurable and support the functionality of native Microsoft Outlook views. To display Microsoft Outlook objects, you can set the default views that Siebel CRM Desktop uses.

About Files in the Customization Package

to customize certain Siebel CRM Desktop features, you can modify XML files and JavaScript files in the customization package. Siebel CRM Desktop includes the following basic customization capabilities:

- Adjusting the business logic to suit the business environment
- Customizing the user interface
- Specifying security and data validation rules

For more information, see "Where Siebel CRM Desktop Stores Data in the File System" on page 87.

XML Files in the Customization Package

Table 17 describes the XML files that Siebel CRM Desktop includes in the customization package. For more information, see "About the Customization Package" on page 32.

Table 17. XML Files in the Customization Package

XML File Name	Description
connector_configuration.xml	This XML file provides the following capabilities: ■ Defines objects that are synchronized ■ Defines the criteria that Siebel CRM Desktop uses to detect duplicate objects in the Siebel database ■ Defines the preset filters for a custom synchronization For more information, see "Customizing Synchronization" on page 149.
dialogs.xml	Defines the layout for a custom dialog box. For more information, see "Customizing a Dialog Box" on page 151.
forms_11.xml forms_12.xml	These XML files provide the following capabilities: ■ Indicates the fields that Siebel CRM Desktop uses to store references between objects ■ Defines the form layout for each object ■ Defines the field validation rules on a form ■ Defines business logic in JavaScript ■ Defines controls that Siebel CRM Desktop uses on a form The business_logic.js file describes most business logic. To define business logic, it is recommended that you use JavaScript in one of the forms_xx.xml files only if Siebel CRM Desktop runs this logic in the current form. If many objects are involved, then it is recommended that you define business logic in the business_logic.js file. **NOTE:** Throughout this document, the term forms_xx.xml refers generically to the forms_11.xml file or the forms_12.xml file. For more information, see "Customizing a Form" on page 150.

Table 17. XML Files in the Customization Package

XML File Name	Description
info.xml	This XML file provides the following capabilities: ■ Defines the product name ■ Defines the version of the package ■ Defines compatible product and package versions ■ Defines the general comments for the package Siebel CRM Desktop does not currently use this information. You can use it to track the package version. You can use the product name and version to check compatibility.
lookup_view_defs.xml	Sets configuration options for the SalesBook control in Siebel CRM Desktop. This control defines references between objects. Siebel CRM Desktop uses it in lookup controls. For more information, see "Customizing the SalesBook Control" on page 152.
package_res.xml	Defines various resources for the customization package. For more information, see "Customizing the Email Address of the Support Team" on page 154.
platform_configuration.xml	Defines the custom data that Siebel CRM Desktop deletes if you remove Siebel CRM Desktop. For more information, see "Customizing Which CRM Data Siebel CRM Desktop Removes if the User Removes Siebel CRM Desktop" on page 149.
siebel_basic_mapping.xml	This XML file provides the following capabilities: ■ Defines field mapping between Siebel CRM Desktop and Microsoft Outlook ■ Defines field mapping between Siebel CRM Desktop and a Siebel application ■ Describes objects to add to Microsoft Outlook ■ Defines the form that Siebel CRM Desktop uses to display an object in Microsoft Outlook ■ Defines a set of custom Microsoft Outlook views that Siebel CRM Desktop applies for an object For more information, see "Customizing How Siebel CRM Desktop Maps Fields" on page 148.

Table 17. XML Files in the Customization Package

XML File Name	Description
siebel_meta_info.xml	This XML file provides the following capabilities: ■ Defines the object types that Siebel CRM Desktop supports ■ Defines fields and their types ■ Defines the XML element names that Siebel CRM Desktop uses to build a Siebel message For more information, see "Customizing Meta Information" on page 152.
toolbars.xml	Defines custom toolbars that Siebel CRM Desktop displays on a native Microsoft Outlook form, custom form, or in a Microsoft Outlook window. To generically refer to this file, this book uses toolbars.xml. Siebel CRM Desktop uses the following files according to the version of Microsoft Outlook installed on the client: ■ toolbars_11.xml for Microsoft Outlook 2003 ■ toolbars_12.xml for Microsoft Outlook 2007 ■ toolbars_14.xml for Microsoft Outlook 2010 External JavaScript files define all programmable actions for a toolbar. For more information, see "Customizing a Toolbar" on page 151.
views.xml	Defines the views that Siebel CRM Desktop uses in Siebel CRM Desktop forms and Microsoft Outlook windows. For more information, see "Customizing a View" on page 152.

JavaScript Files in the Customization Package

Table 18 describes the JavaScript files that Siebel CRM Desktop includes in the customization package.

Table 18. JavaScript Files in the Customization Package

JavaScript File Name	Description
actions.js	Defines actions for the toolbar.
actions_support.js	Defines action support functions for the toolbar.
autoresolver.js	Defines functions to resolve conflicts.
application_script.js	Defines entry points for scripts and to call scripts from other files.

Table 18. JavaScript Files in the Customization Package

JavaScript File Name	Description
business_logic.js	Defines logic for the following items: ■ Activities ■ Mail processing ■ The data model
data_model.js	Defines the data model and functions for objects.
form_helpers.js	Defines functions to handle user interface events.
forms.js	Defines user interface actions.
helpers.js	Defines utility functions.
idle.js	Defines the Idle Processing Manager and idle handlers.
md5.js	Defines how to implement MD5 (Message Digest Algorithm).
mvg_dialogs.js	Defines controls for multi-value groups.
raw_item_functions.js	Defines functions that access Microsoft Outlook items.
recurrence_processing.js	Defines patterns for recurrence processing. For more information, see "How Siebel CRM Desktop Transforms Objects Between Siebel CRM Data and Microsoft Outlook Data" on page 238.
sb_helpers	Defines utility functions that are specific to Siebel CRM.
security_manager.js	Defines the security model.
security_utils.js	Defines security definitions that are specific to Siebel CRM.

Customizing How Siebel CRM Desktop Maps Fields

The siebel_basic_mapping.xml file describes objects you can add to Microsoft Outlook. It defines mapping between a Siebel CRM field and a Microsoft Outlook field. You can also extend a set of fields that native Microsoft Outlook objects reference. Each description for an object includes the following information:

- The forms_xx file is dependent on the siebel_basic_mapping file. The forms_xx file defines the forms. The siebel_basic_mapping identifies the form that each object uses.
- The icons that Siebel CRM Desktop uses to display the object. Siebel CRM Desktop uses these icons in Microsoft Outlook views.
- The folder name for the object in the Microsoft Outlook navigation pane.
- A set of custom Microsoft Outlook views that Siebel CRM Desktop applies to the Microsoft Outlook view.

For more information, see "XML Code to Map a Field" on page 245.

Customizing Which CRM Data Siebel CRM Desktop Removes if the User Removes Siebel CRM Desktop

The platform_configuration.xml file allows you to specify which custom data Siebel CRM Desktop removes from Microsoft Outlook. If the user uninstalls Siebel CRM Desktop or changes credentials, then Siebel CRM Desktop removes from Microsoft Outlook all the custom data that the siebel_basic_mapping.xml file describes. You can configure Siebel CRM Desktop to not delete data for a certain object type. Note that Siebel CRM Desktop does not remove Microsoft Outlook data, which is data that the user creates in the native Microsoft Outlook application. For more information, see "XML Code to Customize the Data That Siebel CRM Desktop Deletes if You Remove Siebel CRM Desktop" on page 250.

To customize which CRM data Siebel CRM Desktop removes if the user removes Siebel CRM Desktop

1. Use an XML editor to open the platform_configuration.xml file.
2. To configure Siebel CRM Desktop to not delete data for a specific object type, define a rule named skip for the appropriate type tag.

   For example:

   ```
   type id="Action" rule="skip"/>
   ```

   In this example, Siebel CRM Desktop does not delete any data that is associated with the Action object type.
3. To configure Siebel CRM Desktop to conditionally not delete data for a specific object type, define an attribute named *language* for the rule you defined in Step 2.

   The language attribute defines the script language in the CDATA section. For example:

   ```
   <type id="Action" rule="script" language="JScript">
       <![CDATA[ … JavaScript code … ]]>
   </type>
   ```

   Siebel CRM Desktop supports only the JScript language.
4. To configure Siebel CRM Desktop to not delete data for multiple object types, add a separate rule for each object type.

   For example:

   ```
   <type id="Action" rule="skip"/>
   <type id="Opportunity" rule="skip"/>
   ```

Customizing Synchronization

The connector_configuration.xml file defines objects that Siebel CRM Desktop synchronizes. It includes synchronization settings that affect the following types of filters:

- **Deduplication filters.** To determine if an item already exists in the Siebel database when Siebel CRM Desktop uploads the item from Microsoft Outlook to the Siebel database, it uses special criteria that the connector_configuration.xml file describes. For more information, see "How Siebel CRM Desktop Prevents Duplicate Records" on page 73.
- **Preset filters.** Defines preset filters for a custom synchronization. These presets help the user to synchronize only the data that the user requires, which helps to avoid downloading data that the user does not require. You can configure any filter preset as the default.

For more information, see "XML Code to Customize Synchronization" on page 251.

Customizing a Form

The forms_11.xml and forms_12.xml files contain a definition for customized forms in Siebel CRM Desktop. These files allow you to customize forms, remove fields, change field names, and set custom list views.

The Microsoft Outlook version on which the Siebel CRM Desktop add-in runs determines which form Siebel CRM Desktop uses:

- The forms_11.xml file describes Microsoft Outlook 2003 forms.
- The forms_12.xml file describes Microsoft Outlook 2007 forms.

Each form is described in a form tag. Siebel CRM Desktop includes the following customization capabilities for these files:

- Changing field labels.
- Removing fields from the form that are not applicable to the work environment.
- Identifying fields on forms.
- Designating field type. Siebel CRM Desktop supports the following:
  - Native Microsoft Outlook field types, such as text box or check box
  - Custom Siebel CRM Desktop controls, such as currency control, lookup, and multiselect list
- Positioning a field on the form.
- Modifying custom Siebel CRM Desktop controls.
- Creating entirely new forms that use the look and feel of Microsoft Outlook. To enter data in these forms, Siebel CRM Desktop uses fields that the user defines.
- Programmatically applying custom behavior and logic to a form or field.
- Setting Microsoft Outlook view controls on a custom form to display relationships between Siebel CRM objects.
- Modifying interface elements, such as text in a system message, a dialog box, a label, or a caption.

For more information, see "XML Code to Customize Forms" on page 258, and "Correct Usage of the Forms_xx.XML File and Object ID" on page 178.

Validation Rules That Siebel CRM Desktop Uses with Customizing a Form

The business_logic.js file contains a description of the validation rules that Siebel CRM Desktop uses for the object that it displays on any form. You can configure Siebel CRM Desktop to examine the format of information that the user enters in a field and then inform the user if this information does not adhere to the format. Because validation uses JavaScript functions, you can combine these functions with JavaScript RegEx (JavaScript Regular Expressions) to configure a wide variety of validation. To define validation rules, you can also use JavaScript in the forms_11.xml file and the forms_12.xml file. For more information, see "Defining the Validation Rules" on page 179.

Business Logic That Siebel CRM Desktop Uses with Customizing a Form

The forms_11.xml file or the forms_12.xml file contain some of the business logic. You must implement them in JavaScript. JavaScript capabilities in Siebel CRM Desktop allow you to access any field of an object, or any property of a Microsoft Outlook form. You can use this access to run code on a specific event, such as opening a form, saving a form, and so forth. You can configure Siebel CRM Desktop to fill fields automatically, format field values automatically, and disable or enable a control, depending on certain criteria.

Customizing a Toolbar

The toolbars.xml file describes the custom toolbars that Siebel CRM Desktop displays on a native Microsoft Outlook form, a custom form, or in a Microsoft Outlook window with programmable actions on the toolbar. Because some of the basic functionality comes predefined with Siebel CRM Desktop, it is possible to use Siebel CRM Desktop without modifying it to meet your basic requirements. For more information, see "XML Code to Customize Toolbars" on page 273.

Customizing a Dialog Box

The dialogs.xml file defines the layout for a custom dialog box that Siebel CRM Desktop uses, such as the dialog box that it uses with an MVG, an address, or for email processing. Because the dialogs.xml file is an extension of the forms_11.xml file and the forms_12.xml file, it contains the same structure as the dialogs root tag instead of the forms tag in the forms_11.xml file and the forms_12.xml file. For more information, see "XML Code to Customize Dialog Boxes" on page 275.

To customize the behavior of the MVG dialog box, you can modify the XML file that Siebel CRM Desktop includes in the customization package. You can customize the following MVG behaviors for Siebel CRM Desktop:

- Object that it uses to create an association in the MVG dialog box.
- Fields that it specifies for the association.
- Format of the fields that it specifies for the association.

- Format in which it displays the Primary record. The following is the default format:

  position (name)

  For example, District Manager 1 (Wasaka Takuda). You can specify the fields that Siebel CRM Desktop displays and the order in which it displays them.

- Fields that it displays in the Microsoft Outlook view that represent the associations you create. Siebel CRM Desktop can display only the attributes of the record. If Siebel CRM Desktop stores the Employee name in the position record, then it can display only the Employee name for the position.
- The user permissions. To describe the user permissions that work with the MVG, the customization package uses security validation rules. For example, if the user is not the primary user, then the user cannot delete users from the collection because Siebel CRM Desktop turns off the delete button for any user who is not a primary user.
- Behavior of a lookup control. To search for a record, a lookup control searches through the File As field of associated objects.
- Hide the details of the parent record, such as the Opportunity Name.
- Add or remove association attributes for the associated record.
- Use an OK button instead of the Save and Close icon.

Customizing a View

The views.xml file describes the view configurations that Siebel CRM Desktop uses on Siebel CRM Desktop forms and in Microsoft Outlook views. Each Microsoft Outlook view uses an XML file that is native to Microsoft Outlook. Siebel CRM Desktop reuses this XML file for custom views. For more information about:

- Customizing a view in Siebel CRM Desktop, see "XML Code to Customize Views" on page 276.
- Customizing a view that is native in Microsoft Outlook, see the Microsoft Outlook documentation on the Microsoft Web site.

Customizing the SalesBook Control

The lookup_view_defs.xml file sets configuration options for the SalesBook control in Microsoft Outlook. This control defines references between objects. Siebel CRM Desktop uses it primarily in lookup controls. To specify the objects that Siebel CRM Desktop makes available through each SalesBook control, you can use the lookup_view_defs.xml file. You can also set filters for the objects you must display in the SalesBook control. For more information, see "XML Code to Customize the SalesBook Control" on page 277.

Customizing Meta Information

The siebel_meta_info.xml file contains the following meta information:

- A description of the object types that Siebel CRM Desktop supports.
- Fields that are defined and the type for each field.
- XML element names that Siebel CRM Desktop uses to build or parse a Siebel message. Siebel CRM Desktop uses information about the relations between objects from the file for the Siebel message.
- The definition of each object that Siebel CRM Desktop supports. This definition contains a unique name, an XML element, and an XML collection element that Siebel CRM Desktop uses in a Siebel message.

Every object field includes a name, a Siebel data type, and an XML element. Siebel CRM Desktop uses this information in a Siebel message to display values and filters for that field.

For more information, see "XML Code for Meta Information" on page 279.

Customizing Localization

To customize localization, you can customize resource files.

To customize localization

1. Add a new resource for the custom label and attribute name or warning message in the following files:
   - Use the package_res.xml file for the default resources.
   - Use the package_res.*xx_YY*.xml file for the specific locale.

   where:
   - *xx_YY* is the language you use in your implementation.

   For example, for Portuguese Brazilian you use package_res.pt_BR.xml.

   The following standards determine the locale naming convention:
   - **xx.** The ISO 639-1 standard for the language.
   - **YY.** The ISO 3166-1 standard for the country. This standard supports dialects and language adoptions for specific countries.
2. Add the XML files to the customization package.
3. Republish the customization package.

   For more information, see "Republishing a Customization Package" on page 127.

Customizing Language for the Forms Files

To customize the behavior of the forms_xx.xml file that comes predefined with Siebel CRM Desktop, you can use a forms file that is specific to a language. For example, for JPN (Japanese), you use forms_12.ja_JP.xml.

To customize language for the forms files

1 In Windows Explorer, navigate to the directory that contains the forms_xx.xml file.

For more information, see "About Files in the Customization Package" on page 144.

2 Right-click the forms_xx.xml file, and then choose the Copy menu item.

3 Rename this copy to indicate that it is a specific to a language.

For example, you can rename the file to forms_12.ja_JP.xml.

4 Use an XML editor to open the file you renamed in Step 3, and then make any changes that are required to support the language.

You can use this file to change the layout of the form, such as adding new fields. For example, in Japanese, you might define three different fields for the Account Name:

- One field represents the native name of the company for the account.
- One field represents the Kanji symbol for the company.
- One field represents the phonetic name of the company.

If you make these changes, then make sure you also perform other configuration that the changes require, such as defining field mappings and form layout changes.

5 (Optional) To change text strings, repeat Step 2 and Step 3 but create a copy of the package_res.xml file, rename it, and then edit the text strings.

For example, you can rename the file to package_res.ja_JP.xml.

Customizing the URL Protocol to Use HTTPS

You can customize the URL protocol to use HTTPS (Hypertext Transfer Protocol Secure). For more information, see "Setting the URL for the Siebel Server" on page 103.

To customize the URL protocol to use HTTPS

1 Open a DOS command line interface, and then type `regedit.exe`.

2 Set the Siebel:Protocol registry parameter to https.

For more information, see "Modifying the Windows Registry to Change Siebel CRM Desktop Behavior" on page 106.

Customizing the Email Address of the Support Team

The package_res.xml file defines various resources for the customization package. In this file, you can specify the email address of the support team to which the user sends feedback.

To customize the email address of the support team

1 Use an XML editor to open the package_res.xml file.

2 Modify the following code of:

```
<!-- Feedback page -->

     <str key="support_email">email_address</str>
```

where:

- *email_address* is the email address to which Siebel CRM Desktop sends requests for support

For example:

```
<str key="support_email">support@your_company.com</str>
```

If you specify the email address in the support_email variable, and if the user clicks Send Feedback on the Feedback tab in the Options dialog box, then Siebel CRM Desktop does the following work:

- Opens a new email message.
- Automatically enters the value that you specify in the support_email variable. It enters this information in the To line of this email message.

If the user clicks the Send Feedback button on the Feedback tab in the Options dialog box, and if the email address is not specified, then Siebel CRM Desktop opens the email without an email address in the To line. A support email address is not specified in the predefined Siebel CRM Desktop product.

Performing Typical Customization Work

This topic describes other customization work you might find useful. It includes the following topics:

- "Checking Out a Project in Siebel Tools" on page 155
- "Displaying Object Types in Siebel Tools" on page 156
- "Customizing the Product Name" on page 156
- "Localizing Values" on page 157
- "Displaying a Custom Siebel CRM Field in a Siebel CRM Desktop Form" on page 159
- "Making a Field in Siebel CRM Desktop Read-Only" on page 163
- "Adding a Default Value to a Field in Siebel CRM Desktop" on page 164
- "Preventing the User From Deleting Records" on page 165

Checking Out a Project in Siebel Tools

This topic describes how to check out a project in Siebel Tools.

To check out a project in Siebel Tools

1 Identify the project you must check out.

For example, if you must modify an applet, then note the value in the Project property for the applet.

2 In the Object Explorer, click Project, and then query the Name property for the project you identified in Step 1.

3 Make sure the Locked property contains a check mark.

Displaying Object Types in Siebel Tools

You can display object types in the Object Explorer that you use to configure Siebel CRM Desktop.

To display object types in Siebel Tools

1 Open Siebel Tools.

2 Choose the View menu, and then the Options menu item.

3 Click the Object Explorer tab.

4 Scroll down through the Object Explorer Hierarchy window until you locate the Integration Object tree.

5 Make sure the Integration Object tree and all child objects of the Integration Object tree include a check mark.

If all child objects in the Integration Object tree include a check mark, then Siebel Tools displays a black check mark with a white background for the tree.

6 Repeat Step 4 for any other object types you must modify.

7 Click OK.

Customizing the Product Name

The Siebel CRM Desktop client displays the following text in a number of locations:

- Siebel
- CRM Desktop
- Outlook

You can change this text to a custom value.

To customize the product name

1 Use an XML editor to open the package_res.xml file.

For more information, see “About Files in the Customization Package” on page 144.

2 Create or modify any of the following attributes, as required:

- <str key="app_name">CRM Desktop</str>
- <str key="pim_name">Outlook</str>
- <str key="remote_app_name">Siebel</str>

For example, in the remote_app_name attribute, change Siebel to the name of your company.

3 Save your changes, republish, and then test your changes.

For more information, see "Republishing a Customization Package" on page 127.

Localizing Values

You can modify a single file to do the following work:

- Support different languages.
- Change a label that Siebel CRM Desktop displays in multiple locations in the user interface.

To localize values

1 Use an XML editor to open the package_res.xml file.

For more information, see "About Files in the Customization Package" on page 144.

2 To define a localizable string, add the following code to the package_res.xml file:

<str key="*string_id*">*localizable_string*</str>

where:

- *string_id* is an Id of the localizable string. The double quotes are required.
- *localizable_string* is the localizable string.

3 Use the localizable string Id in every location where Siebel CRM Desktop must display the string.

You must use different formats to define the string in different types of files. Use values from the following table.

File Type	Description

Any XML file except for the views.xml file.	Use the following format:
	#string_id
	For example:
	`<cell size="22">`
	`<static id="account_label" tab_order="6">`
	`<text>#lbl_account</text>`
	`</static>`
	`</cell>`

The views.xml file.	Use the following format:
	$string_id$
	For example:
	`<str key="all_accounts">`
	`<![CDATA[`
	`<?xml version="1.0"?> <view`
	`type="table"> <viewname>$view_siebel_accounts$</viewname>`
	`..... </view>`
	`]]>`
	`</str`

Any JavaScript file.	Use the following format: *session.res_string("string_id")* For example: `ui.message_box(0, session.res_string("msg_general_error"), session.res_string("msg_general_error_caption"), 0x40);`

Displaying a Custom Siebel CRM Field in a Siebel CRM Desktop Form

This topic gives one example of how to display a custom Siebel field in a Siebel CRM Desktop form. You might use this feature differently, depending on your business model. In this example, you display the Mail Stop field on the Contact form in the Siebel CRM Desktop client. You make this field available through the Siebel API and then customize Siebel CRM Desktop to synchronize and display the field. You modify the following files:

- siebel_meta_info.xml
- siebel_basic_mapping.xml
- forms_12.xml
- package_res.xml

For more information, see "About Files in the Customization Package" on page 144.

To display a custom Siebel CRM field in a Siebel CRM Desktop form

1. In Siebel Tools, make sure the Mail Stop field exists on the Contact business component.

   If it does not, then add it now.

2. Add the Mail Stop field to the CRMDesktopContactIO integration object.

   In order for Siebel CRM Desktop to synchronize data with the Siebel database, you use Siebel CRM Desktop integration objects and integration components to make the objects and fields that you use in this example available. The Contact object is already available but the Mail Stop field is not. To make the Mail Stop field available, you add it to the Contact integration component for each of the required integration objects:

   a. In the Object Explorer, click Integration Object, and then locate the CRMDesktopContactIO integration object in the Integration Objects list.

   b. In the Object Explorer, expand the Integration Object tree, click Integration Component, and then locate the Contact integration component in the Integration Components list.

   c. In the Object Explorer, expand the Integration Component tree, and then click Integration Component Field.

   d. In the Integration Component Fields list, add a new record using values from the following table.

Property	Value
Name	Mail Stop
Data Type	DTYPE_TEXT

3. Repeat Step 2 for the CRMDesktopAccountIO integration object.
4. Repeat Step 2 for the CRMDesktopOpportunityIO integration object.

5 Compile all locked projects.

After compiling finishes, the Mail Stop field is available through the API and you can configure Siebel CRM Desktop to use the field. For more information, see *Using Siebel Tools*.

6 Define the objects and fields to synchronize:

a Use an XML editor to open the siebel_meta_info.xml file.

For more information, see "About Files in the Customization Package" on page 144.

b In the siebel_meta_info.xml file, locate the following tag:

```
object TypeId='Contact'
```

Note that there are several child field tags that reside in the object TypeId='Contact' tag. These children define the fields for the Contact object.

c Add the following field tag as a child to the object TypeId='Contact' tag:

```
<field Name='Mail Stop' Label='Mail Stop' DataType='DTYPE_TEXT'
IOElemName='MailStop' />
```

d Repeat Step b and Step c for the following tag:

```
object TypeId='Account.Contact'
```

e Repeat Step b and Step c for the following tag:

```
object TypeId='Opportunity.Contact'
```

f Save and close the siebel_meta_info.xml file.

7 Map the Mail Stop field from the Contact object in the Siebel database to a field in Siebel CRM Desktop:

a Use an XML editor to open the siebel_basic_mapping.xml file.

For more information, see "About Files in the Customization Package" on page 144.

b In the siebel_basic_mapping.xml file, add a new field tag to the type tag using values from the following table.

Tag	Value
type id	Contact

c Add the following code to the tag you created in Step b.

```
<field id="Mail Stop">
  <reader class="mapi_user">
    <user_field id="sbl Mail Stop" ol_field_type="1"></user_field>
    <convertor class="string"></convertor>
  </reader>
```

```
        <writer class="Microsoft Outlook_user">
          <user_field id="sbl Mail Stop" ol_field_type="1"></user_field>
          <convertor class="string"></convertor>
        </writer>
      </field>
```

d Save and close the siebel_basic_mapping.xml file.

8 Insert a label and the Mail Stop field just below the Job Title field on the Contact form:

a Open the forms_12.xml file, and then locate the cell that contains the #lbl_job_title label control.

b Insert the following XML code immediately after the cell that contains the #lbl_job_title label control:

```
<cell size="22">
  <control id="lbl_MailStop" class="static" tab_order="6">
    <text>#lbl_mail_stop</text>
  </control>
</cell>
```

The following code specifies a key that the package_res.xml file uses to determine the localized value for the label:

```
#lbl_mail_stop
```

c Locate the section that is labeled with the following comment:

```
left side fields
```

This section resides in the tag that resides in the form that contains the SBL Contact ID.

d To add the text field control, insert the following XML code immediately above the cell that contains the ContactToAccount MVG control, and just below the cell that contains the status_image control:

```
<cell size="22">
  <control class="edit" id="MailStop" tab_order="7">
    <field value="string">Mail Stop</field>
  </control>
</cell>
```

e Locate the cell size tag, and then change it to the following value:

```
<cell size="185">
```

To make room for the new field, you must increase the cell size that contains all of the child objects. In this example, you change the cell size from 155 to 185.

9 Add the following code to the package_res.xml file:

```
<str key="lbl_mail_stop">Mail stop:</str>
```

Add this code as a child of the res_root tag under the following comment:

```
<!-Contact Form
```

This code provides localized values and images to the Siebel CRM Desktop Client. This code allows the Contact form to render the Mail Stop label through a key value. The package_res.xml file provides localized values and images to Siebel CRM Desktop. Because you added the new Mail Stop field to the Contact form, you must provide the text for the label. When you modified the forms_12.xml file, you created a label control that contains #lbl_mail_stop for the text value. This control identifies the key to use in the package_res.xml file.

10 Republish the updated package files.

During the next synchronization, Siebel CRM Desktop uses the updated files to apply the modifications to the Contact form. The Mail Stop field is available on the Contact form and Siebel CRM Desktop synchronizes the values in this field with the Siebel Server. For more information, see “Republishing a Customization Package” on page 127.

Making a Field in Siebel CRM Desktop Read-Only

To make a field read-only, you must use a file named ctx.form because it includes form controls that are mapped to corresponding fields. To make a field read-only, you disable the corresponding control on the form that references the field that you must make read-only.

In this example, you make the Opportunity Name on the opportunity form read-only.

To make a field in Siebel CRM Desktop read-only

1. Open the forms_xx.xml file.
2. Locate the id attribute for the control you must modify.

   In this example, locate the following code:

   ```
   <cell size="21">
      <edit id="opportunity" max_chars="100" tab_order="3">
         <field value="string">Name</field>
      </edit>
   </cell>
   ```

   This code controls the Opportunity Name dialog box that Siebel CRM Desktop displays on the Opportunity form. It includes the opportunity identifier.
3. Create a text file named ctx.form.
4. Add the following code to the file you created in Step 3:

   ```
   ctx.form[control_id].enabled = false;
   ```

   where:

   - *control_id* is an identifier of the control that you must make read-only.
   - *enabled* determines if the field is read-only. You can use one of the following values:
     - **True.** Makes the field editable.
     - **False.** Makes the field read-only.

   For this example, add the following code:

   ```
   ctx.form["opportunity"].enabled = false;
   ```
5. Open the Siebel CRM Desktop client and then navigate to the Opportunity form.
6. Open an opportunity and make sure you cannot modify the Opportunity Name.

Adding a Default Value to a Field in Siebel CRM Desktop

This topic describes how to configure Siebel CRM Desktop to add a default value to a field when the user creates a new record. In this example, you configure Siebel CRM Desktop to add the following default value to the Opportunity field:

CRM Opportunity

To add a default value to a field in Siebel CRM Desktop

1. Configure the default value:
   a. Use a JavaScript editor to open the business_logic.js.
   b. Locate the create_siebel_meta_scheme2 function.
   c. Add the following code to the end of the function:

      ```
      scheme.objects.get_object("object_type").get_field("field_name")["default_source"] = "default_value";
      ```

      where:

      - *object_type* is the object type identifier of the object type that resides in the siebel_basic_mapping.xml file. This file contains definitions for object types. Each definition includes an Id attribute. This attribute is the object type identifier that you must use for the object_type. For more information, see "Customizing How Siebel CRM Desktop Maps Fields" on page 148.
      - *field_name* is the name of a field that resides in the definition of the object type in the siebel_basic_mapping.xml file.
      - *default_source* identifies the source for the default value. For more information, see "Setting the Source for the Default Value" on page 165.
      - *default_value* defines the default value that Siebel CRM Desktop adds. You use the default_value variable only if you set the default_source variable to initial_value.

      For this example, you add the following code:

      ```
      scheme.objects.get_object("Opportunity").get_field("Name")["initial_value"] = "CRM Opportunity";
      ```

   d. Save and then close the business_logic.js file.
2. Test your work:
   a. Open the Siebel CRM Desktop client, and then navigate to the opportunity form.
   b. Create a new opportunity.
   c. Verify that Siebel CRM Desktop adds the following default value to the Opportunity Name field:

      CRM Opportunity

Setting the Source for the Default Value

To identify the source of the default value, you can use one of the following values for the default_source variable:

- **initial_value.** Specifies to use the value that you set for the default_value variable as the default value.
- **initial_value_res.** Sets the same value as the initial_value variable except you define the default value in the package_res.xml file.
- **initial_value_fn.** Uses a function that returns a value to the field. For example, to configure Siebel CRM Desktop to enter a value for a date field, you can specify a function that returns the current Windows system date.

Preventing the User From Deleting Records

This topic describes how to configure Siebel CRM Desktop to prevent the user from deleting records in the Siebel CRM Desktop client. You can also configure Siebel CRM Desktop to allow the user to delete a record in the Siebel CRM Desktop client and then confirm that deletion during synchronization. For more information, see "Controlling How Siebel CRM Desktop Deletes Records During Synchronization" on page 116.

To prevent the user from deleting records

1. In Siebel Tools, make sure the integration component object type is displayed.

   For more information, see "Displaying Object Types in Siebel Tools" on page 156.

2. In the Object Explorer, click Integration Object.
3. In the Integration Objects list, query the Name property for CRMDesktopContactIO, and then make sure the Object Locked property contains a check mark.
4. In the Object Explorer, expand the Integration Object tree, and then click Integration Component.
5. In the Object Explorer, expand the Integration Component tree, and then click Integration Component User Prop.
6. In the Integration Component User Prop list, add a new record with the following values.

Property	Value
NoDelete	Y

7. Repeat Step 2 through Step 6 for every CRMDesktop integration object.
8. Compile your changes.
9. Log in to the Siebel CRM Desktop client.
10. Delete a contact.

11 Perform a synchronization.

12 Make sure Siebel CRM Desktop displays a message that is similar to the following:

EAI Adapter call failed with error: No deletes are allowed in Integration Component Action_Contact (SBL-EAI-04183)

Process of Customizing Objects in Siebel CRM Desktop

This topic is a step in "Roadmap for Installing Siebel CRM Desktop" on page 77.

This topic gives one example of customize objects in Siebel CRM Desktop. You might use this feature differently, depending on your business model.

To customize objects in Siebel CRM Desktop, you do the following:

1 "Defining the Custom Object" on page 166

2 "Defining Synchronization for a Custom Object" on page 169

3 "Adding Custom Views in Microsoft Outlook" on page 171

4 "Defining the User Interface" on page 171

5 "Defining the Validation Rules" on page 179

6 "Defining Validation Rules for a Phone Number" on page 180

7 "Adding Custom Logic" on page 182

8 "Defining the Toolbar" on page 183

9 "Defining Other Options" on page 185

10 "Defining Logic for the Custom Form" on page 185

For more information about:

- Overview of XML files that you modify in this example, see Customizing the Siebel CRM Desktop Application on page 143.
- Details about tags in XML files that you modify in this example, see Appendix B, "XML Files Reference."
- Additional XML code that you use in this example, see "Additional Code in the Customization Example" on page 287.

Defining the Custom Object

This task is a step in "Process of Customizing Objects in Siebel CRM Desktop" on page 166.

In this topic, to add a new object to Microsoft Outlook, you describe the structure of the object, create mapping between fields, lists, and so forth. You make these customizations in the siebel_basic_mapping.xml file.

To define the custom object

1 Use an XML editor to open the siebel_basic_mapping.xml file.

For more information, see "About Files in the Customization Package" on page 144.

2 To define the name of the custom object, add the following example code to the siebel_basic_mapping.xml file:

```
<type id="Action" display_name="#obj_activity_plural" folder_type="10">

<form message_class="IPM.Contact.SBL.Activity" icon="type_image:Event:16"
large_icon="type_image:Event:32" display_name="Activity">SBL Activity</form>

</type>
```

For more information, see "Example Code for the SBL Activity Form" on page 168.

3 Define a set of fields for the custom object. The following table describes the fields you define for this example.

Field Label	Field Name	Field Type
Description	Description	Text
Type	Type	Picklist
Priority	Priority	Picklist
Owner	Primary Owner Id	Lookup
Account	Account Id	Lookup
Opportunity	Opportunity Id	Lookup
Contacts	No field on this object	MVG
Employee	No field on this object	MVG
Planned Start	Planned	datetime
Planned Completion	Planned Completion	datetime
Due	Due	datetime
Status	Status	Picklist
Comments	Comment	Textarea

For more information, see "Fields That Siebel CRM Desktop Uses for the Custom Object" on page 168. To examine the code you must add for this step, see "XML Code That Defines a Set of Custom Fields on page 287" on page 287.

4 Define intersection objects for many-to-many relationships.

You do not define many-to-many relationships in Step 3. Because there is a many-to-many relationship between contacts and activities, and between employees and activities, you must define more objects that contain links to activity and contact, or activity and employee. The remaining description for a many-to-many relationship is the same as for other objects where you specify the object name and object fields. This object can also contain a field that indicates if this intersection record is a primary or not a primary.

To examine the code you must add for this step, see "XML Code That Defines a Many-To-Many Relationship" on page 289.

5 Define the lists.

For more information, see "Defining the List" on page 168. To examine the code you must add for this step, see "XML Code That Defines a List" on page 291.

Example Code for the SBL Activity Form

The example code defines SBL Activity as the form to display for this object. You define the form layout later. In this example, you add an Activity object to Microsoft Outlook. In Siebel CRM, this object is named Action. To add this object to Microsoft Outlook, you modify the siebel_basic_mapping.xml file. Note the following:

- You add the type tag to describe the new object.
- To define the folder name for the object in Microsoft Outlook, you can use the display_name attribute of the type tag.
- To define the native Microsoft Outlook object that is the base for the custom object folder type, you can use the folder_type attribute.

Fields That Siebel CRM Desktop Uses for the Custom Object

The siebel_basic_mapping.xml file describes each of the fields for the custom object. Note that Siebel CRM Desktop maps each field to a custom field, except for the following fields:

- Siebel CRM Desktop maps the Description field to the Last Name field, which is a native field in Microsoft Outlook.
- Siebel CRM Desktop maps the Comment field to the Body field because Siebel CRM Desktop does not support the textarea field. Therefore, Siebel CRM Desktop uses the native Microsoft Outlook control that displays the value for the Body field.

Defining the List

Because the custom object stores items that the user chooses in a list, you describe the list field in the same way as you describe a string field. You must describe the object that stores all list values. To store the values of the list, each list uses a separate object. You must build the IDs for these objects according to the following rules:

- Object name and field
- Name and list

You must make sure the Type list on the Activity object includes the ID of the ActionTypePicklist object. To define a list object, you must define the following set of standard fields:

- Label
- Value (string)
- SortOrder (integer)
- IsDefault (bool)

Defining Synchronization for a Custom Object

This task is a step in "Process of Customizing Objects in Siebel CRM Desktop" on page 166.

In this topic, you define a custom object so that Siebel CRM Desktop can synchronize it with a Siebel CRM object.

To define synchronization for a custom object

1. Use an XML editor to open the connector_configuration.xml file.

   For more information, see "About Files in the Customization Package" on page 144.

2. Define the Links section.

   The Links section contains a set of fields that reference other objects. You must define these references to allow the Synchronization Engine to synchronize objects in the correct order and to download the related items. To define the links, you add the following XML code:

```
<type id="Action">

      <view label="Activity" label_plural="Activities"
small_icon="type_image:Event:16" normal_icon="type_image:Event:24"
large_icon="type_image:Event:48"></view>

      <synchronizer name_format=":[:(Description):]">

         <links>

            <link>Account Id</link>

            <link>Opportunity Id</link>

            <link>Primary Owner Id</link>

         </links>

      </synchronizer>

   </type>
```

3 Define the deduplication keys.

Because the business environment for this example requires that activities are the same if their descriptions are the same, one natural key is defined. That key is Description. Add the following XML code to the synchronizer tag:

```
<natural_keys>
 <natural_key>
    <field>Description</field>
 </natural_key>
</natural_keys>
```

4 Add the following descriptions to the connector_configuration.xml file for the intersection records that you defined for contacts and employee in Step 4 on page 168:

```
<type id="Action.Employee.Association">
      <view label="Activity Employee" label_plural="Activity Employees"
small_icon="type_image:Generic:16" normal_icon="type_image:Generic:24"
large_icon="type_image:Generic:48" suppress_sync_ui="true"></view>
      <synchronizer name_format=":[:(UserName) :]">
         <links>
            <link>EmployeeId</link>
            <link>ActionId</link>
         </links>
      </synchronizer>
   </type>
   <type id="Action.Contact.Association">
      <view label="Activity Contact" label_plural="Activity Contacts "
small_icon="type_image:Generic:16" normal_icon="type_image:Generic:24"
large_icon="type_image:Generic:48" suppress_sync_ui="true"></view>
      <synchronizer name_format=":[:(ContactName) :]">
         <links>
            <link>ActionId</link>
            <link>ContactId</link>
         </links>
      </synchronizer>
   </type>
```

Adding Custom Views in Microsoft Outlook

This task is a step in "Process of Customizing Objects in Siebel CRM Desktop" on page 166.

After you define a folder in Microsoft Outlook that displays the custom object, you define the Microsoft Outlook views that apply to this folder.

To add custom views in Microsoft Outlook

1. Use an XML editor to open the siebel_basic_mapping.xml file.

   For more information, see "About Files in the Customization Package" on page 144.

2. Add the following custom_views section to the Activity type tag:

```
<type id="Action" display_name="#obj_activity_plural" folder_type="10">

   <form (…) >SBL Activity</form>

   <custom_views default_name="Siebel Activities">

      <view id="all_activities" name="Siebel Activities"></view>

      <view id="all_activities_by_owner" name="Siebel Activities by Primary
Owner"></view>

      <view id="all_activities_by_priority" name="Siebel Activities by
Priority"></view>

   </custom_views>
```

3. Make sure the views.xml file describes the views that you define in Step 2.

Defining the User Interface

This task is a step in "Process of Customizing Objects in Siebel CRM Desktop" on page 166.

In this topic, you define the user interface for the custom object. To allow the user to work with the custom object, you must define the form that Siebel CRM Desktop uses to display the custom object. The custom object is configured to use the custom SBL Activity form in the siebel_basic_mapping.xml file that you modified in "Defining the Custom Object" on page 166. You must create the SBL Activity form.

Figure 10 illustrates the layout of the Activity Form.

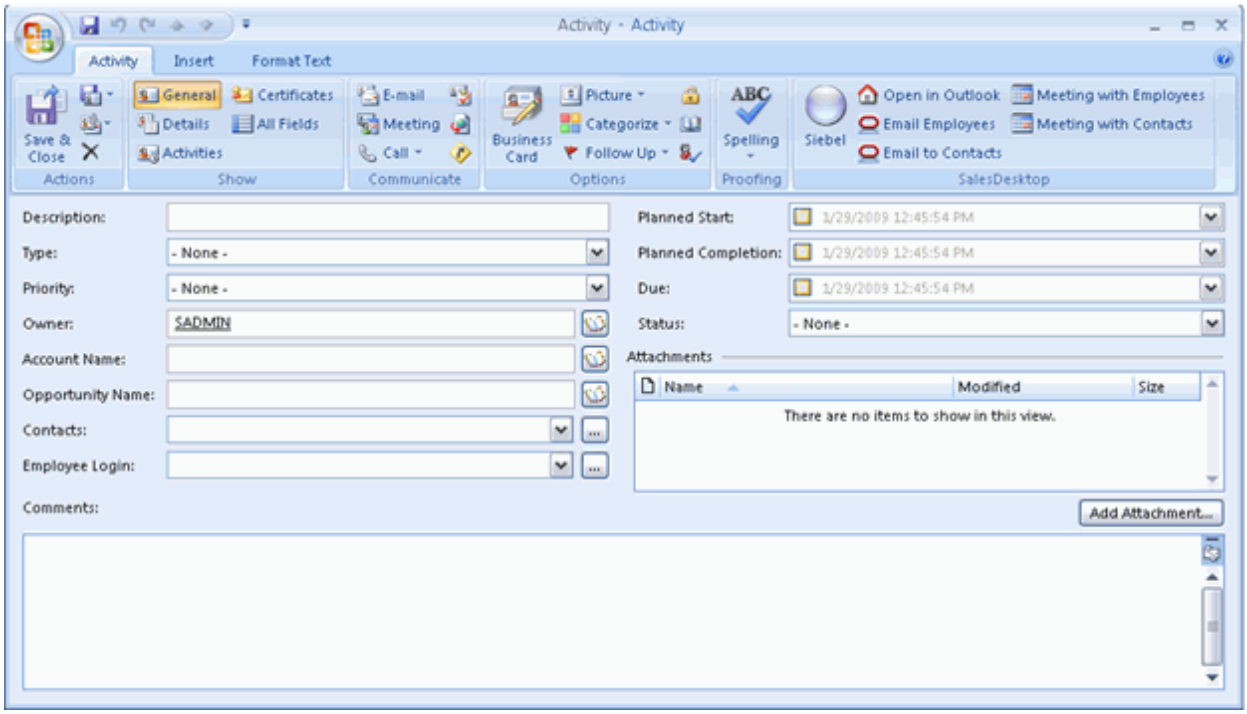


Figure 10. Layout of the Activity Form

A set of *cells* describes the form layout. A cell can be empty or it can contain a control or a *stack*. A single stack can contain numerous cells. The form is divided into the following parts:

- One part includes visible controls.
- One part includes hidden controls.

This example uses this technique because the example uses native Microsoft Outlook forms as a base for custom forms. Although you can modify the position of a native Microsoft Outlook control, you cannot remove it entirely from the form. You move the unused controls that are native to Microsoft Outlook to a location where the user cannot view them, which is typically a very small cell.

This form is a prototype that helps you visualize the cells, stacks of cells, and the order in which you use the cells and stacks. This visualization helps to reduce the wide range of combinations of cells and stacks that you can describe down to only those cells and stacks that you can support. You can add new fields, remove fields, reorder fields, and apply any other changes during development and testing.

To define the user interface

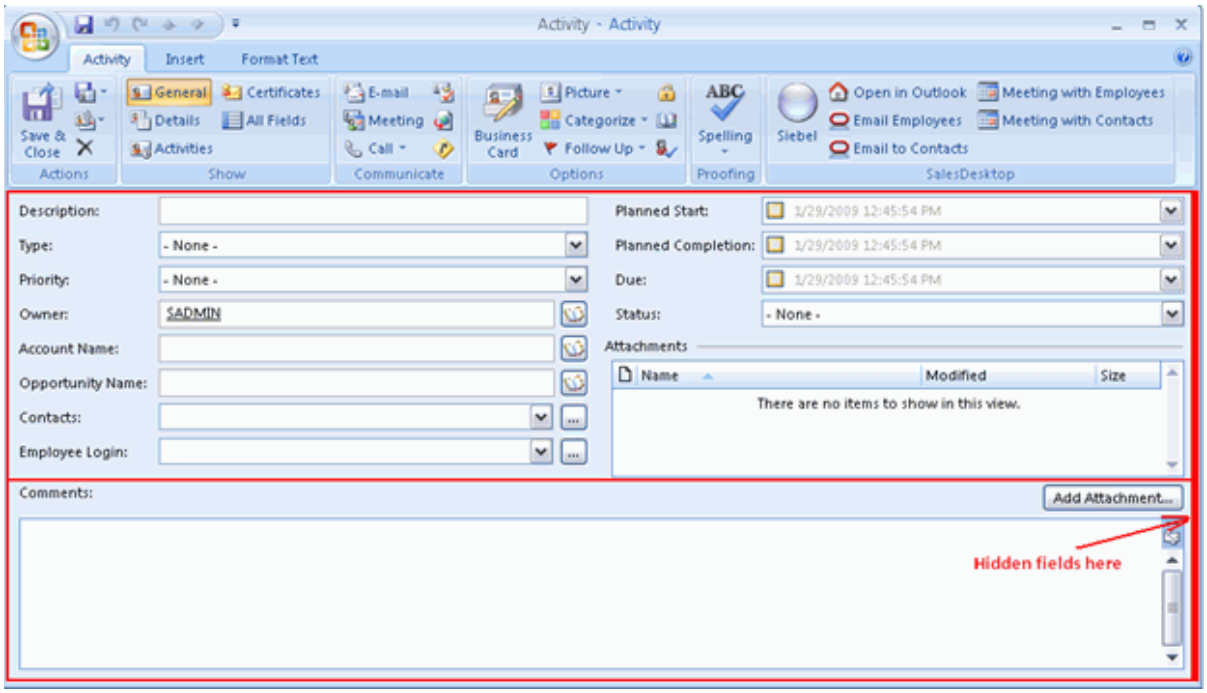
1 Use an XML editor to open the forms_12.xml file.

For more information, see "About Files in the Customization Package" on page 144.

2 To divide the form into a visible section and a hidden section, add the following code to the forms_12.xml file:

```
<form id="SBL Activity">
   <page id="General" tag="0x10A6" min_height="335" min_width="520">
      <cell>
         <stack layout="horz" padding="10">
         <!-- visible section -->
            <cell>
            <!-visible fields here -->
            </cell>
         <!-- hidden section -->
            <cell size="1">
            <!-hidden fields here -->
            </cell>
         </stack>
      </cell>
   </page>
</form>
```

3 Divide the visible section into a top section and a bottom section, as illustrated in the following figure:

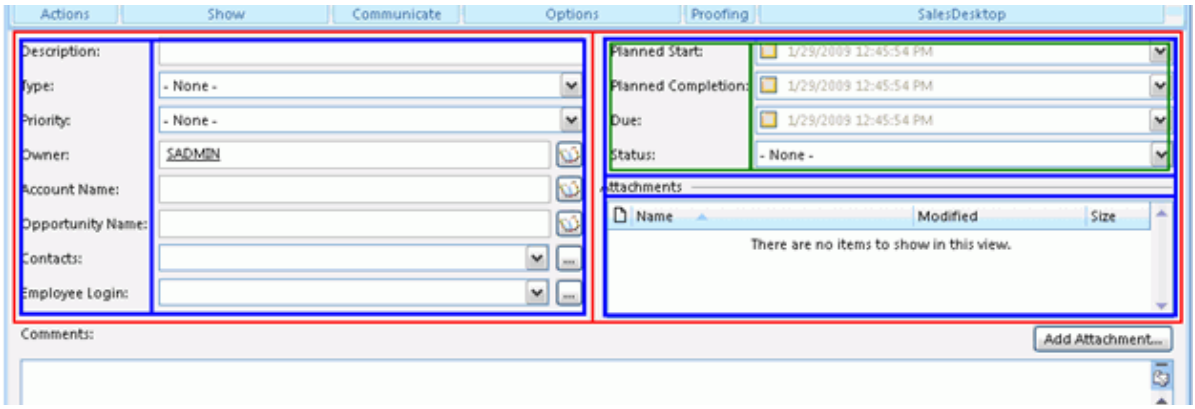


Although not required, this step helps to support the current layout of the form. Add the following code:

```
<form id="SBL Activity">
(……)
   <!-- visible section -->
      <cell>
         <stack layout="vert" padding="5" spacing="5">
            <!-top section-->
            <cell size="220">
               <!-top section fields here-->
            </cell>
            <!-bottom section-->
            <cell>
               <!-bottom section fields here-->
            </cell>
```

```
            </stack>
         </cell>
      (..............)
      </form>
```

4 Divide the top section into two parts that include a left column and a right column of fields, as illustrated in the following figure:

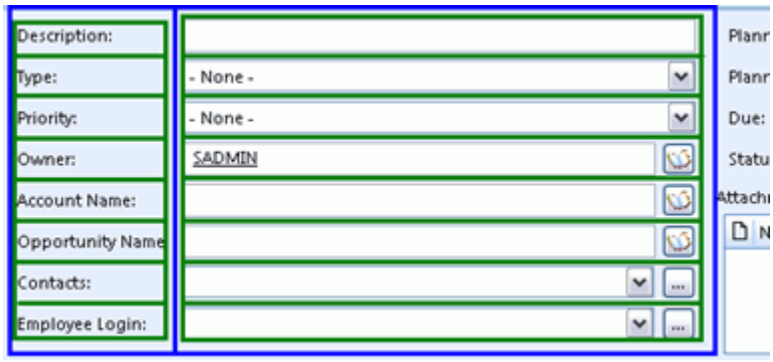


Add the following code:

```
<-- top section -->
<cell size=220>
 <stack layout="horz" spacing="5">
    <cell>
       <stack layout="horz" spacing="3">
       <!-- left side captions -->
          <cell size="105">
          </cell>
       <!-- left side fields -->
          <cell>                          </cell>
       </stack>
    </cell>
    <cell>
       <stack layout="vert" spacing="5">
          <cell size="105">
```

```
            <stack layout="horz" spacing="3">
            <!-- left side captions -->
               <cell size="110">
               </cell>
            <!-- left side fields -->
               <cell>
               </cell>
            </stack>
         </cell>
         <cell size="13">
            <!-attachments caption and separator here -->
         </cell>
         <cell>
            <!-attachments view here -->
         </cell>
      </stack>
   </cell>
 </stack>
</cell>
```

5 Divide the left column into cells where you can place captions for fields, as illustrated in the following figure:

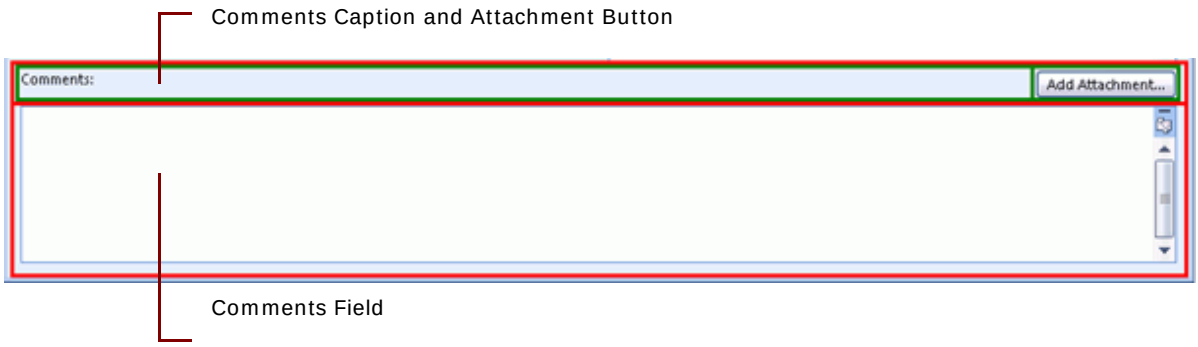


For more information, see “XML Code That Creates Cells” on page 293.

6 Divide the right column into cells where you can place fields.

Because the code that you use to complete this step is similar to the code you use in Step 5, you can copy and modify the code that is described in "XML Code That Creates Cells" on page 293.

7 Define the Comments section of the Activity form, as illustrated in the following figure:



The Comments section is divided into the following parts:

- Comments caption and the Add Attachment button
- Comments field

To hold the caption and the button, the top part includes two parts. Another section describes the Add attachment button. It is not described in the section that it relates to according to the logic of the button. This is not important in this step because you describe the form layout, not the button logic. For more information, see "Defining Logic for the Custom Form" on page 185.

Add the following code:

```
<!-- visible section -->
   <cell>
      <stack layout="vert" padding="5" spacing="5">
      <!-top section-->
         <cell size="220">
         <!-top section fields here-->
         </cell>
         <!-bottom section-->
         <cell size="22">
            <stack layout="horz">
               <cell>
                  <control id="0x20022" class="static" tab_order="30">
```

```
                    <text>Comments:</text>
                  </control>
                </cell>
                <cell size="110" attraction="far">
                  <control class="button" id="0x20059" tab_order="29"
on_click="LinkAttachment()">
                                                        <text>Add attachment…t</
text>
                                                    </control>
                </cell>
                <cell size="5" attraction="far"></cell>
              </stack>
            </cell>
            <cell>
              <control id="0x103f" tab_order="31"></control>
            </cell>
          </stack>
        </cell>
```

8 Make sure the description you create supports the version of Microsoft Outlook that your users use.

For more information, see "Correct Usage of the Forms_xx.XML File and Object ID" on page 178.

Correct Usage of the Forms_xx.XML File and Object ID

Microsoft Outlook 2003 and Microsoft Outlook 2007 describe forms differently. These versions use different IDs for native Microsoft Outlook controls. You must create a different description of the form for each version that your implementation uses. Each form uses the same layout description but the native Microsoft Outlook control ID is different.

For example, assume you create code for Microsoft Outlook 2007. For this code to work correctly with Microsoft Outlook 2003, you must replace object IDs that are native in Microsoft Outlook 2007 with object IDs that are native in Microsoft Outlook 2003. In the example in this topic, you must replace the IDs that are in the hidden section and in the ID of the body control.

Because the code in this topic supports Microsoft Outlook 2007, you must add it to the forms_12.xml file. If you use Microsoft Outlook 2003, then you use the forms_11.xml file and you use Object IDs that are native to Microsoft Outlook 2003.

Defining the Validation Rules

This task is a step in "Process of Customizing Objects in Siebel CRM Desktop" on page 166.

In this topic, you add data validation rules to make sure the user enters information in the form correctly. For this example, assume the business requirements include the following for an activity:

- The Owner field must contain an owner.
- The Planned Start field must contain a date.
- The date in the Planned Complete field must not occur earlier than the date in the Planned Start field.

In this topic, you use the forms_12.xml file. For more information, see "Correct Usage of the Forms_xx.XML File and Object ID" on page 178.

To define validation rules

- To create a validation_rules section for the description of your form, add the following code to the forms_12.xml file:

```
<form id="SBL Activity">

   <validation_rules>

      <rule message="Owner is required.">

         <expression>

            <![CDATA[item["Primary Owner Id"] != null]]>

         </expression>

         <asserted_control id="Owner"></asserted_control>

      </rule>

      <rule message="Planned Start is required" expression="Planned != null">

         <asserted_control id="0x20017"></asserted_control>

      </rule>

      <rule message="#msg_activity_planned_complete_validation">

         <expression>

            <![CDATA[

               item['Planned Completion']!=null ? item['Planned Completion'] >=
               item['Planned'] : true;

            ]]>

         </expression>

         <asserted_control id="PlannedCompletion"></asserted_control>
```

```
    </rule>
</validation_rules>
```

Defining Validation Rules for a Phone Number

To define validation rules for the phone number in a contact record or an account record, you can add the following regular expression to the forms_xx.xml file:

```
var phonetest = new RegExp(/\(?[0-9]{3}\)?[-. ]?[0-9]{3}[-. ]?[0-9]{4}/);

item["Work Phone #"] != '' ? (item["Work Phone #"].match(phonetest) != null ? true
: false) : true;
```

where:

- *RegExp* describes the regular expression. The part of the phone number that the user enters must match the pattern that you defined in the RegExp expression. If the number contains fewer than 10 digits, then Siebel CRM Desktop creates a validation failure message.
- *match(phonetest)* causes Siebel CRM Desktop to search for the substring. The real value must not be the same as the pattern.

In this example, the following phone numbers are valid:

- 1111111111
- (999)-888-4444
- 222.888.9999

The following phone numbers are not valid:

- 1111111111 - x123
- phone: 1111111111
- +1 555.666.8888
- 11111111
- 222.aaa.8888

To define another validation, you can specify another regular expression in the rule tag of the forms_xx.xml file.

This number format is specific to telephone networks in the United States. For a different country, you must use the numbering format that is specific to that country.

To define validation rules for a phone number

1. Use an XML editor to open the forms_xx.xml file.

   For more information, see "About Files in the Customization Package" on page 144.

2. Add the following code to the forms_xx.xml file:

```
<rule message="#msg_business_phone_validation">
  <expression>
     <![CDATA[
var phonetest = new RegExp(/\(?[0-9]{3}\)?[-. ]?[0-9]{3}[-. ]?[0-
9]{4}/);
item["Work Phone #"] != '' ? (item["Work Phone #"].match(phonetest)
!= null ? true : false) : true;
     ]]>
  </expression>
</rule>
<rule message="#msg_mobile_phone_validation">
  <expression>
     <![CDATA[
var phonetest = new RegExp(/\(?[0-9]{3}\)?[-. ]?[0-9]{3}[-. ]?[0-
9]{4}/);
item["Cellular Phone #"] != '' ? (item["Cellular Phone
#"].match(phonetest) != null ? true : false) : true;
     ]]>
  </expression>
</rule>
<rule message="#msg_home_phone_validation">
  <expression>
     <![CDATA[
var phonetest = new RegExp(/\(?[0-9]{3}\)?[-. ]?[0-9]{3}[-. ]?[0-
9]{4}/);
item["Home Phone #"] != '' ? (item["Home Phone #"].match(phonetest)
!= null ? true : false) : true;
     ]]>
  </expression>
</rule>
```

```
<rule message="#msg_fax_phone_validation">
  <expression>
     <![CDATA[
var phonetest = new RegExp(/\(?[0-9]{3}\)?[-. ]?[0-9]{3}[-. ]?[0-
9]{4}/);
item["Fax Phone #"] != '' ? (item["Fax Phone #"].match(phonetest) !=
null ? true : false) : true;
     ]]>
  </expression>
</rule>
```

Adding Custom Logic

This task is a step in "Process of Customizing Objects in Siebel CRM Desktop" on page 166.

To improve usability, sometimes you must add custom logic to Siebel CRM Desktop. Siebel CRM Desktop allows you to manipulate Microsoft Outlook data. For example:

- Automatically include the ID of the current user in the Owner field.
- Cause the first letter of each word in the description of an activity to automatically capitalize.
- Change the state of some controls. For example, to inform the user that the control is required.

Siebel CRM Desktop uses JavaScript to handle custom manipulation of Microsoft Outlook data. Because Siebel CRM Desktop must run this JavaScript on certain events, you must define the exact events that are involved and specify the code that Siebel CRM Desktop must call at each event. To do this, you add the following attributes to the form tag:

```
<form id="SBL Activity" on_open="form_open()" on_saving="form_saving()">
```

Siebel CRM Desktop does the following work:

- If the user opens the form, then Siebel CRM Desktop calls the form_open function.
- If the user saves the form, then Siebel CRM Desktop calls the form_saving function.

You define these functions in the forms_12.xml file. For more information, see "Correct Usage of the Forms_xx.XML File and Object ID" on page 178.

To add custom logic

1. Use an XML editor to open the forms_12.xml file.

   For more information, see "About Files in the Customization Package" on page 144.

2. To add the custom logic for this example, add the following code:

```
<form id="SBL Activity" on_open="form_open()" on_saving="form_saving()"">
    <validation_rules>
       (……..)
    </validation_rules>
    <script>
       <![CDATA[
       function form_open()
       {
          //this makes red border for these controls to inform users that
          //these fields are required
          form.Type.required = true;
          form["0x20017"].required = true;

          //this code pre-fill Owner field by Current user Id
          var defaults = find_item("::Defaults", create_criteria("or"));
          if (form.item.snapshot["Primary Owner Id"] != null)
             form.item["Primary Owner Id"] = defaults.CurrentUser;
       }
       function form_saving()
       {
          //this makes all first letters capitalized
          var fields = form.item.snapshot;
          form.item.Description = fields. Description.replace(/\b[a-z]/
g,function(w){return w.toUpperCase()});
       }
       ]]>
    </script>
```

Defining the Toolbar

This task is a step in "Process of Customizing Objects in Siebel CRM Desktop" on page 166.

For this example, it is desirable to implement some actions for the custom object on a toolbar. An action can be simple, such as attaching a note to the custom object. An action can be more complicated, such as sending an email to all contacts that are related to the custom object. In this example, you add the following buttons to the toolbar:

- Add Open in Siebel CRM
- Add attachment to the toolbar of the form
- Add New Activity to the Siebel CRM Desktop toolbar

To implement these changes, you modify the toolbars.xml file. Because this file typically already contains definitions for the Siebel CRM Desktop and the form toolbars, you define only the custom buttons.

To define the toolbar

1. Encode the icon that represents the new button:
   a. Create a PNG file that includes the icon that represents the button in the Siebel CRM Desktop client.
   b. Encode the PNG file for Base64.

      Because Siebel CRM Desktop stores the icon in the resource file in a Base64 encoded string, you must encode the PNG file. You can use any standard Base64 encoder, such as base64.exe.
   c. Remove the line breaks from the contents of the output file you created in Step b.
   d. Add the contents of the output file you created in Step b to the package_res.xml file.
2. To define a new button for the Siebel CRM Desktop toolbar, add the following code to the toolbars.xml file:

```
<toolbar caption="Siebel CRM Desktop" for="explorer">

  (…….)

  <button name="New Activity" small_image="type_image:Activity:16">

     <action class="create_item" item_type="Action"/>

  </button>

  (…….)

</toolbar>
```

3. To define a new button for the form toolbar, add the following code to the toolbars.xml file. Note that the code uses the term *inspector* to reference the form:

```
<toolbar caption="Siebel CRM Desktop" for="inspector">

  (…….)

  <button name="Attach File" small_image="attach_btn_img">
```

```
    <action class="create_attachment" accept_type="Action">
       <attachment type="Attachment" name_field="Name" body_field="Body"
linking_field="ParentId"/>
    </action>
  </button>
  (…….)
</toolbar>
```

Defining Other Options

This task is a step in "Process of Customizing Objects in Siebel CRM Desktop" on page 166.

You can define other options.

To define other options

1. Define a support email address.

   For more information, see "Customizing the Email Address of the Support Team" on page 154.

2. Define setup options.

   For more information, see "Options for Installing the Siebel CRM Desktop Add-In" on page 92.

Defining Logic for the Custom Form

In this topic, you define logic for the custom form.

To define logic for the custom form

1. Use a JavaScript editor to open the forms.js file.
2. Register the controls you added.
3. Register the view control with the corresponding button.

Process of Adding a Predefined Siebel Picklist to Siebel CRM Desktop

This topic gives one example of how to add a predefined Siebel picklist to Siebel CRM Desktop. You might use this feature differently, depending on your business model. In this example, you add a picklist to the Contact form that allows the user to choose from a set of predefined contact methods. These methods indicate how the contact prefers to be contacted, such as through email, pager, or phone.

To add a predefined Siebel picklist to Siebel CRM Desktop, you do the following:

1 "Identifying Predefined Picklist Objects in Siebel CRM" on page 186.

2 "Creating an Integration Object for the Contact Method Picklist" on page 188

3 "Extending an Integration Object for the Contact Method Picklist" on page 189

4 "Adding a Field to the Customization Package" on page 190

5 "Customizing the Physical Layout for the Pick List" on page 198

6 "Publishing and Testing a Custom Picklist" on page 202

Identifying Predefined Picklist Objects in Siebel CRM

This task is a step in "Process of Adding a Predefined Siebel Picklist to Siebel CRM Desktop" on page 185.

In this topic, you identify the picklist objects that you use to add a picklist. These object come predefined with Siebel CRM.

To identify predefined picklist objects in Siebel CRM

1 Identify the field that Siebel CRM associates with the picklist you must add:

   a Open Siebel Call Center.

   b Navigate to the Contacts list, and then click a name in the Last Name field.

   c Click the More Info tab, and then click the down arrow in the Contact Method field.

      Siebel CRM displays the pick list for the Contact Method field. This is the pick list you customize in this example.

   d Choose the Help menu, and then the About View menu item.

      The About View dialog box lists the applets in the order in which Siebel Call Center displays them.

   e Note the applet name in which Siebel Call Center displays the Contact Method pick list.

      In this example, this is the Contact Form Applet - Child applet.

   f In Siebel Tools, in the Object Explorer, click Applet.

   g In the Applets list, query the Name property for the applet you noted in Step e, which is Contact Form Applet - Child.

   h Right-click the Contact Form Applet - Child applet, and then choose Edit Web Layout.

      If Siebel Tools displays the Read-only Object dialog box, then click OK. In this task, you do not modify the form so you can use a read-only version.

   i In the Applet Web Template editor, click the Contact Method control.

j In the Properties window, note the value for the following property.

Property	Value
Field	Preferred Communications

In this example, Siebel CRM associates the Preferred Communications field with the Contact Method pick list.

2 Identify the pick list:

a Open Siebel Tools.

b In the Object Explorer, click Business Component.

c In the Business Components list, query the Name property for Contact.

d In the Object Explorer, expand the Business Component tree, and then click Field.

e In the Fields list, query the Name property for Preferred Communications, and then note the value for the following property.

Property	Value
PickList	Comm Media Picklist

3 Identify the business component that the pick list references:

a In the Object Explorer, click Pick List.

b In the Picklists list, query the Name property for Comm Media Picklist, and then note the value for the following property.

Property	Value
Business Component	PickList Hierarchical Sub-Area

4 Identify the parent business object of the business component that the pick list references:

a In the Object Explorer, click the Flat tab, and then click Business Object Component.

b In the Business Object Components list, query the Bus Comp property for PickList Hierarchical Sub-Area, and then note the value for the following property.

Property	Value
Parent Business Object	CommSrv CM Channel Type PickList Administration

If your query does not return a result, then create a new business object component using values from the table in this step. If you create a new business object component, then make sure the business component in which it resides is the primary business component for the business object that it references.

Creating an Integration Object for the Contact Method Picklist

This task is a step in "Process of Adding a Predefined Siebel Picklist to Siebel CRM Desktop" on page 185.

In this topic, you create an integration object for the Contact Method picklist.

To create an integration object for the Contact Method picklist

1. In Siebel Tools, make sure the integration component object type is displayed.

   For more information, see "Displaying Object Types in Siebel Tools" on page 156.

2. Choose the File Menu, and then the New Object menu item.
3. Click the EAI tab, click Integration Object, and then click OK.
4. In the Integration Object Builder Dialog box, choose values for the following items, and then click Next.

Property	Value
Project	Choose a project. It is recommended that you create a separate project for any customization you make to Siebel CRM Desktop. For example, use a project named Siebel CRM Desktop.
Business Service	EAI Siebel Wizard

5. Choose values for the following items, and then click Next.

Property	Value
Source Object	CommSrv CM Channel Type PickList Administration This value is the name of the business object.
Source Root	PickList Hierarchical Sub-Area This value is the name of the business component.
Integration Object Name	CRMDesktopPickListHierarchialSubArea This value is the name of the integration object you are creating.

6. Accept the default values, click Next, and then click Finish.

7 In the Object Explorer, click Integration Object, query the Name property for CRMDesktopPickListHierarchialSubArea, and then note the following property of the integration object you created in Step 6.

Property	Value
XML Tag	ListOfCrmdesktoppicklisthierarchialsubarea

Extending an Integration Object for the Contact Method Picklist

This task is a step in "Process of Adding a Predefined Siebel Picklist to Siebel CRM Desktop" on page 185.

In this topic, you extend the Contact integration component. Because the Contact integration component is included in multiple locations, you must extend it in each of the following integration objects:

- CRMDesktopContactIO
- CRMDesktopAccountIO
- CRMDesktopOpportunityIO

To extend an integration object for the Contact Method picklist

1 In Siebel Tools, make sure the integration component object type is displayed.

For more information, see "Displaying Object Types in Siebel Tools" on page 156.

2 In the Object Explorer, click Integration Object.

3 In the Integration Objects list, query the Name property for CRMDesktopContactIO, and then make sure the Object Locked property contains a check mark.

4 In the Object Explorer, expand the Integration Object tree, and then click Integration Component.

5 In the Integration Components list, query the Name property for Contact.

6 In the Object Explorer, expand the Integration Components tree, and then click Integration Component Field.

7 In the Integration Component Fields list, add a new record with the following values.

Property	Value
Name	Preferred Communications
External Name	The value for each of these properties must match the field name on the Contact business component. In this example, Preferred Communications is the field you must reference. This is the value you noted in Step j on page 187.

Property	Value
Length	30 The value for this property must match the value that is set in the Length property of the Preferred Communications field.
Physical Data Type	DTYPE_TEXT
External Data Type	The value for each of these properties must match the value that is set in the Type property of the Preferred Communications field.
External Sequence	500 For more information, see "Requirements for the Sequence Property" on page 190.
XML Sequence	This value must equal the value in the External Sequence property. In this example, that value is 500.
XML Tag	PreferredCommunications The value for this property must match the field name on the Contact business component but with the spaces removed. This is the value you noted in Step j on page 187.

8 Repeat Step 3 through Step 7 for the CRMDesktopAccountIO integration object.

9 Repeat Step 3 through Step 7 for the CRMDesktopOpportunityIO integration object.

10 Compile your changes.

For more information, see *Using Siebel Tools*.

Requirements for the Sequence Property

You can enter any numeric value in the sequence property. Example properties include External Sequence and XML Sequence. This value must be unique. It must not display in the same sequence property for any other integration component in the Opportunity integration object.

Adding a Field to the Customization Package

This task is a step in "Process of Adding a Predefined Siebel Picklist to Siebel CRM Desktop" on page 185.

In this topic you add a field to the customization package.

To add a field to the customization package

1 Create a working set of files for the customization package:

a Open a DOS command prompt, and then navigate to the directory that contains the current files of the customization package.

For more information, see "About Files in the Customization Package" on page 144.

b Create a copy of the current set of customization package files.

c Move the original set of files to a backup directory.

If necessary, to restore the configuration that existed before you started this customization effort, you can revert to this backup set of files.

d To create a working set of customization package files, rename the set of files you copied in Step b.

For example, enter the following command:

```
rename v01* v02*
```

This command renames the prefix for all files in the directory that currently use v01 as the prefix. For example, it renames v01_forms_12.xml to v02_forms_12.xml. It is recommended that you use this technique to indicate that you have modified the customization package.

2 Verify that Siebel Tools added the integration object:

Use an XML editor to open the siebel_meta_info.xml file.

For more information, see "About Files in the Customization Package" on page 144.

a To locate the PickList_Preferred_Communications object, search for the following code:

```
<object TypeId="PickList_Preferred_Communications"
```

b In the header of the PickList_Preferred_Communications object, make sure the following attributes exist and with the correct value.

Attribute	Value
IntObjName	CRMDesktopPreferredCommPickList
SiebMsgXmlElemName	PicklistHierarchicalSub-Area
SiebMsgXmlCollectionElemName	ListOfCrmdesktoppreferredcommpicklist

3 To define the picklist, add the following element to the siebel_meta_info.xml file:

```
<picklist TypeId='PickList_Preferred_Communications' CollectionTypeFld
Name='Type' SrcObjectTypeId='PickList_Preferred_Communications'
ValueFldName='Value' LabelFldName='Value' LangFldName='Language' >

 <master_filter_expr>

   <![CDATA]

      [Parent] = LookupValue ('OFFER_MEDIA', 'Package')

 ]]>

 </master_filter_expr>
```

```
</picklist>
```

For more information, see "Defining Attributes of the Pick List Element" on page 193.

4 Add the Preferred Communications field to the Contact object:

a To locate the Contact object, search for the following code:

```
object TypeID='Contact'
```

b Add the following code to the Contact object:

```
<field Name="Preferred Communications" Label="Preferred Communications"
DataType="DTYPE_TEXT" HasPicklist="yes" PicklistIsStatic="yes"
PicklistCollectionType="OFFER_MEDIA" PicklistTypeId="PickList Preferred
Communications" IOElemName="PreferredCommunications" />
```

5 Repeat Step 4 for each of the following objects:

- Account.Contact
- Opportunity.Contact

In this example, these objects in the siebel_meta_info.xml file must include the Preferred Communications field. You must add this field to each object.

6 Add code that creates a map for the pick list between the Siebel Server and Siebel CRM Desktop for the parent Contact object:

a Use an XML editor to open the siebel_basic_mapping.xml file.

b Create a new object type for the pick list.

For more information, see "Code to Create a New Object Type for the Pick List" on page 193

c To locate the parent object, search the siebel_basic_mapping.xml file for the following code:

```
<type id="Contact"
```

d Add code to the Contact object that defines a map between the Siebel Server and Siebel CRM Desktop.

For more information, see "Code to Define a Map Between the Siebel Server and Siebel CRM Desktop" on page 195.

7 Add code that creates a map for the pick list between the Siebel Server and Siebel CRM Desktop for the child Account Contacts object:

a Choose the code from the contact object that you use to map the child account object.

For more information, see "Mapping Child Objects for a Custom Picklist" on page 196.

b Copy this code to the clipboard.

c To locate the Account Contacts child object, search the siebel_basic_mapping.xml file for the following code:

```
type id="Account.Contact.Association"
```

d To locate the Contact field, search in the Account.Contact.Association object for the following text:

```
field id="ContactId"
```

e To locate the matching ContactId, search the ContactId field for the following text:

```
user_field id="sbl Contact ID"
```

f Paste the contact fields that you copied to the clipboard in Step a into the following user field:

```
sbl Contact ID
```

For more information, see "Mapping Child Objects for a Custom Picklist" on page 196.

g To map the Preferred Communications field, add the following code immediately after the code you pasted in Step f:

```
<field from="Preferred Communications" to="ContactPreferred
Communications"></field>
```

8 Repeat Step 7 for the opportunity child object.

Defining Attributes of the Pick List Element

If you define a pick list element in the siebel_meta_info.xml file, then the TypeId attribute and the SrcObjectTypeId attribute of this element must match the value in the PicklistTypeId attribute. For example, assume you add the following field:

```
<field Name='Note Type' Label='#fld_account_account_note@note_type'
DataType='DTYPE_TEXT' HasPicklist='yes' PicklistIsStatic='yes'
PicklistTypeId='AccountNoteType' IOElemName='NoteType' />
```

In this example, you must set the TypeId attribute in the pick list element to AccountNoteType.

Table 19 describes values to use in the pick list element for a Siebel LOV. If you do not use a Siebel LOV, then adjust these values so they match the field names on the integration component field.

Table 19. Values to Use in the Picklist Element for a Siebel LOV

Attribute	Value
CollectionTypeFldName	Type
ValueFldName	Value
LangFldName	Language

To add more filters, you can use the master_filter_expr attribute. This attribute typically must match the value in the Search Specification property of the object definition for the pick list in Siebel Tools. In the example on Step 3 on page 189, the master_filter_expr attribute constrains the values to the correct LOV Type.

Code to Create a New Object Type for the Pick List

To create a new object type for the pick list, you must use the following format for the type id attribute:

```
<type id="object_namefield_namePicklist" predefined_folder="1">
```

where:

- *object_name* is the name of the object type in the siebel_meta_info.xml file.
- *field_name* is the name of the field that resides in the object you define in the object_name.

For example:

```
<type id="ContactPreferred CommunicationsPicklist" predefined_folder="1">
```

To create a new object type for a pick list, add the following code to the siebel_basic_mapping.xml file:

```
<type id="ContactPreferred CommunicationsPicklist" predefined_folder="1">
    <form message_class="IPM.Contact.SBL.ContactPreferred CommunicationsPicklist"></
form>
    <field id="Label">
        <reader>
          <mapi_user><user_field id="sbl picklistLabel" ol_field_type="1"></user_field>
                <convertor><string/></convertor>
            </mapi_user>
        </reader>
        <writer>
            <outlook_user><user_field id="sbl picklistLabel" ol_field_type="1"></
user_field>
                <convertor><string/></convertor>
            </outlook_user>
        </writer>
    </field>
    <field id="Value">
        <reader>
          <mapi_user><user_field id="sbl picklistValue" ol_field_type="1"></user_field>
                <convertor><string/></convertor>
            </mapi_user>
        </reader>
        <writer>
            <outlook_user><user_field id="sbl picklistValue" ol_field_type="1"></
user_field>
                <convertor><string/></convertor>
            </outlook_user>
        </writer>
```

```
    </field>
    <field id="SortOrder">
        <reader>
            <mapi_user><user_field id="sbl SortOrder" ol_field_type="3"></user_field>
                <convertor><integer/></convertor>
            </mapi_user>
        </reader>
        <writer>
           <outlook_user><user_field id="sbl SortOrder" ol_field_type="3"></user_field>
                <convertor><integer/></convertor>
            </outlook_user>
        </writer>
    </field>
    <field id="IsDefault">
        <reader>
            <mapi_user><user_field id="sbl IsDefault" ol_field_type="6"></user_field>
                <convertor><bool/></convertor>
            </mapi_user>
        </reader>
        <writer>
           <outlook_user><user_field id="sbl IsDefault" ol_field_type="6"></user_field>
                <convertor><bool/></convertor>
            </outlook_user>
        </writer>
    </field>
</type>
```

Code to Define a Map Between the Siebel Server and Siebel CRM Desktop

To define a map between the Siebel Server and Siebel CRM Desktop, add the following code to the Contact object of the siebel_basic_mapping.xml file:

```
   <field id="Preferred Communications">
       <reader>
         <mapi_user><user_field id="sbl Preferred Communications" ol_field_type="1"></
   user_field>
```

```
            <convertor><string/></convertor>

        </mapi_user>

    </reader>

    <writer>

        <outlook_user><user_field id="sbl Preferred Communications"
ol_field_type="1"></user_field>

            <convertor><string/></convertor>

        </outlook_user>

    </writer>

</field>
```

Mapping Child Objects for a Custom Picklist

It is recommended that you do not map a child object directly in the child object. Instead, you can copy values from the parent object and then paste them into the child object. This technique provides the following advantages:

- Allows Siebel CRM Desktop to copy values on the contact to the child object, such as the account or opportunity.
- If the user changes the value in a contact, then Siebel CRM Desktop automatically updates the child objects.

Example Code to Map Child Objects for a Custom Picklist

Figure 11 illustrates example code to map a child object for a custom picklist.

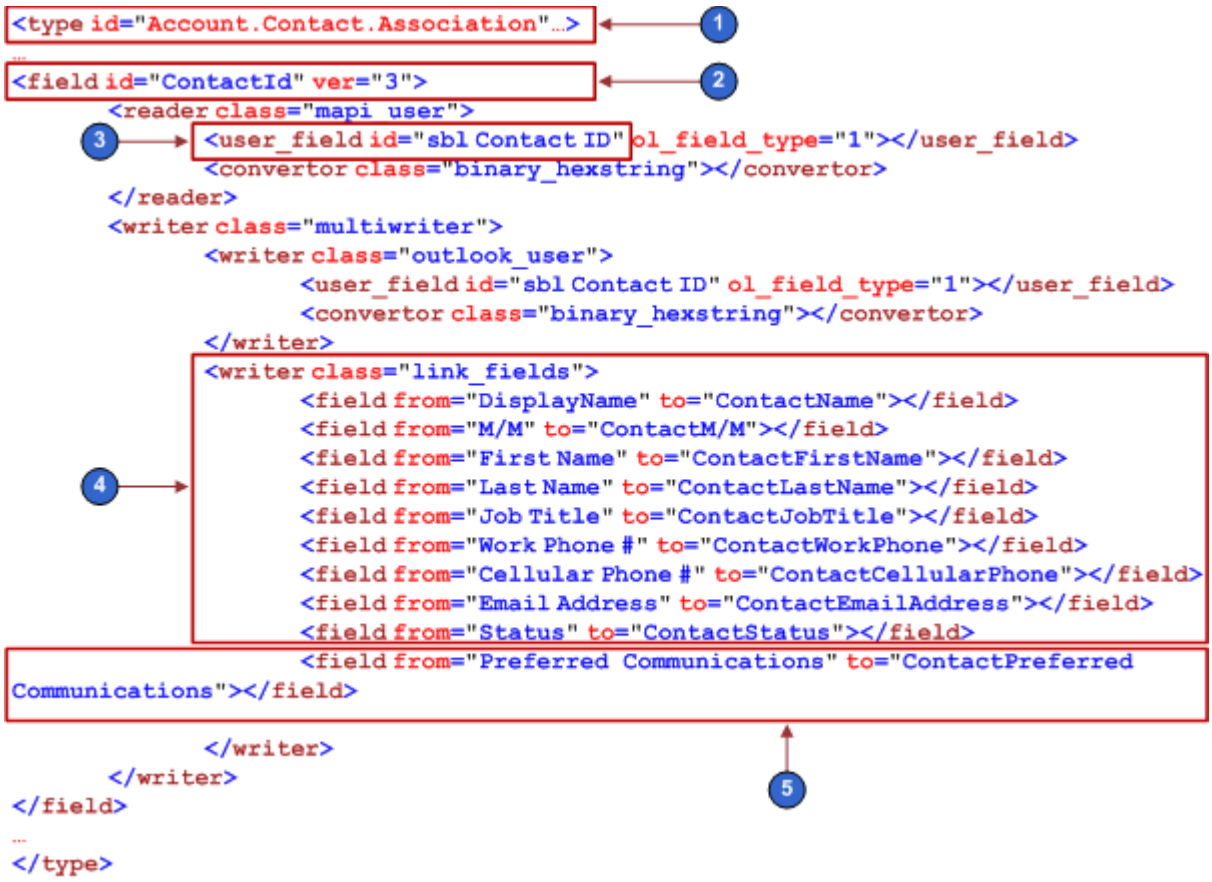


Figure 11. Example Code to Map Child Objects for a Custom Picklist

The example code to map a child object for a custom picklist includes the following items:

1. The following attribute identifies the account child object of the parent contact:

   `type id="Account.Contact.Association"`

2. The following tag identifies the Contact field in the account object:

   `field id="ContactId"`

3. The following attribute identifies the matching ContactId:

   `user_field id="sbl Contact ID"`

4. You copy these fields from the parent contact object and then paste them into the account child object.

5 You add the Preferred Communications field to allow you to add it to Account Contact forms. You define this field in "Code to Define a Map Between the Siebel Server and Siebel CRM Desktop" on page 195.

Customizing the Physical Layout for the Pick List

This task is a step in "Process of Adding a Predefined Siebel Picklist to Siebel CRM Desktop" on page 185.

Figure 12 illustrates the Contact Details section of the contact form you customize in this example.

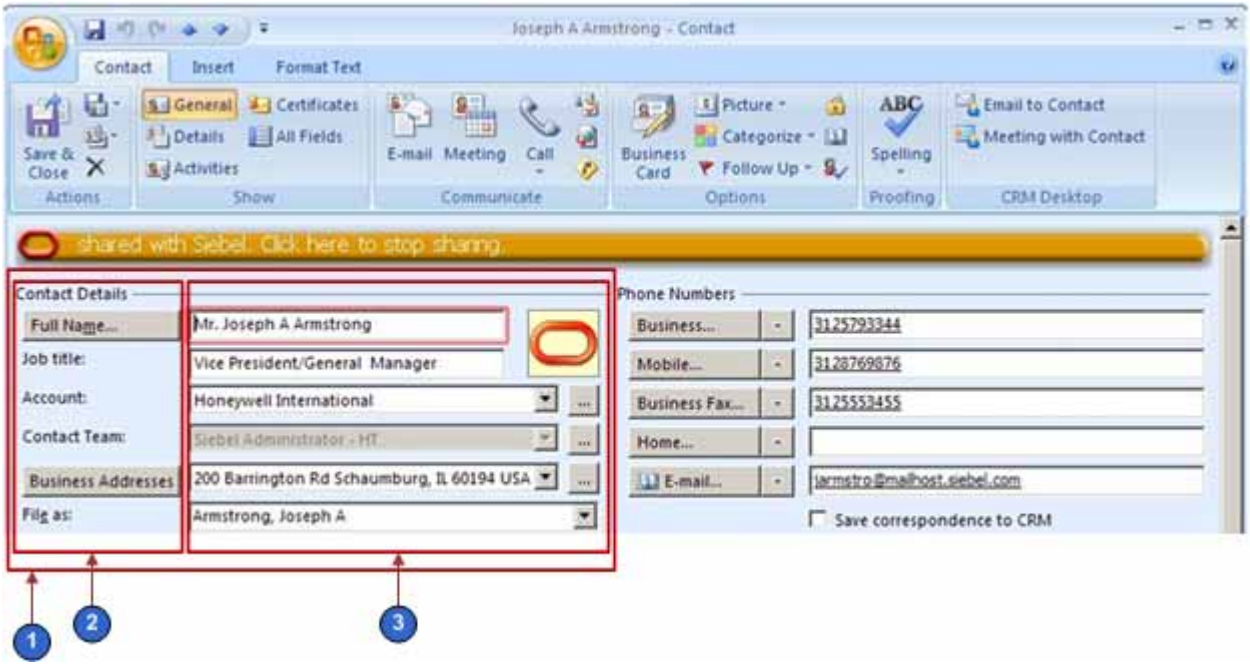


Figure 12. Contact Details Section of the Contact Form

The Contact form includes the following parts:

1 The following object in the forms_11.xml file identifies the Contact Details section:

```
SBL Contact
```

Siebel CRM Desktop considers this area as a single cell. This cell includes two child regions that are placed horizontally in relationship to one another.

2 The left side of the cell includes six cells that are arranged vertically.

3 The right side of the cell includes the following items:

- The Oracle oval link and the ellipses (. . .) buttons include their own cell layers.
- The top cell includes the following items:

- Two horizontal subcells. These subcells accommodate the Oracle oval link.
- Two vertical subcells that accommodate the fields to the left of the Oracle oval link.

- The remaining cells below the top cell include subcells that accommodate their picklists and ellipses (. . .) buttons.

To customize the physical layout for the pick list

1. Examine the physical location of where to place the custom field:
   a. Open the Siebel CRM Desktop client, and then navigate to the Contact form.

      For more information, see Figure 12 on page 198.

   b. In the Contact Details section, locate the Business Addresses field.

      In this example, you add the custom Preferred Contact Method field under the existing Business Address field.

2. To provide sufficient vertical room for the custom field and label, increase the cell size:
   a. Locate the cell size attribute for the cell you must increase.

      You must increase the size of the cell that contains all the other cells. In this example, examine Figure 12, and then note that you must increase the height of cells that are labeled 2 and 3. To do this, you increase the size of the cell that is labeled 1.

   b. Increase the cell size by 30.

      For example, if the current cell size is 155, then change it to 185:

      ```
      <cell size="185">
      ```

3. Add the label for the custom field:
   a. Use an XML editor to open the forms_11.xml file.

      In this example, assume you are using Microsoft Outlook 2003. For more information, see "Customizing a Form" on page 150.

   b. To locate the form you must modify, locate the following code:

      ```
      form id="SBL Contact"
      ```

      Each form includes the SBL prefix which is immediately followed by the object name, such as Contact.

   c. Locate the following code, which defines the label for the existing business address:

      ```
      <cell size="22">
         <control id="dd_addresses" class="dropdown"
         caption="#head_business_addresses" tab_order="12" />
      </cell>
      ```

      Code for the label and fields is located in the following object after the validation rules:

      ```
      SBL Contact
      ```

d To define the label for the new contact method, add the following code immediately below the code you located in Step c:

```
<cell size="22">

    <static id="ContactMethod" tab_order="9">

        <text>#lbl_ContactMethod</text>

    <static>

</cell>
```

For more information, see "Format of the Label for a Custom Field" on page 201.

4 Add the custom field:

a Locate the following code, which defines the field for the existing business address:

```
<cell size="21">
   <stack layout="horz" spacing="5">
   <cell>
   <combobox id="business_address_mvg" tab_order="13">
   <field>Primary Address Id</field>
        DETAILS DELETED
     </control>
   </cell>
 </stack>
</cell>
```

To simplify this step, search for unique text, such as business_address_mvg. For brevity, *DETAILS DELETED* indicates that details for this tag are removed from this book.

b To define the field for the new contact method, add the following code immediately below the code you located in Step a:

```
<cell size="22">

    <combobox id="cbx_ContactPreferred CommunicationsPicklist">

        <field>Preferred Communications</field>

       <source type="ContactPreferred CommunicationsPicklist" field="Value"
format=":[:(Label):]">  </source>

    </combobox>

</cell>
```

For more information, see "Format of the Custom Field" on page 201.

5 Define the custom symbolic strings:

a Use an XML editor to open the package_res.xml file.

b Add the following code anywhere in the file:

```
<str key="lbl_ContactMethod">Contact Method:</str>
<str key="head_contact_method">Contact Method</str>
```

You can place these strings anywhere in the file. To assist with maintenance, it is recommended that you place them with similar strings. If necessary, to create symbolic strings that accommodate another language, you can create alternate package_resource_xml files. For more information, see "Customizing Localization" on page 153.

Format of the Label for a Custom Field

To define the label for the custom contact method field, you add the following code:

```
<cell size="22">

    <static id="ContactMethod" tab_order="9">

        <text>#lbl_ContactMethod</text>

    <static>

</cell>
```

where:

- *Id* is an arbitrary, unique value for the form.
- *Tab_order* determines the order in which Siebel CRM Desktop places the cursor in the fields in the form when the user presses the TAB key.
- *Text* defines the text that Siebel CRM Desktop uses for the label.
- # indicates the symbolic string that the package_res.xml file defines. You can use this string for a global deployment.

Format of the Custom Field

To define the custom field for the contact method, you add the following code:

```
<cell size="22">

    <combobox id="cbx_ContactPreferred CommunicationsPicklist">

        <field>Preferred Communications</field>

        <source type="ContactPreferred CommunicationsPicklist" field="Value"
format=":[:(Label):]">  </source>

    </combobox>

</cell>
```

where:

- *Id* is an arbitrary, unique value in the form. A picklist must include the cbx prefix.
- *Tab_order* determines the order in which Siebel CRM Desktop places the cursor in the fields in the form when the user presses the TAB key. The value you enter here must be greater than the value you enter in the Tab_order for the label.
- *Source* determines where to get the data.
- *Type* identifies the object name. This name is defined in the siebel_basic_mapping.xml file.
- *Field* specifies the field that provides the values that the user chooses from the pick list. In this example, the Value field provides these values.
- <*Field*> identifies the field name from the object definition that populates the picklist.
- *Format* specifies how to display text in the picklist.

The format tag allows you to use a combination of static text and fields in the picklist. It uses the following format:

```
*any_static_text*:[:(field1_name):]*any_static_text*:[:(field2_name):]*any_static_text*
```

You must use the bracket, colon, and parentheses. For example:

```
Contact :[:(First Name):] :[:(Last Name):] ? Contact: John Smith
```

Publishing and Testing a Custom Picklist

This task is a step in "Process of Adding a Predefined Siebel Picklist to Siebel CRM Desktop" on page 185.

In this topic, you publish and test your customization.

To publish and test a custom picklist

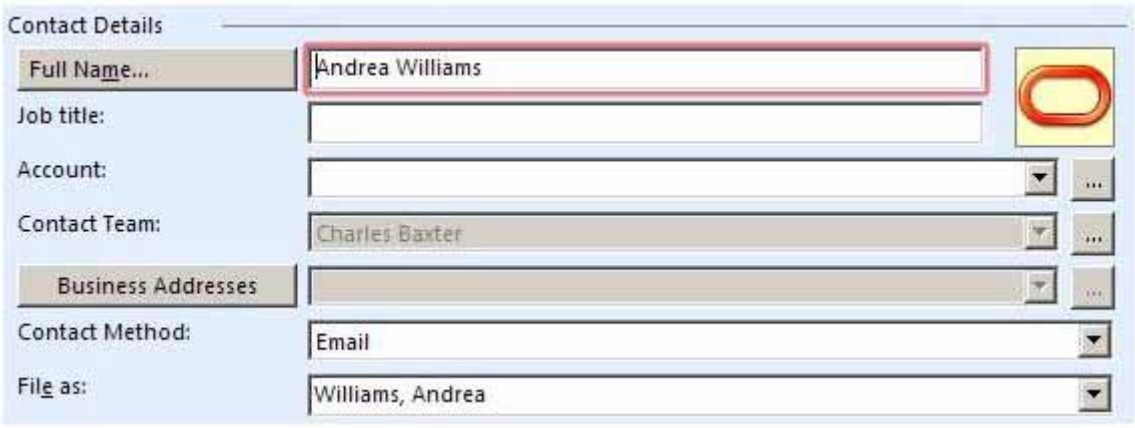
1. Publish your changes.

   For more information, see "Republishing a Customization Package" on page 127.

2. Test your changes:

   a. Open the Siebel CRM Desktop Client, and then navigate to the Contact form.

b Verify that the form includes a label and picklist for the Contact Method field, as illustrated in the following diagram:



c Click the down arrow next to the Contact Method field, and then verify that Siebel CRM Desktop displays a picklist that contains the following values:

- None
- Chat
- Email
- Fax
- Pager
- Phone
- Wireless Message.

d Choose a value in the picklist, and then verify that Siebel CRM Desktop changes the value in the Contact Method field to the value you choose.

Process of Adding an MVG Field

This topic gives one example of how to add an MVG field. You might use this feature differently, depending on your business model. It includes the following topics:

To add an MVG field, you do the following:

1 "Identifying Predefined MVG Objects in Siebel CRM" on page 204

2 "Process of Making Siebel CRM Data Available to Add an MVG" on page 206

3 "Process of Modifying the Customization Package to Add an MVG" on page 211

4 "Publishing and Testing a Custom MVG Field" on page 216

An MVG field displays an association with other objects. This association can be a many to many association, or a many to one association. The user can use an MVG field to do the following:

- Add or remove an association
- Change a primary association
- Browse existing associations

You can also use an MVG to economize the layout of a form. You can display a set of associated records in a single-line control instead of using the outlook_view control.

In this example, you add an MVG control to the Opportunity form. This MVG displays an association between an opportunity and channel partners.

Identifying Predefined MVG Objects in Siebel CRM

This task is a step in "Process of Adding an MVG Field" on page 203.

In this topic, you identify the MVG objects that you use to add an MVG field for this example. These objects come predefined with Siebel CRM.

To identify predefined MVG objects in Siebel CRM

1. Identify the field that Siebel CRM associates with the MVG you must add:
   a. Open Siebel Call Center.
   b. Navigate to the Opportunities list, and then click the link in the Opportunity Name field of an opportunity.
   c. Click the More Info tab.
   d. Locate the More Info field.
   e. Choose the Help menu, and then the About View menu item.

      The About View dialog box lists the applets in the order in which Siebel Call Center displays them.
   f. Note the applet name in which Siebel Call Center displays the More Info field.

      In this example, this is the Contact Form Applet - Child applet.
   g. In Siebel Tools, in the Object Explorer, click Applet.
   h. In the Applets list, query the Name property for Opportunity Form Applet - Child Big.
   i. Right-click the Opportunity Form Applet - Child Big applet, and then choose Edit Web Layout.

      If Siebel Tools displays the Read-only Object dialog box, then you must check out the project. For more information, see "Checking Out a Project in Siebel Tools" on page 155.
   j. In the Applet Web Template editor, click the More Info control.

k In the Properties window, note the values for the following properties.

Property	Value
Field	Mail Stop
MVG Applet	Partner Lead Name Mvg Applet

In this example, Siebel CRM associates the Partner field with the Lead Partner MVG.

2 Identify the MVG link.

a In the Object Explorer, click Business Component.

b In the Business Components list, query the Name property for Opportunity.

c In the Object Explorer, expand the Business Component tree, and then click Field.

d In the Fields list, query the Name property for Partner, and then note the values for the following properties.

Property	Value
Multi Valued	TRUE
Multi Valued Link	Channel Partner

e In the Object Explorer, click Multi Value Link.

f In the Multi Value Links list, query the Name property for the Multi Valued Link that the field references that you noted in Step d. In this example, this link is Channel Partner.

g Note the values for the following properties.

Property	Value
Destination Link	Opportunity/Channel Partner
Primary Id Field	Primary Partner Id

h In the Object Explorer, click Link.

i In the Links list, query the Name property for the Destination Link that you noted in Step g.

In this example, this link is Opportunity/Channel Partner. Note the values for the properties described in the following table.

Property	Value
Inter Table	S_OPTY_ORG If the Inter Table property: ■ Contains an intersection table, then the link maintains a many to many association. ■ Is empty, then the link maintains a one to many association.
Child Business Component	Channel Partner The business component in the Child Business Component property must be displayed.

To identify the required business component, you can also examine the Business Component property of the Partner Lead Name MVG Applet.

Process of Making Siebel CRM Data Available to Add an MVG

This task is a step in "Process of Adding an MVG Field" on page 203.

To make Siebel CRM data available to add an MVG, you do the following:

1 "Creating an Integration Object for the Channel Partner MVG" on page 206

2 "Creating an Integration Component for the Channel Partner MVG" on page 208

3 "Extending an Integration Object for the Primary Id Field" on page 210

This topic describes how to make certain Siebel CRM data available to Siebel CRM Desktop. Some Siebel CRM data is available without customize Siebel CRM Desktop, such as opportunities, accounts, and contacts. Other Siebel CRM data is not available. For example, if your implementation requires Channel Partner data, then you must configure integration objects to make this data available to Siebel CRM Desktop.

Creating an Integration Object for the Channel Partner MVG

This task is a step in "Process of Making Siebel CRM Data Available to Add an MVG" on page 206.

In this topic, you create a new integration object that makes the channel partner data available to the EAI Siebel Adapter. The work you perform in this topic allows the PIM Client Sync Service business service to access channel partner data. For more information, see "Siebel Enterprise Components That Siebel CRM Desktop Uses" on page 25.

To create an integration object for the channel partner MVG

1 In Siebel Tools, choose the File Menu, and then the New Object menu item.

2 Click the EAI tab, click Integration Object, and then click OK.

3 In the Integration Object Builder Dialog box, choose values for the following items, and then click Next.

Property	Value
Project	Choose a project. It is recommended that you create a separate project for any customizations you make to Siebel CRM Desktop. For example, use a project named Siebel CRM Desktop.
Business Service	EAI Siebel Wizard

4 Choose values for the following items, and then click Next.

Property	Value
Source Object	Channel Partner
Source Root	Channel Partner
Integration Object Name	CRMDesktopChannelPartnerIO

5 Expand the Channel Partner tree, choose the integration components you must include with this integration object, and then click Next.

As the default, Siebel Tools includes a check mark for each integration component. For this example, accept the default.

6 Click Next and then click Finish.

7 Examine the properties of the integration object you created in Step 6:

a In the Object Explorer, click Integration Object, query the Name property for CRMDesktopChannelPartnerIO, and then note the following property.

Property	Value
External Name	Channel Partner You will use this property as a value for the IntObjName attribute.

b In the Object Explorer, expand the Integration Object tree, and then click Integration Object Component.

c In the Integration Object Components list, query the Name property for Channel Partner, and then note the following properties.

Property	Value
XML Tag	ChannelPartner You will use this property as a value for the SiebMsgXmlElemName attribute.
XML Container Element	ListOfChannelPartner You will use this property as a value for the SiebMsgXmlCollectionElemName attribute.

Creating an Integration Component for the Channel Partner MVG

This task is a step in "Process of Making Siebel CRM Data Available to Add an MVG" on page 206.

In this topic, you add a channel partner integration component as a child of the integration object that Siebel CRM Desktop uses for opportunities. This allows Siebel CRM Desktop to query the channel partners that are associated with the opportunities for a user.

To create an integration component for the channel partner MVG

1 In Siebel Tools, display the object type named Integration Object.

For more information, see "Displaying Object Types in Siebel Tools" on page 156.

2 In the Object Explorer, click Integration Object.

3 In the Integration Objects list, query the Name property for CRMDesktopOpportunityIO, and then make sure the Object Locked property contains a check mark.

Siebel CRM Desktop adds the CRMDesktopOpportunityIO integration object to the repository when you install Siebel CRM Desktop on the Siebel Server. You must install Siebel CRM Desktop before you can complete this task.

4 In the Object Explorer, expand the Integration Object tree, and then click Integration Component.

5 In the Integration Components list, add a new record with the following values.

Property	Value
Business Component	Channel Partner
Name	Opportunity_ChannelPartner
External Name	Channel Partner
External Sequence	2 For more information, see "Requirements for the Sequence Property" on page 190.

Property	Value
XML Sequence	10002 For more information, see "Requirements for the Sequence Property" on page 190.
XML Container Element	ListOfOpportunity_ChannelPartner
XML Tag	Opportunity_ChannelPartner

6 In the Object Explorer, expand the Integration Component tree, and then click Integration Component Field.

7 In the Integration Component Fields list, add new records with the following values.

Name	Data Type	Length
IsPrimaryMVG	DTYPE_TEXT	1
Location	DTYPE_TEXT	50
Organization BU Name	DTYPE_TEXT	50
Partner	DTYPE_TEXT	100
Partner Id	DTYPE_Id	30
Partner Status	DTYPE_TEXT	30
operation	DTYPE_TEXT	30
searchspec	DTYPE_TEXT	250

8 In the Object Explorer, click Integration Component Key.

9 In the Integration Component Keys list, add new records with the following values.

Name	Key Sequence Number	Key Type
Modification Key	1	Modification Key
Primary Key	1	User Key
Status Key	1	Status Key

10 In the Object Explorer, click Integration Component User Prop.

11 In the Integration Component User Props list, add new records with the following values.

Name	Value
MVGAssociation	Y
MVGLink	Channel Partner

12 Compile your changes.

For more information, see *Using Siebel Tools*.

13 In the Object Explorer, click Integration Component, and then note the following properties of the integration component that you added in Step 5. You use these values when you modify the metadata for the customization package.

Property	Value
Parent Name	CRMDesktopOpportunityIO You will use this property as a value for the IntObjName attribute.
XML Tag	Opportunity_ChannelPartner You will use this property as a value for the SiebMsgXmlElemName attribute.
XML Container Element	ListOfOpportunity_ChannelPartner You will use this property as a value for the SiebMsgXmlCollectionElemName attribute.

Extending an Integration Object for the Primary Id Field

This task is a step in "Process of Making Siebel CRM Data Available to Add an MVG" on page 206.

In this topic, you make the Primary Id field available to Siebel CRM Desktop. You make the primary on the opportunity available so that Siebel CRM Desktop can identify the record to display in the opportunity form if the opportunity includes more than one channel partner.

In this example, the Primary Id Field property of the MVG link contains a value. If this property were empty, then you would skip this topic and proceed to Step 1 on page 211.

To extend an integration object for the Primary Id field

1 In the Object Explorer, click Integration Object.

2 In the Integration Objects list, query the Name property for CRMDesktopChannelPartnerIO, and then make sure the Object Locked property contains a check mark.

You created the CRMDesktopChannelPartnerIO integration object in Step 6 on page 207.

3 In the Object Explorer, expand the Integration Object tree, and then click Integration Component.

4 In the Integration Components list, query the Name property for Opportunity.

5 In the Object Explorer, expand the Integration Components tree, and then click Integration Component Field.

6 In the Integration Component Fields list, add a new record with the following values.

Property	Value
Name	Primary Partner Id
External Name	Primary Partner Id
Length	15
Data Type	DTYPE_ID
External Data Type	DTYPE_ID
External Sequence	139 For more information, see "Requirements for the Sequence Property" on page 190.
XML Sequence	139 For more information, see "Requirements for the Sequence Property" on page 190.
XML Tag	PrimaryPartnerId

7 Compile your changes.

For more information, see *Using Siebel Tools*.

Process of Modifying the Customization Package to Add an MVG

This task is a step in "Process of Adding an MVG Field" on page 203.

To modify the customization package to add an MVG, you do the following:

1 "Adding a Custom Object" on page 211

2 "Adding the MVG Link" on page 212

3 "Adding the Primary Field" on page 212

4 "Adding a Field" on page 213

5 "Adding a Lookup View" on page 214

6 "Code to Define a Salesbook Control" on page 225

7 "Adding a Label, a Button, and a Selector Control" on page 215

8 "Customizing the Validation Message and Labels" on page 216

Adding a Custom Object

This task is a step in "Process of Modifying the Customization Package to Add an MVG" on page 211.

To add a custom object to Siebel CRM Desktop, you display the object in Siebel CRM and then modify customization package XML files. In this example, you make available and then add the Channel Partner object.

To add a custom object

1. Make sure the object you must add is available.

   For more information, see "Creating an Integration Object for the Channel Partner MVG" on page 206.

2. To add a custom object type, modify the siebel_meta_info.xml file.

   For more information, see "Code to Add a Custom Object Type" on page 217.

3. To map objects, modify the siebel_basic_mapping.xml file.

   For more information, see "Code to Map a Custom Object" on page 218.

4. To configure synchronization for the custom object, modify the connector_configuration.xml file.

For more information, see "Code to Configure Synchronization for a Custom Object" on page 220.

Adding the MVG Link

This task is a step in "Process of Modifying the Customization Package to Add an MVG" on page 211.

In this topic, you add the MVG link to the business logic file.

To add the MVG link

1. Use a JavaScript editor to open the business_logic.js file.

   For more information, see "About Files in the Customization Package" on page 144

2. Locate the following function:

   ```
   create_siebel_meta_scheme2
   ```

3. Add the code to add a new association.

   For more information, see "Code to Add a New Association" on page 223.

Adding the Primary Field

This task is a step in "Process of Modifying the Customization Package to Add an MVG" on page 211.

In this topic, you add the primary field to the customization package. This field displays in the MVG dialog box that you add for this example.

To add the primary field

1. To add the primary field, you do the following:

   a. Use an XML editor to open the siebel_meta_info.xml file.

b Add the following code to the Opportunity object:

```
<field Name='Primary Partner Id' Label='Primary Partner Id'
DataType='DTYPE_ID' IsFilterable='no' IsRefObjId='yes'
RefObjTypeId='ChannelPartner' IOElemName='PrimaryPartnerId' />
```

c Save and then close the siebel_meta_info.xml file.

d Use an XML editor to open the siebel_basic_mapping.xml file:

e Add the following code to the Opportunity type tag:

```
<field id="Primary Partner Id">
 <reader>
   <mapi_user>
      <user_field id="sbl Primary Partner Id" ol_field_type="1"></user_field>
     <convertor><binary_hexstring/></convertor>
   </mapi_user>
  </reader>
 <writer>
   <outlook_user>
      <user_field id="sbl Primary Partner Id" ol_field_type="1"></user_field>
     <convertor><binary_hexstring/></convertor>
   </outlook_user>
  </writer>
</field>
```

f Save and then close the siebel_basic_mapping.xml file.

2 Add the link:

a Use an XML editor to open the connector_configuration.xml file.

b Add the following code to the Opportunity type tag:

```
<link>Primary Partner Id</link>
```

c Save and then close the connector_configuration.xml file.

Adding a Field

This task is a step in "Process of Modifying the Customization Package to Add an MVG" on page 211.

This topic describes how to add the ChannelPartnerStatus field.

To add a field

1 Use an XML editor to open the siebel_basic_mapping.xml file.

2 Add the following code to the Opportunity.Channel_Partner.Association type:

```
<field id="ChannelPartnerStatus">
 <reader>
   <mapi_user>
     <user_field id="sbl ChannelPartnerStatus" ol_field_type="1"></user_field>
     <convertor><string/></convertor>
   </mapi_user>
  </reader>
  <writer>
    <outlook_user>
      <user_field id="sbl ChannelPartnerStatus" ol_field_type="1"></user_field>
      <convertor><string/></convertor>
    </outlook_user>
  </writer>
</field>
```

3 Save and then close the siebel_basic_mapping.xml file.

Adding a Lookup View

This task is a step in "Process of Modifying the Customization Package to Add an MVG" on page 211.

To customize a salesbook control, you use the definition of a lookup view. A salesbook control displays a list of objects and then allows the user to choose any object from this list. Siebel CRM Desktop uses this control on forms and MVG dialogs. In this example, you add the definition for a lookup view for Channel Partner objects.

To add a lookup view

1 Use an XML editor to open the lookup_view_defs.xml file.

2 Add the following code to the array tag that contains the lookup types in the file:

```
<item value="ChannelPartner"></item>
```

3 Add the definition for the lookup view.

For more information, see "Code to Add a Lookup View" on page 221.

4 Save and then close the lookup_view_defs.xml file.

Adding a Label, a Button, and a Selector Control

This task is a step in "Process of Modifying the Customization Package to Add an MVG" on page 211.

In this topic, you add a label, a button, and a selector control.

To add a label, a button, and a selector control

1. Use an XML editor to open the forms_xx.xml file.
2. Add a label for the MVG:
   a. Locate the following section:

```
right side captions
```

   b. Insert the following code immediately under the code that defines the Probability label:

```
<cell size="22">
  <static id="0x20014" tab_order="166">
      <text>#lbl_channel_partner</text>
   </static>
</cell>
```

3. Add the button and primary selector control:
   a. Locate the following section:

```
right side fields
```

   b. Add the following code immediately under the code that defines the Probability control:

```
<cell size="22">
 <stack layout="horz">
   <cell>
     <mvg_primary_selector id="channel_partner_mvg">
       <source type="Opportunity.Channel_Partner.Association"
left_id="OpportunityId" item_value="ChannelPartnerId"
display_format=":[:(PartnerName):]"></source>
       <field>Primary Partner Id</field>
     </mvg_primary_selector>
   </cell>
   <cell size="5"></cell>
   <cell size="22">
     <button id="btn_mvgChannelPartner">
```

```
            <text>...</text>
          </button>
        </cell>
      </stack>
    </cell>
```

Make sure you specify the Id for the button and the Id for the primary selector in the same way that you specified them in the script.

4 Increase the size of the cell that contains the label, the button, and the selector control:

a Locate the following section:

```
Category bar
```

b Locate the fifth cell that is included in the Category bar section.

c Change the code of the fifth cell to the following code:

```
<cell size="153">
```

5 Save and then close the forms_xx.xml file.

Customizing the Validation Message and Labels

This task is a step in "Process of Modifying the Customization Package to Add an MVG" on page 211.

In this topic, you customize the validation message and a label for the dialog box and forms.

To customize the validation message and labels

1 Use an XML editor to open the package_res.xml file.

2 Add the following code to the Script section:

```
<str key="msg_channel_partner_present">This channel partner is already present in
the list.</str> <str key="msg_channel_partner_is_primary">This channel partner is
primary. The primary record cannot be removed. To remove this record, make another
record primary first.</str> <str key="msg_channel_partner_add_caption">Add
channel partner.</str><str key="lbl_channel_partner">Lead Partner Name</str>
```

3 Add the same code that you added in Step 2 to the Messages section.

4 Close and then save the package_res.xml file.

The following code specifies the values for labels you use in the dialog box and form layouts:

```
<str key="lbl_channel_partner">Lead Partner Name</str>
```

Publishing and Testing a Custom MVG Field

This task is a step in "Process of Adding an MVG Field" on page 203.

In this topic, you publish and test your customization.

To publish and test a custom MVG field

1. Publish your changes.

   For more information, see "Republishing a Customization Package" on page 127.

2. Test your changes:

   a. Open the Siebel CRM Desktop Client, and then navigate to the Opportunity form.

   b. Verify that the form includes an MVG for the Lead Partner Name field.

   c. Click the MVG that Siebel CRM Desktop displays next to the Lead Partner Name field, and then verify that Siebel CRM Desktop does the following:

      - Displays the Channel Partners MVG dialog box
      - Displays the list of partners in the Associated Channel Partners window of the dialog box
      - Includes a partner record in the Primary window

   d. Enter letters in the Enter Value to Find Record window.

   e. Verify that Siebel CRM Desktop automatically displays records in accordance with the letters you enter.

   f. To verify the salesbook control, click the salesbook icon and then verify that Siebel CRM Desktop displays the SalesBook dialog box, and that this dialog box displays a list of channel partners.

Example Code You Use to Add an MVG

This topic describes some of the code you use to add an MVG in this example. It includes the following topics:

- "Code to Add a Custom Object Type" on page 217
- "Code to Map a Custom Object" on page 218
- "Code to Configure Synchronization for a Custom Object" on page 220
- "Code to Add a Lookup View" on page 221
- "Code to Add a View" on page 221
- "Code to Add a View" on page 221
- "Code to Add a New Association" on page 223
- "Code to Define a Salesbook Control" on page 225

Code to Add a Custom Object Type

To add a custom object type, you add the following code anywhere in the siebel_meta_info.xml file. To assist with debugging, it is recommended that you place this code after the last Type definition:

```
<object TypeId='ChannelPartner' Label='Channel Partner' LabelPlural='Channel
Partners' EnableGetIDsBatching='true' ViewMode='Sales Rep' IntObjName='Channel
Partner' SiebMsgXmlElemName='ChannelPartner'
SiebMsgXmlCollectionElemName='ListOfChannelPartner' >

  <field Name='DS Updated' Label='DS Updated' DataType='DTYPE_DATETIME'
IsFilterable='no' IsHidden='yes' IOElemName='DSUpdated' />

  <field Name='Id' Label='Id' IsPrimaryKey='yes' DataType='DTYPE_ID'
IsFilterable='no' IsHidden='yes' IOElemName='Id'/>

  <field Name='Name' Label='Name' DataType='DTYPE_TEXT' IsPartOfUserKey='yes'
IOElemName='Name' />

  <field Name='Location' Label='Location' DataType='DTYPE_TEXT'
IsPartOfUserKey='yes' IOElemName='Location'/>

</object>
```

This code does the following:

- Uses properties of the integration object and integration component as the values for attributes.
- References only a few of the many fields that exist in the Channel Partner object in Siebel CRM.
- Uses the Name and Location fields as parts of the user key. You define the natural key in the connector_configuration.xml file.

Code to Map a Custom Object

You can map a field of a Siebel CRM object to a Microsoft Outlook field or to a custom Siebel CRM Desktop field. For example, you can do the following:

- Map the Name field in Siebel CRM to the LastName field in Microsoft Outlook
- Map the mapLocation field in Microsoft Outlook to the Location field in Siebel CRM

To map objects, you add the following code to the siebel_basic_mapping.xml file:

```
<type id="ChannelPartner" hidden_folder="true" folder_type="10"
display_name="CHPT">

 <form message_class="IPM.Contact.SBL.Channel_Partner" display_name="Channel
Partner" icon="type_image:User:16"></form>

  <field id="Name">

   <reader>

    <mapi_std>

     <mapi_tag id="0x3A110000"></mapi_tag>

     <convertor><string/></convertor>

    </mapi_std>

   </reader>
```

```
 <writer>
  <outlook_std>
   <outlook_field id="LastName"></outlook_field>
   <convertor><string/></convertor>
  </outlook_std>
 </writer>
 <reader>
  <mapi_std>
   <mapi_tag id="0x3A060000"></mapi_tag>
   <convertor><string/></convertor>
  </mapi_std>
 </reader>
 <writer>
  <outlook_std>
   <outlook_field id="FirstName"></outlook_field>
   <convertor><string/></convertor>
  </outlook_std>
 </writer>
</field>
<field id="Location">
 <reader>
  <mapi_user>
   <user_field id="sbl Location" ol_field_type="1"></user_field>
   <convertor><string/></convertor>
  </mapi_user>
 </reader>
 <writer>
  <outlook_user>
   <user_field id="sbl Location" ol_field_type="1"></user_field>
   <convertor><string/></convertor>
```

```
      </outlook_user>
    </writer>
  </field>
</type>
```

Note the following requirements:

- The value for the *type id* tag in the siebel_basic_mapping.xml file must equal the value in the TypeId object in the siebel_meta_info.xml file.
- If you define a new type, then you must include the *form* tag. The forms_xx.xml file includes the form definition. The value for the form tag must equal the *form id* attribute in the form definition. Because this example does not require a form, the form tag is empty. Even if your example does not require a form, to uniquely identify a type, you must define a value for the mesage_class attribute. This value must start with the following code:

  ```
  IPM.Contact.SBL
  ```

Code to Configure Synchronization for a Custom Object

To configure synchronization for a custom object, you add the following code to the connector_configuration.xml file:

```
<type id="ChannelPartner">
  <view label="Channel Partner" label_plural="Channel Partners"
small_icon="type_image:Account:16" normal_icon="type_image:Account:24"
large_icon="type_image:Account:48"></view>
  <synchronizer name_format=":[:(Name):]">
    <links>
    </links>
    <natural_keys>
      <natural_key>
        <field>Name</field>
        <field>Location</field>
      </natural_key>
    </natural_keys>
  </synchronizer>
</type>
```

Note the following:

- The value in the *type id* tag must equal the value in the TypeId object in the meta_info.xml file.

- The natural_key tag includes the Name and Location fields as part of the user key.

Code to Add a Lookup View

To add a lookup view, you add the following code to the lookup_view_defs.xml file:

```
<lookup_view_def key="lookup:channel_partners">

    <display name="Channel Partners"></display>

    <filter dasl="[http://schemas.microsoft.com/mapi/proptag/0x001A001E] &gt;=
'IPM.Contact.SBL.Channel_Partner' AND [http://schemas.microsoft.com/mapi/proptag/
0x001A001E] &lt;= 'IPM.Contact.SBL.Channel_Partner'"></filter>

    <view id="channel_partner:salesbook"></view>

    <quick_lookup dasl_format="[http://schemas.microsoft.com/mapi/id/{00062004-
0000-0000-C000-000000000046}/8005001E] = '%s'"></quick_lookup>

    <type id=""></type>

</lookup_view_def>
```

Table 20 describes important attributes you use in this code.

Table 20. Attributes in the Code to Add a Lookup View

Attribute	Description
key	Id of the lookup control.
display name	Caption of the lookup control.
filter	Object that Siebel CRM Desktop displays on the lookup control. To specify an object type, you use the message_type attribute in the siebel_basic_mapping.xml file.
view	Id of the salesbook control that Siebel CRM Desktop uses for this lookup.
type id	The type of object that Siebel CRM Desktop creates if the user clicks New on the lookup control. If this attribute is empty, then Siebel CRM Desktop disables the button.

Code to Add a View

To add the definition for a view, you add the following code to the views.xml file:

```
<str key="channel_partner:mvg">

    <![CDATA[<?xml version="1.0"?>

    <view type="table">

    <viewname>Phone List</viewname>
```

```
    <viewstyle>table-layout:fixed;width:100%;font-family:Segoe UI;font-
style:normal;font-weight:normal;font-size:8pt;color:Black;font-charset:0</
viewstyle>

    <viewtime>0</viewtime>

    <linecolor>8421504</linecolor>

    <linestyle>3</linestyle>

    <gridlines>1</gridlines>

    <collapsestate/>

    <rowstyle>background-color:window;color:windowtext</rowstyle>

    <headerstyle>background-color:#D3D3D3</headerstyle>

    <previewstyle/>

    <arrangement>

        <autogroup>0</autogroup>

        <collapseclient/>

    </arrangement>

    <multiline>

        <width>0</width>

    </multiline>

    <column>

        <name>HREF</name>

        <prop>DAV:href</prop>

        <checkbox>1</checkbox>

    </column>

    <column>

        <type>string</type>

        <heading>Name</heading>

        <prop>http://schemas.microsoft.com/mapi/string/{00020329-0000-0000-C000-
000000000046}/sbl%20PartnerName</prop>

        <width>426</width>

        <style>padding-left:3px;;text-align:left</style>

        <editable>1</editable>
```

```
            <userheading>Name</userheading>
        </column>
        <column>
            <type>string</type>
            <heading>Location</heading>
            <prop>http://schemas.microsoft.com/mapi/string/{00020329-0000-0000-C000-
000000000046}/sbl%20PartnerLocation</prop>
            <width>426</width>
            <style>padding-left:3px;;text-align:left</style>
            <editable>1</editable>
            <userheading>Location</userheading>
        </column>
        <orderby>
            <order>
                <heading>File As</heading>
                <prop>urn:schemas:contacts:fileas</prop>
                <type>string</type>
                <sort>asc</sort>
            </order>
        </orderby>
        <groupbydefault>0</groupbydefault>
        <previewpane>
            <markasread>0</markasread>
        </previewpane>
        </view>]]>
    </str>
```

Code to Add a New Association

To add a new association, you add the following code to the business_logic.js file:

```
    var opportunity_channel_partner =
            add_mvg_link("Opportunity", "ChannelPartner", "Primary Partner Id", null,
```

```
                "Opportunity.Channel_Partner.Association", "OpportunityId",
  "ChannelPartnerId",
                null, "ChannelPartnerStatus",
                null, ["lookup:channel_partners"],
                false, false, false, true);
    deny_primary_delete(opportunity_channel_partner.mvg1);//optional
    opportunity_channel_partner.mvg1.dialog_template_params = {
        "dialog_caption": "#obj_activity_employee_plural",
        "autocomplete_display_format": ":[:(Name):]",
        "associations_view_caption": "#head_associated_channel",
        "associations_view_id": "channel_partner:mvg",
        "primary_selector_display_format": ":[:(PartnerName):]"
    }
```

The ChannelPartnerStatus field contains the status of the Channel Partner object, such as unsaved, deleted, and so forth.

Table 21 describes the important parameters you can use with the add_mvg_link function.

Table 21. Attributes in the Code to Add a New Association

Attribute	Description
left_type	The type of the first linked object. For example, Opportunity.
right_type	The type of the second linked object. For example, Channel Partner.
left_obj_primary	The field of the Opportunity object. This field contains the primary Id for the Channel Partner object.
right_obj_primary	The field of the Channel Partner object. This field contains the primary Id for the Opportunity object.
assoc_type	The Id of the association type described in the siebel_meta_info.xml file and the siebel_basic_mapping.xml file.
left_link	The field of the association object that contains the Id of the opportunity.
right_link	The field of the association object that contains the Id of the channel partner.
left_assoc_status	The field of the association that contains the status of the opportunity.
right_assoc_status	The field of the association that contains the status of the channel partner.
left_objs_view_ids	The list of lookup view definitions you must use to display opportunity objects.

Table 21. Attributes in the Code to Add a New Association

Attribute	Description
right_objs_view_ids	The list of lookup view definitions you must use to display channel partner objects.
left_primary_refresh_required	If the primary object is changed, and if the left_primary_refresh_required attribute is true, then Siebel CRM Desktop updates the object Id for the primary field of the opportunity that you specify in the left_obj_primary attribute. If you use the following tag in the siebel_basic_mapping.xml file for this field, then set the left_primary_refresh_required attribute to true: `writer class="link_fields"`
right_primary_refresh_required	If the primary object is changed, and if the right_primary_refresh_required attribute is true, then Siebel CRM Desktop updates the object Id for the primary field of the channel partner that you specify in the right_obj_primary attribute. If you use the following tag in the siebel_basic_mapping.xml file for this field, then set the right_primary_refresh_required attribute to true: `writer class="link_fields"`
assoc_left_link_refresh_required	If the opportunity object is changed, then Siebel CRM Desktop updates the OpportunityId field of the association that you specify in the left_link attribute. If you use the following tag in the siebel_basic_mapping.xml file for this field, then set the assoc_left_link_refresh_required attribute to true: `writer class="link_fields"`
assoc_right_link_refresh_required	If the channel partner object is changed, then Siebel CRM Desktop updates the ChannelPartnerId field of the association that you specify in the right_link attribute. If you use the following tag in the siebel_basic_mapping.xml file for this field, then set the assoc_right_link_refresh_required attribute to true: `writer class="link_fields"`

Code to Define a Salesbook Control

To define a salesbook control, you add the following code to the views.xml file:

```
<str key="channel_partner:salesbook">
    <![CDATA[<?xml version="1.0"?>
        <view type="table">
            <viewname>Phone List</viewname>
```

```
            <viewstyle>table-layout:fixed;width:100%;font-family:Segoe UI;font-
style:normal;font-weight:normal;font-size:8pt;color:Black;font-charset:0</
viewstyle>

            <viewtime>0</viewtime>

            <linecolor>8421504</linecolor>

            <linestyle>3</linestyle>

            <gridlines>1</gridlines>

            <newitemrow>1</newitemrow>

            <collapsestate/>

            <rowstyle>background-color:window;color:windowtext</rowstyle>

            <headerstyle>background-color:#D3D3D3</headerstyle>

            <previewstyle/>

            <arrangement>

                <autogroup>0</autogroup>

                <collapseclient/>

            </arrangement>

            <multiline>

                <width>0</width>

            </multiline>

            <column>

                <name>HREF</name>

                <prop>DAV:href</prop>

                <checkbox>1</checkbox>

            </column>

            <column>

                <heading>Last Name</heading>

                <prop>urn:schemas:contacts:sn</prop>

                <type>string</type>

                <width>322</width>

                <style>padding-left:3px;;text-align:left</style>

                <editable>1</editable>
```

```
            </column>
            <column>
                <type>string</type>
                <heading>Location</heading>
               <prop>http://schemas.microsoft.com/mapi/string/{00020329-0000-0000-
C000-000000000046}/sbl%20Location</prop>
                <width>322</width>
                <style>padding-left:3px;;text-align:left</style>
                <editable>1</editable>
                <userheading>Location</userheading>
            </column>
            <orderby>
                <order>
                    <heading>File As</heading>
                    <prop>urn:schemas:contacts:fileas</prop>
                    <type>string</type>
                    <sort>asc</sort>
                </order>
            </orderby>
            <groupbydefault>0</groupbydefault>
            <previewpane>
                <markasread>0</markasread>
            </previewpane>
        </view>]]>
</str>
```

A How Siebel CRM Desktop Maps Fields Between Siebel CRM Data and Microsoft Outlook Data

This appendix describes how Siebel CRM Desktop maps fields between Siebel CRM data and Microsoft Outlook data. It includes the following topics:

- How Siebel CRM Desktop Maps Fields Between Siebel Activities and Outlook Calendar on page 229
- How Siebel CRM Desktop Maps Fields Between Siebel Activities and Outlook Tasks on page 232
- How Siebel CRM Desktop Maps Fields Between a Siebel CRM Activity and an Outlook Email on page 236
- How Siebel CRM Desktop Transforms Objects Between Siebel CRM Data and Microsoft Outlook Data on page 238

How Siebel CRM Desktop Maps Fields Between Siebel Activities and Outlook Calendar

Table 22 describes how Siebel CRM Desktop maps objects between a Siebel CRM activity and Outlook Calendar.

Table 22. How Siebel CRM Desktop Maps Fields Between Siebel Activities and Outlook Calendar

Siebel Field	Outlook Field	Required	Comments
Owner	For more information, see "How Siebel CRM Assigns the Meeting Organizer" on page 42.	Yes	Not applicable.
PIM Meeting Participants	List of participants for the appointment, delimited by a semicolon (;).	No	Not applicable.
Employees	List of employees that is received from the list of participants when Siebel CRM Desktop resolves the primary email of the employees.	No	Siebel CRM Desktop performs the preliminary resolution when an appointment is saved. To improve the quality of this list, after the next synchronization, the Siebel Server analyzes the processing that occurs on the server.

Table 22. How Siebel CRM Desktop Maps Fields Between Siebel Activities and Outlook Calendar

Siebel Field	Outlook Field	Required	Comments
Contacts	List of contacts that is received from the list of participants that remain after employees are removed from this list when Siebel CRM Desktop resolves the primary email of the contacts.	No	The logic that applies for the Employees field also applies for the Contacts field.
Type	Type that the user specifies when linking the CRM record. The default value is Appointment.	Yes	The user can choose the activity type in the form of the Outlook calendar item only for an activity that contains the following *Display in* value: Calendar and Activities The customization package defines the default value. You can change this value.
Description	For more information, see "How Siebel CRM Desktop Handles an Item That is Marked Private" on page 231.	No	The value that is set for a private task is defined in the customization package and is included in the localization resources.
Priority Values are: ■ 1-ASAP ■ 2-High ■ 3-Medium ■ 4-Low	Outlook priority Values are: ■ 1-High ■ 2-High ■ 3-Medium ■ 4-Low	No	For more information, see "How Siebel CRM Desktop Maps the Priority Field" on page 232.
Comments	Description A private calendar item does not include a description.	No	Not applicable.
Account	Account association that the user specifies when the user links a CRM record.	No	Not applicable.
Opportunity	Opportunity association that the user specifies when the user links a CRM record.	No	Not applicable.
Created by	User ID of the employee who creates the activity.	Yes	Not applicable.

Table 22. How Siebel CRM Desktop Maps Fields Between Siebel Activities and Outlook Calendar

Siebel Field	Outlook Field	Required	Comments
Created Date	Timestamp of when the activity is created.	Yes	Not applicable.
Planned	Start Date/Time	Yes	Not applicable.
Planned Completion/ End time	End Date/Time	Yes	Not applicable.
Duration	Duration	Yes	Not applicable.
Meeting Location	Location	No	Not applicable.
Private	Private	No	Not applicable.
Repeat	Recurring flag	No	Not applicable.
Repeat Until	Recurrence Range End	No	Not applicable.
Repeat Frequency	For recurrence, Siebel CRM Desktop uses a binary field that combines the data from a number of Microsoft Outlook fields. The frequency is daily, weekly, monthly, or yearly.	No	Not applicable.

How Siebel CRM Desktop Handles an Item That is Marked Private

Siebel CRM Desktop handles an item that is marked Private in the following ways:

- From Microsoft Outlook to Siebel CRM. Assume the user shares a meeting with Siebel CRM and also adds a check mark to the Private check box to make this meeting private. In this situation, Siebel CRM Desktop synchronizes the meeting and the setting of the Private check box to the Siebel Server. It performs this synchronization in the same way that it synchronizes any other field. Siebel CRM does not display the contents of this meeting to any user who interacts with the Siebel Server except the meeting owner.
- From Siebel CRM to Microsoft Outlook**.** If the user is directly associated with an activity, then a filter on the action object causes Siebel CRM Desktop to synchronize this activity to Microsoft Outlook even if the Private check box for this field contains a check mark. For example, assume the Private check box in an activity for an account contains a check mark. In this situation, Siebel CRM Desktop does the following:
  - Downloads and displays this activity for the user who is associated with it.
  - Does not download or display this activity to any user who is not associated with this activity record, even if this user is on the Account Team.

How Siebel CRM Desktop Maps the Priority Field

Siebel CRM data includes an ASAP priority value but Microsoft Outlook data does not. Because it is not possible to save the ASAP value during synchronization, Siebel CRM Desktop does not include it in the mapping. If a Siebel task is marked 1-ASAP, then Siebel CRM Desktop creates a mapping error.

A Microsoft Outlook activity includes another field to store the Priority value that is derived from the Siebel CRM data. Note the following behavior:

- If a Microsoft Outlook item is updated, and if the Priority of the Outlook item is High, then Siebel CRM Desktop validates the stored value.
- If the value that is stored for the activity is ASAP, then Siebel CRM Desktop uses the ASAP value from the activity.
- If the value that is stored for the activity is not ASAP, then Siebel CRM Desktop uses the value from the Microsoft Outlook record.

How Siebel CRM Desktop Maps Fields Between Siebel Activities and Outlook Tasks

This topic describes how Siebel CRM Desktop maps fields between a Siebel CRM activity and a Microsoft Outlook task. Table 23 describes how Siebel CRM Desktop maps fields between a Siebel CRM activity and an Outlook task.

Table 23. How Siebel CRM Desktop Maps Fields Between Siebel Activities and Outlook Tasks

Siebel Field	Outlook Field	Required	Description
Owner	Owner	Yes	For more information, see "How Siebel CRM Desktop Maps the Owner Field Between a Siebel CRM Activity and an Outlook Task" on page 234.
Type	The type that the user specifies when the user links a CRM record. The default value is To Do.	Yes	If a user shares a task, then Siebel CRM Desktop creates an activity in Microsoft Outlook with a *Display in* value that includes the following: To Do and Activities The customization package defines the default value. You can change this value.

Table 23. How Siebel CRM Desktop Maps Fields Between Siebel Activities and Outlook Tasks

Siebel Field	Outlook Field	Required	Description
Description	For more information, see "How Siebel CRM Desktop Handles an Item That is Marked Private" on page 231.	No	The customization package defines the value that is set for a private task. Localization resources include this value.
Priority Values are: ■ 1-ASAP ■ 2-High ■ 3-Medium ■ 4-Low	Outlook priority Values are: ■ 1-High ■ 2-High ■ 3-Medium ■ 4-Low	No	For more information, see "How Siebel CRM Desktop Maps the Priority Field" on page 232.
Comments	Description A private task does not include a description.	No	If the Private check box is set to allow a shared task, then Siebel CRM Desktop sets the Comments field to a value that the customization package defines, such as Unavailable.
Start Date	Start Date	No	The Start Date for an activity is the start date that the user sets for a task. If the user does not enter the start date, then the start date for the activity is also empty.
Done	Completed Date	No	The Done date is the date that Siebel CRM displays as the Actual End date in the Siebel Web Client.
Completed flag	Completed flag	No	Not applicable.
Due	Due Date	No	Not applicable.
Percent complete	Percent complete	No	The value of the Percent complete field is related to the status. For more information, see "How Siebel CRM Desktop Maps the Status Field of an Activity" on page 235.
Status	Status	No	For more information, see "How Siebel CRM Desktop Maps the Status Field of an Activity" on page 235.

Table 23. How Siebel CRM Desktop Maps Fields Between Siebel Activities and Outlook Tasks

Siebel Field	Outlook Field	Required	Description
Account	Account association that the user specifies when the user links a CRM record.	No	Not applicable.
Opportunity	Opportunity association that the user specifies when the user links a CRM record.	No	Not applicable.
Contacts	Contact association that the user specifies when the user links a CRM record.	No	Not applicable.
Created by	User ID of the employee who creates the activity.	Yes	Not applicable.
Created Date	Timestamp of when the activity is created.	Yes	Not applicable.

How Siebel CRM Desktop Maps the Owner Field Between a Siebel CRM Activity and an Outlook Task

This topic describes how Siebel CRM Desktop maps the owner field between a Siebel CRM activity and an Outlook task. The following logic applies:

- If a shared task is assigned to another employee, then Siebel CRM Desktop sets the assignee as the owner of the activity.
- If a shared task is assigned to a number of employees, then Siebel CRM Desktop sets each assignee as the owner of their own activity.
- If a shared task is assigned to a shared contact or to an external person, then Siebel CRM Desktop sets the employee who created the task as the owner of the activity.
- If a task is created by an external person and assigned to an employee, and if this employee shares the task, then Siebel CRM Desktop sets the employee as the owner of the activity.
- If an external person who is not a Siebel employee sends a task to a number of employees who are Siebel CRM Desktop users, then Siebel CRM Desktop sets each employee as the owner of the activity and adds the remaining employees to the employee team. Siebel CRM Desktop performs this work after each employee accepts and shares the task.

How Siebel CRM Desktop Maps the Status Field of an Activity

Table 24 describes how Siebel CRM Desktop maps the status of a Siebel CRM activity to the status of an Outlook task when the user shares a task.

Table 24. How Siebel CRM Desktop Maps the Status Field of an Activity

Siebel Status	Outlook Status
Not Started	Not Started
In Progress	In Progress
Done	Completed
On Hold	Waiting on someone else
Canceled	Deferred

Table 25 describes how Siebel CRM Desktop maps the Status field of a Siebel CRM activity when the value of the Display In field of a To Do activity contains the following:

To Do and Activities

Siebel CRM Desktop performs this work during synchronization.

Table 25. How Siebel CRM Desktop Maps the Status of a Siebel CRM Activity When To Do Contains *To Do and Activities*

Siebel Status	Outlook Status
Not Started/Acknowledged	Not Started
In Progress	In Progress
Done	Completed
On Hold	Waiting on someone else
Canceled/Declined	Deferred
Any other value	The logic described in Table 26 on page 236 determines the value for the status.

Table 26 describes how Siebel CRM Desktop uses the Percent Complete value in an Outlook task to determine the value of the Status field.

Table 26. How Siebel CRM Desktop Maps the Status of a Siebel CRM Activity That Is Determined By Percent Complete

Siebel Status	Value of Percent Complete for the Outlook Task
Not Started	0
In Progress	Greater than zero and less than one hundred
Completed	100

Scenario for Mapping the Status Field of an Activity

This topic gives one example of how Siebel CRM Desktop maps the status field of an activity. Siebel CRM Desktop performs the following sequence:

1. A user creates a shared task in Microsoft Outlook that contains a status that is In Progress and a Percent Complete that is 0.
2. Siebel CRM Desktop creates an activity on the Siebel Server that contains a status of In Progress.
3. The user synchronizes with the Siebel Server.
4. On the Siebel Server, Siebel CRM Desktop changes the status to Requested.
5. The user synchronizes with the Siebel Server again.
6. In Microsoft Outlook, Siebel CRM Desktop updates the status of the activity to Requested.
7. At this point, the Percent Complete field of the task is 0. Siebel CRM Desktop updates the status of the task to Not Started.

How Siebel CRM Desktop Maps Fields Between a Siebel CRM Activity and an Outlook Email

Table 27 describes how Siebel CRM Desktop maps fields between a Siebel CRM activity and an Outlook email.

Table 27. How Siebel CRM Desktop Maps Fields Between a Siebel CRM Activity and an Outlook Email

Siebel Field	Required	Outlook Field
Type	Yes	One of the following: ■ If the user is the receiver, then Email - Inbound ■ If the user is the sender, then Email - Outbound
Description	No	Email Subject

Table 27. How Siebel CRM Desktop Maps Fields Between a Siebel CRM Activity and an Outlook Email

Siebel Field	Required	Outlook Field
Display	Yes	Changes to Communication and Activities, which is the default.
Email Attachment Flag	No	Changes to Y, which is the default.
Email BCC Line	No	Email BCC Line
Email Body	No	Email Body
Email CC Line	No	Email CC Line
Email Sender Address	No	Siebel CRM Desktop maps part of the value in the From Line to the Email Address. For example, assume the From Address includes the following address: Jane Smith, or Jane Smith <jane.smith@pcscomputing.com> In this example, Siebel CRM Desktop resolves the Email Sender Address to the following address: jane.smith@pcscomputing.com
Email Sender Name	No	Siebel CRM Desktop maps part of the From Line to the Display Name. For example, using the example from the Email Sender Address, the Email Sender Name resolves to Jane Smith.
Email To Line	No	Email To Line.
Priority	No	Importance. Siebel CRM Desktop applies the following mapping: ■ 1-ASAP in Siebel CRM data is ASAP in Microsoft Outlook ■ 2-High in Siebel CRM data is High in Microsoft Outlook ■ 3-Medium in Siebel CRM data is Normal in Microsoft Outlook ■ 4-Low in Siebel CRM data is Low in Microsoft Outlook For more information, see "How Siebel CRM Desktop Maps the Priority Field" on page 232.
Account Id	No	The Primary account association that the user specifies when the user creates the activity.
Opportunity Id	No	The opportunity association that the user specifies when the user creates the activity.
Attachment	No	The original email that Siebel CRM Desktop saves as an attachment to the activity.

How Siebel CRM Desktop Transforms Objects Between Siebel CRM Data and Microsoft Outlook Data

This topic describes how Siebel CRM Desktop transforms objects between Siebel CRM Data and Microsoft Outlook data. It includes the following topics:

- "How Siebel CRM Desktop Transforms a Calendar Event That Does Not Recur" on page 238
- "How Siebel CRM Desktop Transforms a Recurring Event That Matches a Siebel Recurrence Pattern" on page 239
- "How Siebel CRM Desktop Transforms a Recurrent Event That Does Not Match a Siebel Recurrence Pattern" on page 241
- "How Siebel CRM Desktop Transforms a Siebel CRM Activity That Is Not Repeated" on page 242
- "How Siebel CRM Desktop Transforms a Siebel CRM Activity That Is Repeated" on page 242
- "How Siebel CRM Desktop Maps Fields Between Certain Siebel Appointments and Microsoft Outlook Appointments" on page 244

Siebel CRM Desktop runs the following logic when the user synchronizes a recurring activity between the Siebel Server and Microsoft Outlook:

1. Maps changes between Siebel activities in Microsoft Outlook and native Outlook items.

   Siebel CRM Desktop updates changes in the Outlook calendar event in the activity that is linked to the Calendar event.

2. Synchronizes changes between Siebel activities in Microsoft Outlook and Siebel CRM activities.

   The change is updated in Siebel CRM data after the synchronization.

How Siebel CRM Desktop Transforms a Calendar Event That Does Not Recur

Table 28 describes how Siebel CRM Desktop transforms a calendar event in Microsoft Outlook that does not recur.

Table 28. How Siebel CRM Desktop Transforms a Calendar Event That Does Not Recur

Action in Microsoft Outlook	Work That Siebel CRM Desktop Performs
The user creates an activity that does not recur in Microsoft Outlook.	Creates a corresponding Siebel CRM activity.
The user changes an activity that does not recur in Microsoft Outlook.	Changes the corresponding Siebel CRM activity.
The user deletes an activity that does not recur in Microsoft Outlook.	Deletes the corresponding Siebel CRM activity.

Table 28. How Siebel CRM Desktop Transforms a Calendar Event That Does Not Recur

Action in Microsoft Outlook	Work That Siebel CRM Desktop Performs
The user changes an activity that does not recur in Microsoft Outlook to a recurring event that matches the Siebel recurrence pattern.	Changes the corresponding Siebel CRM activity.
The user changes an activity that does not recur in Microsoft Outlook to a recurring event that does not match the Siebel recurrence pattern.	Performs the following work: ■ Changes the corresponding activity. Siebel CRM Desktop uses the Siebel recurrence pattern that contains the longest interval between occurrences and that can incorporate all occurrences of the chosen Microsoft Outlook pattern. ■ To match the calendars in a Siebel application and in Microsoft Outlook, Siebel CRM Desktop identifies the list of exceptions for the activity.

How Siebel CRM Desktop Transforms a Recurring Event That Matches a Siebel Recurrence Pattern

Table 29 describes how Siebel CRM Desktop transforms a recurring event that matches a Siebel recurrence pattern.

Table 29. How Siebel CRM Desktop Transforms a Recurring Event That Matches a Siebel Recurrence Pattern

Action in Microsoft Outlook	Work That Siebel CRM Desktop Performs
The user creates a recurring event that matches a Siebel recurrence pattern.	Creates a corresponding Siebel CRM recurrent activity.
The user changes a single occurrence for a recurrent event that matches a Siebel recurrence pattern.	Performs the following work in Microsoft Outlook: ■ Adds the date of the exception that changed to the exceptions list for the target activity. ■ Adds the new, nonrecurrent activity to the newly created calendar item in Microsoft Outlook.
The user changes a recurrent event that matches a Siebel recurrence pattern.	Changes the corresponding Siebel CRM recurrent activity.

Table 29. How Siebel CRM Desktop Transforms a Recurring Event That Matches a Siebel Recurrence Pattern

Action in Microsoft Outlook	Work That Siebel CRM Desktop Performs
The user deletes a single occurrence for a recurrent event that matches a Siebel recurrence pattern.	Performs the following work in Microsoft Outlook: ■ Adds the date of the exception to the exceptions list for the target activity. ■ Deletes this single occurrence of the activity.
The user changes a recurrent event that matches a Siebel recurrence pattern into an event that does not recur.	Performs the following work in Microsoft Outlook: ■ Changes the corresponding Siebel CRM activity to an activity that does not recur.
The user changes a recurrent event that matches a Siebel recurrence pattern into a recurrent event that does not match any Siebel recurrence patterns.	Performs the following work in Microsoft Outlook: ■ Changes the corresponding Siebel CRM activity. Siebel CRM Desktop uses the Siebel recurrence pattern that contains the longest interval between occurrences and that can incorporate all occurrences of the chosen Outlook pattern. ■ To match Microsoft Outlook and Siebel calendars, Siebel CRM Desktop identifies the list of exceptions for this activity.

How Siebel CRM Desktop Transforms a Recurrent Event That Does Not Match a Siebel Recurrence Pattern

Table 30 describes how Siebel CRM Desktop transforms a recurrent event that does not match a Siebel recurrence pattern.

Table 30. How Siebel CRM Desktop Transforms a Recurrent Event That Does Not Match a Siebel Recurrence Pattern

Action in Microsoft Outlook	Work That Siebel CRM Desktop Performs
The user creates a recurrent event that does not match a Siebel recurrence pattern.	Performs the following work in Microsoft Outlook: ■ Creates the corresponding activity. To identify this activity, Siebel CRM Desktop uses the Siebel recurrence pattern that contains the longest interval between occurrences and that can incorporate all occurrences of the chosen Outlook pattern. ■ To match the calendars in Microsoft Outlook and Siebel CRM Desktop, creates the list of exceptions for this activity.
The user changes a single occurrence for a recurrent event that does not match any Siebel recurrence pattern.	Performs the following work in the Outlook Calendar: ■ Creates a new calendar item that is nonrecurrent. ■ Deletes the occurrence that changed for the Outlook calendar item. ■ Creates an exception in Microsoft Outlook for the occurrence date that changed. Performs the following work in Microsoft Outlook: ■ Adds an exception to the exceptions dates list of the activity. ■ Adds a new activity that does not recur.
The user deletes a single occurrence for a recurrent event that does not match a Siebel recurrence pattern.	Performs the following work in Microsoft Outlook: ■ Adds an exception to the exceptions dates list of the activity. ■ Deletes this single occurrence of the activity.
The user deletes a recurrent event that does not match a Siebel recurrence pattern.	Deletes the corresponding recurrent activity.

How Siebel CRM Desktop Transforms a Siebel CRM Activity That Is Not Repeated

Table 31 describes how Siebel CRM Desktop transforms a Siebel CRM activity that is not repeated.

Table 31. How Siebel CRM Desktop Transforms a Siebel CRM Activity That Is Not Repeated

Work That Occurs in a Siebel Application	Work That Siebel CRM Desktop Performs
The user creates a Siebel CRM activity that is not repeated.	Creates the corresponding Outlook activity that does not recur.
The user changes a Siebel CRM activity that is not repeated.	Changes the corresponding Outlook activity that does not recur.
The user changes a Siebel CRM activity that is not repeated to an activity that is repeated.	
The user deletes a Siebel CRM activity that is not repeated.	Deletes the corresponding Outlook activity that does not recur.

How Siebel CRM Desktop Transforms a Siebel CRM Activity That Is Repeated

Table 32 describes how Siebel CRM Desktop transforms an activity in a Siebel application that is repeated.

Table 32. How Siebel CRM Desktop Transforms a Repeated Activity in a Siebel Application

Work That Occurs in a Siebel Application	Work That Siebel CRM Desktop Performs
The user creates a recurrent activity in a Siebel application.	Creates a corresponding recurrent event in Microsoft Outlook.

Table 32. How Siebel CRM Desktop Transforms a Repeated Activity in a Siebel Application

Work That Occurs in a Siebel Application	Work That Siebel CRM Desktop Performs
The user changes a single occurrence of the recurring activity in Siebel CRM: ■ Creates a delete activity. ■ Creates a new activity that does not recur.	Deletes the occurrence of the Outlook calendar event for the date of the occurrence that is deleted in Siebel CRM.
The user can modify a recurring activity in Siebel CRM. Because the activity was created in Microsoft Outlook from a recurring calendar event, the activity can originate in Siebel CRM or Siebel CRM Desktop can synchronize it to Siebel CRM from Microsoft Outlook. If the user changes all instances of the recurring activity in Siebel CRM, then Siebel CRM Desktop deletes every instance of the recurring activity from the current day forward. Any instances before the current day remain on the calendar. If the user deletes only one instance, then Siebel CRM still schedules every other instance. If the user changes a recurring Siebel CRM activity, then Siebel CRM Desktop does the following work: ■ Changes the following date of the initial activity to the date minus one occurrence: repeat until ■ Creates a new repeated event that the date of the change for the repeat determines. This situation is true until the date is the following date of the parent repeated event: repeat until ■ Relinks the delete records to the corresponding recurring activities, depending on the exception date.	Performs the following work: ■ Changes the end date of the initial recurring activity ■ Creates a new recurring activity that the activity that was added in a Siebel application determines ■ Changes the links for the delete activities depending on the changes that are downloaded from a Siebel application

How Siebel CRM Desktop Maps Fields Between Certain Siebel Appointments and Microsoft Outlook Appointments

Table 33 describes how Siebel CRM Desktop maps certain fields between a Siebel appointment that is repeated and an Outlook appointment that recurs.

Table 33. How Siebel CRM Desktop Maps Fields Between Certain Siebel Appointments and Outlook Appointments

Siebel CRM		Microsoft Outlook		
Frequency	**Start and End Date**	**Frequency**	**Occurrence**	**Start and End Date**
Daily	Start Date Repeat Until	Daily	Every 1 day	Start is Start date End by is Repeat Until
Weekly	Start Date Repeat Until	Weekly	Every 1 week Weekday is the weekday of the Siebel Start Date	Start is Start date End by is Repeat Until
Monthly	Start Date Repeat Until	Monthly	Every 1 month Day is day of Siebel Start Date	End by is Repeat Until
Quarterly	Start Date Repeat Until	Monthly	Every 3 months Day is day of Siebel Start Date	End by is Repeat Until
Yearly	Start Date Repeat Until	Yearly	Date is date of the Siebel Start Date	End by is Repeat Until

B XML Files Reference

This appendix describes the code in the XML files that Siebel CRM Desktop includes in the customization package. It includes the following topics:

- XML Code to Map a Field on page 245
- XML Code to Customize the Data That Siebel CRM Desktop Deletes if You Remove Siebel CRM Desktop on page 250
- XML Code to Customize Synchronization on page 251
- XML Code to Customize Forms on page 258
- XML Code to Customize Toolbars on page 273
- XML Code to Customize Dialog Boxes on page 275
- XML Code to Customize Views on page 276
- XML Code to Customize the SalesBook Control on page 277
- XML Code for Meta Information on page 279

XML Code to Map a Field

This topic describes the code of the siebel_basic_mapping.xml file. It includes the following topics:

- “Example Code of the siebel_basic_mapping.xml File” on page 246
- “Type Tag of the siebel_basic_mapping.xml File” on page 247
- “Form Tag of the siebel_basic_mapping.xml File” on page 247
- “Alt Message Classes Tag of the siebel_basic_mapping.xml File” on page 248
- “Custom Views Tag of the siebel_basic_mapping.xml File” on page 248
- “Field Tag of the siebel_basic_mapping.xml File” on page 249
- “Writer Tag of the siebel_basic_mapping.xml File” on page 249

For more information, see “Customizing How Siebel CRM Desktop Maps Fields” on page 148.

Example Code of the siebel_basic_mapping.xml File

This topic gives one example of code that Siebel CRM Desktop uses in the siebel_basic_mapping.xml file. You might use this feature differently, depending on your business model. The following code is an example of the siebel_basic_mapping.xml file:

```
<?xml version="1.0" encoding="utf-8"?>

<sd2_meta>

    <types>

        <type id="Contact" predefined_folder="10">

            <form message_class="IPM.Contact.SBL.Contact"
icon="type_image:Contact:16" large_icon="type_image:Contact:32"
display_name="Contact">SBL Contact</form>

            <alt_messageclasses>

                <alt_messageclass ext="Private" display_name="Private Contact"
icon="type_image:Contact.Private:16"
large_icon="type_image:Contact.Private:32">SBL Contact</alt_messageclass>

            </alt_messageclasses>

            <custom_views default_name="Siebel Contacts">

                <view id="all_contacts" name="Siebel Contacts"></view>

            </custom_views>

            <field id=" First Name ">

                <reader class="mapi_std">

                    <mapi_tag id="0x3A060000"></mapi_tag>

                    <convertor class="string"></convertor>

                </reader>

                <writer class="Microsoft Outlook_std">

                <Microsoft Outlook_field id=" FirstName "></Microsoft Outlook_field>

                    <convertor class="string"></convertor>

                </writer>

            </field>

            <field id="Location">

               <reader class="mapi_user">

                  <user_field id="sbl Location" ol_field_type="1"></user_field>
```

```
                    <convertor class="string"></convertor>
                </reader>
                <writer class="Microsoft Outlook_user">
                        <user_field id="sbl Location" ol_field_type="1"></user_field>
                        <convertor class="string"></convertor>
                </writer>
            </field>
        </type>
    </types>
</sd2_meta>
```

Type Tag of the siebel_basic_mapping.xml File

The type tag defines the Siebel CRM object that Siebel CRM Desktop maps to Microsoft Outlook.

Siebel CRM Desktop includes the following attributes in the type tag:

- **id.** Defines the ID or name of the Siebel CRM object that Siebel CRM Desktop maps to Microsoft Outlook.
- **display_name.** Defines the name of the folder that Siebel CRM Desktop displays in the Microsoft Outlook tree view that Siebel CRM Desktop uses to store the records of the Siebel CRM object.
- **folder_type.** Defines the type of the Microsoft Outlook folder. The value for the folder_type attribute is typically 10 for a custom Siebel CRM object.
- **hidden_folder.** Determines if the folder that the display_name attribute specifies is visible in the Microsoft Outlook tree view.
- **predefined_folder.** Defines the type of Microsoft Outlook folder. If you must store Siebel CRM objects in a native Microsoft Outlook folder, then Siebel CRM Desktop uses the predefined_folder attribute.
- **prohibit_user_modification.** Prohibits the modification to an object that is declared in the type tag. The default value is false.
- **ver.** Used during development. Siebel CRM Desktop does not apply any change to the description for the object that the id attribute defines until the value of the ver attribute is increased.

Form Tag of the siebel_basic_mapping.xml File

The form tag defines the ID of the form that Siebel CRM Desktop uses to display the object that is defined in the type tag. The form with the corresponding ID is described in the forms_xx.xml file. For more information, see "XML Code to Customize Forms" on page 258.

Siebel CRM Desktop includes the following attributes in the form tag:

- **message_class.** Defines the Microsoft Outlook message class for the form. The message class is an extension of native Microsoft Outlook message classes. For example, IPM.Contact.SBL.Contact, or IPM.Contact.SBL.Account.
- **icon.** Defines the icon that Siebel CRM Desktop uses to display the following objects:
  - The object that is defined in the type tag in a Microsoft Outlook table view.
  - The icon views for small icons and list view modes.
  - A form caption icon, which displays in the top-left corner in front of the form caption. Siebel CRM Desktop stores icons in the platform_images.xml file. The value of the icon attribute must be the key value of the required image.
- **large_icon.** Defines the icon to use to represent the object that is defined in the type tag when the user uses the large icons mode. Siebel CRM Desktop stores icons in the platform_images.xml file. The value of the large_icon attribute must be the key value of the necessary image.
- **display_name.** Defines the name that Siebel CRM Desktop displays on the form caption.

Alt Message Classes Tag of the siebel_basic_mapping.xml File

To define an alternative message class to an object in Microsoft Outlook, you can use the alt_messageclasses tag. This tag is useful if a CRM object might be in one of several different states. For example, a contact might be shared or not shared.

This tag must contain a set of alt_messageclass tags that define each state of an object. The alt_messageclass tag can also define a form that you use for an object with this message class. This way, you can use different forms for the same object, but in different states.

Each alt_messageclass tag includes the following attributes:

- **Ext.** Indicates an extension that is added to an original message class.
- **display_name.** The same as the message_class tag.
- **icon.** The same as the message_class tag.
- **large_icon.** The same as the message_class tag.

Custom Views Tag of the siebel_basic_mapping.xml File

The custom_views tag defines a set of custom Microsoft Outlook views that Siebel CRM Desktop applies for the object that is defined in the type tag. The default_name attribute sets the default view that Siebel CRM Desktop applies after the installation.

Siebel CRM Desktop includes the following tags in the custom_views tag:

- **view.** Describes each view. Each view tag includes the following attributes:
  - **id.** Defines the ID of the view. The view is described in the views.xml file.
  - **name.** Defines the name of the view that is specified in the id attribute to display in the Current view menu. To access this menu in Microsoft Outlook, the user chooses the View menu, Arrange By, and then the Current view menu.

The following code is an example of the custom_views tag:

```
<custom_views default_name="All Activities">

    <view id="all_activities" name="All Activities"></view>

   <view id="all_activities_by_duedate" name="Activities by Due Date"></view>

   <view id="all_activities_by_owner" name=" Activities by Owner"></view>

   <view id="all_activities_by_priority" name="Activities by Priority"></view>

</custom_views>
```

Field Tag of the siebel_basic_mapping.xml File

The field tag describes the field mapping. Because the field tag describes one field, the number of field tags must be the same as the number of fields that are mapped.

The following attributes are included in the field tag:

- **id.** Defines the field identifier. For example, the API name of the field. To assign a control to this field on a form, Siebel CRM Desktop also uses the value of this attribute in the forms_xx.xml file.
- **ver.** Used during development. Siebel CRM Desktop does not apply any change to the field description until the value of the ver attribute is increased.

Writer Tag of the siebel_basic_mapping.xml File

The writer tag defines write access to the field that Siebel CRM Desktop maps to Microsoft Outlook. It contains only the class attribute. The following values are available for the class attribute:

- binhex_link
- custom:links_first
- multiwriter
- Microsoft Outlook_document_content
- Microsoft Outlook_document_filename
- Microsoft Outlook_recipients
- Microsoft Outlook_std

- Microsoft Outlook_user

The writer tag can contain the following tags:

- link_writer
- resolved_writer
- Microsoft Outlook_field
- user_field
- convertor
- writer

XML Code to Customize the Data That Siebel CRM Desktop Deletes if You Remove Siebel CRM Desktop

This topic gives one example of code that Siebel CRM Desktop uses in the platform_configuration.xml file. You might use this feature differently, depending on your business model. For more information, see "Customizing Which CRM Data Siebel CRM Desktop Removes if the User Removes Siebel CRM Desktop" on page 149.

The following code is an example of the platform_configuration.xml file:

```
<platform>

<items_remover>

<rules>

<type id="Mail" rule="skip" />

<type id="Task" rule="skip" />

<type id="Event" rule="skip" />

<type id="Action" rule="script" language="JScript">

<![CDATA[


                //"Calendar and Activities";

                //"To Do and Activities";

                //"Activities Only";

                var pim_id = item.PIMObjectId;

                if (!item["Appt PIM Flag"] && pim_id != null)

                {
```

```
                    var pim_item = open_item(pim_id);
                    if (pim_item != null)
                       pim_item.remove();
                 }

              true;// allow to process this item

]]>
</type>
</rules>
</items_remover>
</platform>
```

XML Code to Customize Synchronization

This topic describes the code of the connector_configuration.xml file. It includes the following topics:

- "Example Code for the connector_configuration.xml File" on page 252
- "Types Tag of the connector_configuration.xml File" on page 253
- "Type Tag of the connector_configuration.xml File" on page 253
- "View Tag of the connector_configuration.xml File" on page 253
- "Synchronizer Tag of the connector_configuration.xml File" on page 254
- "Links Tag of the connector_configuration.xml File" on page 254
- "Natural Key Tag of the connector_configuration.xml File" on page 255
- "Filter Presets Tag of the connector_configuration.xml File" on page 255

For more information, see "Customizing Synchronization" on page 149.

Example Code for the connector_configuration.xml File

This topic gives one example of code that Siebel CRM Desktop uses in the connector_configuration.xml file. You might use this feature differently, depending on your business model. The following code is an example of the connector_configuration.xml file:

```
<root>
    <types>
        <type id="Opportunity">
            <view label="Opportunity" label_plural="Opportunities"
small_icon="type_image:Opportunity:16" normal_icon="type_image:Opportunity:24"
large_icon="type_image:Opportunity:48"></view>
            <synchronizer name_format=":[:(Name):]">
                <links>
                    <link>Account Id</link>
                    <link>Currency Code</link>
                </links>
                <natural_keys>
                    <natural_key>
                        <field>Name</field>
                    </natural_key>
                </natural_keys>
            </synchronizer>
        </type>
    </types>
    <filter_presets>
        <preset name="Test filters">
            <type id="Action">
                <group link="and">
                    <binary field="Planned" condition="ge">
                        <value type="function">today</value>
                    </binary>
                </group>
```

```
            </type>
        </preset>
    </filter_presets>
</root>
```

Types Tag of the connector_configuration.xml File

The types tag describes the types of objects to synchronize. It does not contain attributes. It does contain a set of type tags. You must describe these types in the siebel_basic_mapping.xml file.

Type Tag of the connector_configuration.xml File

The type tag describes the type to synchronize. You must describe it in the siebel_basic_mapping.xml file. It includes the id attribute, which defines the ID of the object.

The type tag must contain the following tags:

- view
- synchronizer

View Tag of the connector_configuration.xml File

The view tag defines the type in the user interface, in particular, the Filter Records tab on the Synchronization Control Panel dialog box.

The view tag includes the following attributes:

- **label.** Label that Siebel CRM Desktop uses to display this object in the Synchronization Control Panel dialog box if Siebel CRM Desktop cannot resolve the name of this object.
- **label_plural**. Label that Siebel CRM Desktop uses to display this object in the Synchronization Control Panel dialog box if the label is most appropriately displayed in plural.
- **small icon**. Defines a 16-by-16 pixel icon that Siebel CRM Desktop uses for this object on the Synchronization Control Panel dialog box.
- **normal_icon**. Defines the icon that Siebel CRM Desktop displays next to the object type in the Filter Records tab of the Synchronization Control Panel dialog box.
- **large_icon**. Defines the icon that Siebel CRM Desktop displays on the Siebel CRM Desktop Synchronization dialog box while Siebel CRM Desktop synchronizes this type of object.
- **suppress_sync_ui**. If suppress_sync_ui is true, then this attribute hides object of this type from the Filter Records tab of the Synchronization Control Panel dialog box. If suppress_sync_ui is not defined, then Siebel CRM Desktop applies the false value by default.

For more information, see “Controlling the Object Types That Siebel CRM Desktop Displays in the Filter Records Tab” on page 114.

Synchronizer Tag of the connector_configuration.xml File

The synchronizer tag describes more settings for synchronization. It defines attributes that the Synchronization Engine requires. For example, relations between objects or criteria to identify duplicate objects.

The synchronizer tag includes the name_format attribute, which defines the format of the output string for objects of this type. Siebel CRM Desktop uses this string if objects of this type are displayed on the Check Issues, Resolve Conflicts, Resolve Duplicates, or Confirm Synchronization tab of the Control Panel in the Siebel CRM Desktop application.

The synchronizer tag can contain the following tags:

- links
- natural_keys

Links Tag of the connector_configuration.xml File

The links tag describes the references between types. You must define it in the synchronizer tag. Note the following requirements:

- You must use the links tag to describe all fields that Siebel CRM Desktop uses to store references between objects. A link field is an example of a field that Siebel CRM Desktop uses to store a reference between objects.
- You must describe all links in the links tag.

The links and link tags do not contain attributes.

Example Code of the Links Tag

The following code is an example of the links tag:

```
<links>
    <link>Account Id</link>
    <link>Opportunity Id</link>
    <link>Created By</link>
    <link>Primary Owner Id</link>
</links>
```

Natural Key Tag of the connector_configuration.xml File

The natural_key tag is defined in the synchronizer tag. You use it to configure criteria to identify duplicated records during synchronization. The natural_key tag contains the following:

- A set of natural_key tags that describes the criteria. Siebel CRM Desktop uses OR logic for all criteria that the natural_key tag describes.
- A set of field tags, each of which contains a field name that Siebel CRM Desktop examines to identify duplicates. Siebel CRM Desktop uses AND logic for all field tags.

Example Code of the Natural Key Tag

The following code is an example of the natural_keys tag:

```
<natural_keys>
    <natural_key>
        <field>First Name</field>
        <field>Last Name</field>
    </natural_key>
    <natural_key>
        <field>Email Address</field>
    </natural_key>
</natural_keys>
```

In this code, two objects are detected as duplicates if one of the following situations is true:

- First Name AND Last Name contain the same values
- Email Address fields contain the same values

Filter Presets Tag of the connector_configuration.xml File

The filter_presets tag contains predefined filter criteria. The preset tag describes this criteria. The preset tag includes a single attribute, name, which defines the name for this criteria. The preset tag contains a set of type tags that define filter criteria for each type.

The type tag defines the object type to which Siebel CRM Desktop applies this filter criteria. You must define the object type in the id attribute of this tag. The group tag describes a group of criteria.

Example Code of the Filter Presets Tag

The following code is an example usage of the filter_presets tag of the connector_configuration.xml file:

```
<filter_presets>
    <preset name="Test filters">
        <type id="Opportunity">
            <group link="and">
                <binary field="Status" condition="in">
                    <value type="array">
                        <value type="string">Accepted</value>
                        <value type="string">Pending</value>
                        <value type="string">Rejected</value>
                        <value type="string">Rerouted</value>
                    </value>
                </binary>
            </group>
        </type>
        <type id="Action">
            <group link="and">
                <binary field="Planned" condition="ge">
                    <value type="function">today</value>
                </binary>
            </group>
        </type>
    </preset>
</filter_presets>
```

Example Code to Set the Size and Type of Field

This topic describes code you can use to set the size and type of field. For more information, see "Regulating the Size and Type of Synchronized Records" on page 130. The following code is an example usage of the group tag of the connector_configuration.xml file to set the size and type of field:

```
<group link="and">
    <binary field="FileSize" condition="le">
    <value type="integer">5242880</value>
```

```
</binary>
<group link="or">
<binary field="FileExt" condition="eq">
<value type="string">doc</value>
</binary>
<binary field="FileExt" condition="eq">
<value type="string">docx</value>
</binary>
<binary field="FileExt" condition="eq">
<value type="string">xls</value>
</binary>
<binary field="FileExt" condition="eq">
<value type="string">xlsx</value>
</binary>
<binary field="FileExt" condition="eq">
<value type="string">msg</value>
</binary>
<binary field="FileExt" condition="eq">
<value type="string">txt</value>
</binary>
<binary field="FileExt" condition="eq">
<value type="string">rtf</value>
</binary>
<binary field="FileExt" condition="eq">
<value type="string">html</value>
</binary>
<binary field="FileExt" condition="eq">
<value type="string">ppt</value>
</binary>
<binary field="FileExt" condition="eq">
```

```
        <value type="string">pptx</value>
        </binary>
        <binary field="FileExt" condition="eq">
        <value type="string">pdf</value>
        </binary>
        <binary field="FileExt" condition="eq">
        <value type="string">mht</value>
        </binary>
        <binary field="FileExt" condition="eq">
        <value type="string">mpp</value>
        </binary>
        <binary field="FileExt" condition="eq">
        <value type="string">vsd</value>
        </binary>
        </group>
    </group>
```

XML Code to Customize Forms

This topic describes the code of the forms_11.xml and forms_12.xml files. It includes the following topics:

- "Form Tag of the forms_xx.xml File" on page 259
- "Validation Rules Tag of the forms_xx.xml File" on page 261
- "Script Tag of the forms_xx.xml File" on page 262
- "Info Bar Tag of the forms_xx.xml File" on page 263
- "Page Tag of the forms_xx.xml File" on page 263
- "Stack Tag of the forms_xx.xml File" on page 264
- "Control Tag of the forms_xx.xml File" on page 265
- "Types of Controls for the Control Tag of the forms_xx.xml File" on page 267

For more information, see "Customizing a Form" on page 150.

Form Tag of the forms_xx.xml File

The form tag describes the user interface for each custom Microsoft Outlook form. It can include the following attributes:

- **id.** Defines the unique name for the current form.
- **on_open**. JavaScript code that Siebel CRM Desktop runs if a user opens a form.
- **on_saving**. JavaScript code that Siebel CRM Desktop runs if a user attempts to save a form.
- **on_saved.** JavaScript code that Siebel CRM Desktop runs if a user saves a form.

The values for each attribute contain the name of the JavaScript function to run on a specific event. An example event is the event that occurs when a user opens a form.

The form tag contains the following tags:

- validation_rules
- script
- info_bar
- page

Each of these tags describe a specific part of the form. You can ignore each tag.

Example Code of the Form Tag

The following code is an example of the form tag. This code defines the Contact Note form:

```
<forms>
    <form id="SBL Contact Note" on_open="form_open()">
        <validation_rules>
           <rule message="Note Type is required." expression="item['Note Type'] != ''">
                <asserted_control id="NoteType"></asserted_control>
            </rule>
        </validation_rules>
        <script>
            <![CDATA[
                 function form_open()
                {
                      form.NoteType.required = true;
                }
```

```
            ]]>
        </script>
        <page id="General" tag="0x10A6" min_height="155" min_width="520">
            <cell>
                <stack layout="vert" padding="5">
                    <cell>
                        <stack layout="horz" spacing="3">
                            <cell size="65">
                                <stack layout="vert" spacing="5">
                                    <cell size="22">
                                        <control class="static" id="0x20000">
                                            <text>Type:</text>
                                        </control>
                                    </cell>
                                    <cell size="22">
                                        <control class="static" id="0x20002">
                                            <text>Description:</text>
                                        </control>
                                    </cell>
                                </stack>
                            </cell>
                            <cell>
                                <stack layout="vert" spacing="5">
                                    <cell size="22">
                                        <control class="combobox" id="NoteType"
tab_order="1">
                                            <source type="Contact.Contact_NoteNote
TypePicklist" field="Value" format=":[:(Label):]"></source>
                                            <field>Note Type</field>
                                        </control>
                                    </cell>
```

```
                                        <cell>
                                          <control id="0x103f" tab_order="2"></control>
                                        </cell>
                                    </stack>
                                </cell>
                            </stack>
                        </cell>
                    </stack>
                </cell>
            </page>
        </form>
    </forms>
```

Validation Rules Tag of the forms_xx.xml File

The validation_rules tag contains descriptions of validation rules. There is no limitation on the number of validation rules that you can define. Each rule is described in a rule tag.

The validation_rules tag can include the following attributes:

- **message.** Contains the message text as a value. If the validation rule that the rule tag describes fails, then Siebel CRM Desktop displays this message to the user.
- **expression.** Contains the validation expression as a value. If the expression returns a false value, then the validation rule fails.

The validation_rules tag can contain the following tags:

- **expression.** Performs the same function as the expression attribute. The only difference is that if you use the expression tag, then you must describe the validation expression in the CDATA section of the validation_rules tag.
- **asserted_control.** Defines a control to highlight if the validation rule fails. The ID of this control is defined in the id attribute of the asserted_control tag.

Example Code of the Validation Rules Tag

The following code is an example of the validation_rules tag:

```
<validation_rules>
    <rule message="Last Name is required." expression="item['Last Name'] != ''">
        <asserted_control id="0x1000"></asserted_control>
```

```
    </rule>
    <rule message="Business phone value format is incorrect.">
        <expression>
            <![CDATA[
                var phonetest = new RegExp(/\(?[0-9]{3}\)?[-. ]?[0-9]{3}[-. ]?[0-
9]{4}/);
               item["Work Phone #"] != '' ? (item["Work Phone #"].match(phonetest)
!= null ? true : false) : true;
            ]]>
        </expression>
    </rule>
</validation_rules>
```

Script Tag of the forms_xx.xml File

The script tag stores all JavaScript functions that Siebel CRM Desktop uses in the user interface of the current form. You can use these scripts for different purposes, as required. You must describe all script functions in the CDATA section of the validation_rules tag.

Example Code of the Script Tag

The following code is an example of the script tag:

```
<script>
    <![CDATA[
        function form_open()
        {
            form.NoteType.required = true;
        }
        function form_save()
        {
            if (form.item["Created"] == null)
            {
                form.item["Created"] = (new Date()).getVarDate();
            }
```

```
        }
    ]]>
</script>
```

Info Bar Tag of the forms_xx.xml File

The info_bar tag is an alternative to the page tag. You can use it to add a layout that the original form layout does not determine. You can use the info_bar tag to extend the original form, but not to modify the original form.

For example, the predefined email form includes an Info Bar. The Info Bar tag contains the Share With Siebel content that Siebel CRM Desktop displays if the user clicks the Share Bar. The Info Bar does not affect other parts of the form.

Note the following differences between how you can use the info_bar tag and the page tag:

- To customize only a section of a predefined Microsoft Outlook form, you can use the info_bar tab. For example, on the Mail, Task, or Calendar, forms.
- To customize an entire form, you can use the page tag. For example, the page tag completely replaces the native Contact form in Microsoft Outlook with a custom Siebel CRM Desktop form.

Page Tag of the forms_xx.xml File

The page tag describes the layout of the form. The layout is defined by a set of cell tags that contain the user interface elements and data that make up the form. Note the following requirements for the cell tag:

- A cell tag can be empty or can contain a stack of cell elements in the stack tag or a control.
- A cell tag can contain a user interface element or a piece of data from a native Microsoft Outlook object and from a standard or custom Siebel CRM object. If you must place more than one object in a cell, then you must use a stack tag in the cell tag.

A page tag can only contain one cell.

The page tag can include the following attributes:

- **id.** Contains the page name and is used nowhere else.
- **tag.** Contains the control ID of the first control on the page of a standard form. It is a page that Siebel CRM Desktop uses to apply a customized layout. For example, the first control on a General tab of a Contact form uses *id=0x402*, *Details page - 0?11cf*, and so forth.
- **min_height.** Describes the minimum height, in pixels.
- **min_width.** Describes the minimum width, in pixels.

Example Code of the Page Tag

The following code is an example of the page tag:

```
<page id="General" tag="0x0402">

    <cell>

    </cell>

</page>
```

Cell Tag of the Page Tag of the forms_xx.xml File

The cell tag is a layout cell that contains a stack of cells or a control. It defines the position of the stack. It includes the following attributes:

- **size.** Defines the cell size, in pixels. You must define the size if the cell is situated in a stack of cells:
  - In a stack of cells that is arranged horizontally, the size defines the cell width.
  - In a stack of cells that is arranged vertically, the size defines the height.
- **attraction.** Defines cell docking. You must define the attraction if the cell is in a stack of cells. The following values are available:
  - **near.** Cell docks to the left side of a horizontal stack of objects or to the top of a vertical stack of objects.
  - **far.** Cell docks to the right side of a horizontal stack of objects or to the bottom of a vertical stack of objects.
  - **both.** Cell consumes the entire free space. If more than one cell is set to *both*, then Siebel CRM Desktop divides the free space equally between all cells that are set to *both*.

  The default value is *near* if the cell size is defined. The default value is *both* if there is no size.

Stack Tag of the forms_xx.xml File

The stack tag defines a stack of cells that is placed in a cell. Siebel CRM Desktop docks a cell that contains the near attribute in the following ways:

- To the top border in a vertical stack
- To the left border in a horizontal stack

The stack tag includes the following attributes:

- **layout.** Defines the type of stack. Possible values include horz (horizontal) or vert (vertical).
- **spacing.** Defines the space between cells in a stack. The default value is 0.
- **padding.** Defines the space between the cell and the stack border. The cell defines the stack border. This cell contains the stack. The default value is 0.

Example Code of the Stack Tag

The following code is an example of the stack tag:

```
<page id="General" tag="0x0402">
    <cell>
        <stack layout="horz">
            <cell size="25"/><!--attraction="near"-->
            <cell size="30"/>
            <cell/><!--attraction="both"-->
            <cell attraction="both">
            <cell size="20" attraction="far">
   </stack>
    </cell>
</page>
```

Control Tag of the forms_xx.xml File

The control tag defines a control that is located in a cell. It includes the following attributes:

- **id.** Defines the control ID. If you use the id attribute to specify the ID of a native Microsoft Outlook control, then Siebel CRM Desktop requires no more attributes except tab_order, if necessary. The class attribute defines the control type.
- **tab_order.** Defines the tab order of the control. Siebel CRM Desktop displays the control that contains the smallest tab_order value as the first control in the tab order. You can start at 1. If the control contains no tab_order, then Siebel CRM Desktop does not include the control in the tab order.
- **class.** Defines the type of the control. Depending on the value of the class attribute, you must define more attributes and tags. For more information, see "Types of Controls for the Control Tag of the forms_xx.xml File" on page 267.
- **allow_negative.** For multi_currency control only.
- **caption.** For list control only.
- **type.** For Microsoft Outlook_view control only.
- **view_id.** For Microsoft Outlook_view control only.

The following controls support the control tag:

- button
- check box
- gradient_checkbox

The control tag includes the following tags. These tags depend on the value that is set for the class attribute of the parent control tag:

- **source.** Used for multivalue controls, such as combobox, lookup, and mvg_primary_selector. It defines the objects that Siebel CRM Desktop displays in this control. It includes the following attributes:
  - **type.** Defines a type of an object that Siebel CRM Desktop displays in the current control. Objects of this type must be described in the basic_mapping.xml file.
  - **display_format.** Defines what object fields Siebel CRM Desktop displays in a control. Used for the mvg_primary_selector control only.
  - **format.** The same as the display_format tag, but applied to other controls.
  - **field.** Defines what object field is chosen as a value in a combobox. Used for the combobox control only.
  - **left_id.** Defines a field on a related object where Siebel CRM Desktop stores the ID of the parent object. This is the ID of the current object on which the left_id control is used. Used for the mvg_primary_selector control only.
  - **item_value.** Defines a field on a related object where Siebel CRM Desktop stores the ID of the parent object. This is the ID of the chosen object. Used for the mvg_primary_selector control only.
- **text.** Defines the text that Siebel CRM Desktop uses for the control. You use this tag primarily for the label control.
- **field**. Contains the field identifier with which this control works:
  - This field tag must be described in the siebel_basic_mapping.xml file.
  - Contains one attribute, value, which Siebel CRM Desktop uses for edit controls only. This attribute describes the type of the value for the field.
  - The following values are available: string, binary, int, double, or currency.
  - The *API name* field is an example of a field identifier.

Example Code of the Control Tag

The following code is an example of the control tag:

```
<cell>

    <control id="0x10000" class="edit">

        <field value="string"> Name</field>

    </control>

</cell>
```

Types of Controls for the Control Tag of the forms_xx.xml File

This topic describes the types of controls you can configure for the control tag of the forms_xx.xml file. It includes the following topics:

- "Values of the Control Tag of the forms_xx.xml File" on page 267
- "Combobox Control of the forms_xx.xml File" on page 268
- "Dropdown Control of the forms_xx.xml File" on page 269
- "Lookup Control of the forms_xx.xml File" on page 269
- "Multicurrency Control of the forms_xx.xml File" on page 270
- "MVG Primary Selector Control of the forms_xx.xml File" on page 270
- "Microsoft Outlook View Control of the forms_xx.xml File" on page 271
- "Subform Control of the forms_xx.xml File" on page 271
- "Web_page Control of the forms_xx.xml File" on page 272

Values of the Control Tag of the forms_xx.xml File

Table 34 describes the values you can specify for the class attribute of the control tag of the forms_11.xml and forms_12.xml files.

Table 34. Values You Can Specify for the Class Attribute of the Control Tag

Value	Description
button	A button control. if you use this control, then it is not necessary to use a field tag.
checkbox	A check box control. The field that the field tag assigns to this control must include the following configuration: ■ The name of the field ■ The field type must be Boolean
combobox	A simple control that allows the user to choose any value from a list. For more information, see "Combobox Control of the forms_xx.xml File" on page 268.
datetime	A datetime control that allows the user to choose a date from a calendar. The field tag must contain the date or datetime field.
dropdown	A control that displays a menu when the user clicks the control. For example, if the user clicks Addresses on the contact form, then Siebel CRM Desktop displays Personal Addresses or Business Addresses. For more information, see "Dropdown Control of the forms_xx.xml File" on page 269.

Table 34. Values You Can Specify for the Class Attribute of the Control Tag

Value	Description
edge	A panel control that includes a border. If you use this control in a cell where the size is 1, such as with a separator, then it is not necessary to use a field tag.
edit	A simple edit box control.
gradient_checkbox	A control that behaves like a check box control, but uses a different graphical interface. This control displays on the Sharing bar.
lookup	A control that the user can use to choose any object. Siebel CRM Desktop uses a lookup control to establish a relationship between objects. For example, to link an account with a contact. For more information, see "Lookup Control of the forms_xx.xml File" on page 269.
multi_currency	A control that Siebel CRM Desktop uses to display the values from more than one field, such as price, revenue, and so forth. An example usage of the multi_currency control is where Siebel CRM Desktop must display the amount and currency values in a single field. For more information, see "Multicurrency Control of the forms_xx.xml File" on page 270.
mvg_primary_selector	A control that Siebel CRM Desktop uses to display the Primary association in a many-to-many (M:M) relationship. For more information, see "MVG Primary Selector Control of the forms_xx.xml File" on page 270.
Microsoft Outlook_view	A control that Siebel CRM Desktop uses to display a set of related items in the Microsoft Outlook view or form. For more information, see Microsoft Outlook View Control of the forms_xx.xml File on page 271.
static	A static control that you can use as a label on a form. If you use this control, then it is not necessary to use a field tag.
subform	A group of controls that you can use to display the fields of one object on the form of another object. For example, you can display a Siebel CRM activity on the native form for a Microsoft Outlook task or calendar item. For more information, see "Subform Control of the forms_xx.xml File" on page 271.

Combobox Control of the forms_xx.xml File

If you set the class attribute of the control tag to combobox, then you must define the following tags in the control tag:

- **source.** Describes the list values for this control. The source tag includes the following attributes:
  - **type.** Contains the ID of a list that the siebel_basic_mapping.xml file describes.
  - **field.** Contains the name of the field that Siebel CRM Desktop displays as a list value.
  - **format.** Defines the mask for this field output. Note that attributes are usually the same for all lists that drop down, which includes the following format:

```
field="Value"

format=":[:(Label):]"
```

Although Label is a variable, you must specify it as an absolute value, as described here.

- **field.** A field of an object that stores a value that the user chooses in a list.

Example Code of the Combobox Control

The following code illustrates use of the combobox control:

```
<control id="0x20105" class="combobox">

  <source type="ContactLeadSourcePicklist" field="Value" format=":[:(Label):]"/>

  <field>LeadSource</field>

</control>
```

Dropdown Control of the forms_xx.xml File

The dropdown control of the forms_xx.xml file is a button that includes menu options. If the user clicks this button, then Siebel CRM Desktop displays the menu. Menu items for this menu are added to scripts.

If you set the class attribute of the control tag to dropdown, then you must define the following tags in the control tag:

- **control.** Contains more tags that you can use to describe the dropdown control.
- **text.** The value in the control tag that you can use to define the text of the dropdown control.

You can use the caption attribute to define the caption for the dropdown control.

Example Code of the Dropdown Control

The following code illustrates use of the dropdown control:

```
<cell size="22">

  <control id="dd_contacts" class="dropdown" caption="#lbl_contacts" tab_order="1"
visible="false"></control>

</cell>
```

Lookup Control of the forms_xx.xml File

If you set the class attribute of the control tag to lookup, then you must define the following tags in the control tag:

- **source.** Describes the list values for this lookup control. The source tag includes the following attributes:
  - **type.** Contains the ID of an object that Siebel CRM Desktop uses in this control.

- **field.** Not used.
- **format.** Defines the mask for this field output. For example:

```
format=":[:(First Name) :]:[:(Last Name):]"
```

- **resource_id.** Defines the ID of the description for the lookup dialog box that the lookup_view_defs.xml file describes.

- **field.** The field of an object that stores the ID of the object that the user chooses in a lookup object.

Example Code of the Lookup Control

The following code illustrates use of the lookup control:

```
<control id="0x20100" class="lookup">

    <source type="All Items" format=":[:(FirstName) :]:[:(LastName):]"
resource_id="lookup:all_types"/>

    <source type="Contact" format=":[:(FirstName) :]:[:(LastName):]"
resource_id="lookup:contacts"/>

</control>
```

Multicurrency Control of the forms_xx.xml File

If you set the class attribute of the control tag to multi_currency, then you must define the following tags in the control tag:

- **value_field.** Contains the amount field name.
- **currency_field.** Contains the currency field name.
- **exchangedate_field.** Contains the exchange date field value. This tag is optional.

Example Code of the Multi Currency Control

The following code illustrates use of the multi_currency control:

```
<control id="1" class=" multi_currency " tab_order="1">

                <value_field>Revenue</value_field>

                <currency_field>Currency</currency_field>

                <exchangedate_field>Date</exchangedate_field>

</control>
```

MVG Primary Selector Control of the forms_xx.xml File

If you set the class attribute of the control tag to mvg_primary_selector, then you must define the following tags in the control tag:

- **source.** Behavior is similar to the lookup control. The source tag includes the following attributes:
  - **type.** Defines the many-to-many association ID that Siebel CRM Desktop uses for this control.
  - **linking_field.** Contains the field name of this association where the ID of the parent object is saved.
  - **flag_field.** Contains the field name that Siebel CRM Desktop uses to set the primary flag.
  - **display_format.** Defines the output mask.

Microsoft Outlook View Control of the forms_xx.xml File

If you set the class attribute of the control tag to Microsoft Outlook_view, then you must define the following attributes in the control tag:

- **view_id.** Contains the ID of the view that Siebel CRM Desktop uses for this control. The views.xml file describes the view IDs.
- **type.** Defines the ID of the type of objects to display in this control. The siebel_basic_mapping.xml file describes these objects.

If you set the class attribute of the control tag to Microsoft Outlook_view, then you must define the following tag in the control tag:

- **dasl.** Contains filters that Siebel CRM Desktop applies to this view. This filter must be in dasl format. For more information about dasl, see the relevant Microsoft documentation. Consider the following example:

```
<control id="activities_view" class="Microsoft Outlook_view"
view_id="activities:form_view" type="Action">

 <dasl><![CDATA["http://schemas.microsoft.com/mapi/string/{00020329-0000-0000-
C000-000000000046}/sbl%20Account%20ID" = '{{id|s}}']]></dasl>

</control>

<text>Opportunity:sf Account Id:AccountId</text>

<control id="0x10113" class="Microsoft Outlook_view">

 <text>Opportunity:sf Account Id:AccountId</text>

 <resource_id>opportunities:salesbook</resource_id>

</control>
```

Subform Control of the forms_xx.xml File

The following code illustrates use of the subform control:

```
<cell size="22">

    <control class="subform" id="activity_subform">
```

```
            <cell size="22">
                <stack layout="horz" spacing="20" padding="6">
                    <cell>
                        <control class="static" id="ActivityLabel">
                            <text>Activity Name:</text>
                        </control>
                    </cell>
                    <cell>
                        <control class="edit" id="ActivityName">
                            <field>Name</field>
                        </control>
                    </cell>
                </stack>
            </cell>
        </control>
    </cell>
```

Web_page Control of the forms_xx.xml File

If you set the class attribute of the control tag to web_page, then you must define the url attribute in the control tag. The following code illustrates use of the web_page control:

```
<control class="web_page" id="linkedin_search">
     <url>http://www.linkedin.com/</url>
  </control>
```

You can define a static or a dynamic URL as the value of the url attribute. If the URL is dynamic, then JavaScript supports it. For example, you can present a dynamic personal page for a business contact on the Contact form. The following is an example of this JavaScript:

```
if (!is_new)
  form.linkedin_search.navigate = "http://www.linkedin.com/pub/dir/?last=" +
form.item['Last Name'] + "&first=" + form.item['First Name'];
```

If the url attribute is not set, then Siebel CRM Desktop loads the about:blank page by default.

XML Code to Customize Toolbars

This topic describes the code of the toolbars.xml file. It includes the following topics:

- "Example Code of the toolbars.xml File" on page 273
- "Toolbars Tag of the toolbars.xml File" on page 273

For more information, see "Customizing a Toolbar" on page 151.

Example Code of the toolbars.xml File

This topic gives one example of code that you can use in the toolbars.xml file. You might use this feature differently, depending on your business model. The following code is an example of the toolbars.xml file:

```
<button id="meeting_with_contact" name="#btn_meeting_with_contact"
small_image="orcl_meeting_with_contact:16">

  <action class="scriptable" id="meeting_with_contact"/>

</button>
```

Toolbars Tag of the toolbars.xml File

The toolbars tag is the root tag of the toolbars.xml file. It describes the toolbar that Siebel CRM Desktop adds to a native Microsoft Outlook form or Microsoft Outlook window.

The caption attribute of the toolbars tag defines the caption of the toolbar.

The toolbars tag includes the following tags:

- **for.** Determines if Siebel CRM Desktop displays this tag in a Microsoft Outlook window or Microsoft Outlook form, depending on if the value of the for tag is explorer or inspector.
- **button.** A description of a button on a toolbar. For more information, see "Button Tag of the Toolbars Tag of the toolbars.xml File" on page 273.

Button Tag of the Toolbars Tag of the toolbars.xml File

The button tag includes the following attributes:

- **name.** The caption for the button.
- **small_image.** The resource ID of the icon that Siebel CRM Desktop uses as the small icon for this button.
- **large_image.** The resource ID of the icon that Siebel CRM Desktop uses as a large icon for this button. The large_image attribute is valid for Microsoft Outlook 2007 but is ignored for Microsoft Outlook 2003.

- **begin_group.** Determines if Siebel CRM Desktop displays the separator of the toolbar button for this button. It is useful if you must group toolbar buttons.

Action Tag of the Button Tag

The button tag includes the action tag, which defines the action that Siebel CRM Desktop calls if the user clicks the button. You can use a predefined action or write a custom action. You must set this action in the class attribute of the action tag.

The action tag includes the class attribute. The class attribute can include the following values:

- **create_attachment.** Opens the Select Attachment dialog box to allow the user to choose a file to attach to the current object. To define which object can contain an attachment, use the accept_type attribute. The accept_type attribute defines the visibility of the button, depending on which object type is currently chosen. The value of the accept_type attribute must be the object type for which you must make this button visible.
- **create_linking_item.** Creates a new object and associates it with the current object. You can also define the object types for which this button is visible. These object types must be described by type tags that reside in a types tag.

The action tag includes the attachment tag. To define attachment settings, you must define the following attributes of the attachment tag:

- **type.** The type of the attachment object.
- **name_field.** The name of the field of the attachment object where Siebel CRM Desktop stores the name of the file.
- **body_field.** The name of the field of the attachment object where Siebel CRM Desktop stores the body of the file.
- **linking_field.** The name of the field of the attachment object where Siebel CRM Desktop stores the reference to the parent object.

Example Code of the Action Tag of the Scriptable Action

Siebel CRM Desktop supports the scriptable action class. The action tag of the scriptable action includes the id attribute. You can use the id attribute in a script to specify the action to perform. The following example specifies a button with a scriptable action:

```
<button id="new_account" name="#btn_new_account">

  <action class="scriptable" id="new_account"/>

</button>
```

In this example, Siebel CRM Desktop passes the value for the new_account attribute to the script when it handles the click event of the button. The script includes predefined logic that the new_account attribute starts.

XML Code to Customize Dialog Boxes

This topic describes the code of the dialogs.xml file. It includes the following topics:

- "Dialog Tag of the dialogs.xml File" on page 275
- "Layout Tag of the dialogs.xml File" on page 275
- "Appearance Tag of the dialogs.xml File" on page 275

For more information, see "Customizing a Dialog Box" on page 151.

Dialog Tag of the dialogs.xml File

The dialog tag describes each dialog box. This tag is similar to the form tag of the forms_11.xml and forms_12.xml files, but it does not support the on_saved and on_saving attributes. The dialog tag in the dialogs.xml file behaves in the same way as the form tag in the forms_xx.xml file, except for the following differences:

- The dialog box description includes the layout tag and the appearance tag.
- The dialogs.xml file does not contain a validation_rules tag.
- You cannot use native Microsoft Outlook controls in the dialogs.xml file.

Layout Tag of the dialogs.xml File

The layout tag is similar to the page tag of the forms_11.xml and forms_12.xml files, but it includes different attributes. These attributes include:

- **sizable.** Determines if the user can change the size of the dialog box.
- **visible.** Sets the visibility for the dialog box during creation. If the visible attribute is set to false, then Siebel CRM Desktop creates the dialog box in the background. To make it visible, you must use JavaScript code to change the value for this attribute.
- **caption.** Defines the caption for the dialog box.
- **small_icon.** Defines the icon that displays next to the caption. The value of this attribute must be an ID of an image resource from Siebel CRM Desktop.

Appearance Tag of the dialogs.xml File

The appearance tag defines the position and size of the dialog box. It includes the following attributes:

- **height.** Defines the height of the dialog box, in pixels.
- **width.** Defines the width of the dialog box, in pixels.
- **position.** Defines the position of the dialog box in a screen. The position attribute can contain the following values:

- **parent_center.** Displayed on the center of a parent window.
- **desktop_center.** Displayed on the center of the desktop.
- **custom.** A custom position.
- **top.** The top position of the dialog box. The position attribute must include a custom value.
- **left.** The left position of the dialog box. The position attribute must include a custom value.

XML Code to Customize Views

This topic describes the code of the views.xml file. This file contains the set of str tags that define the configuration of the Microsoft Outlook view. The only important attribute of this tag defines the unique name, or ID, for the view. For more information, see "Customizing a View" on page 152.

The following code is an example of the views.xml file:

```
<res_root>
    <str key="sample_view">
     <![CDATA[<?xml version="1.0"?>
<view type="table">
  <viewname>Sample view</viewname>
  <linecolor>8421504</linecolor>
  <linestyle>3</linestyle>
  <gridlines>1</gridlines>
  <newitemrow>0</newitemrow>
  <usequickflags>0</usequickflags>
  <collapsestate/>
  <previewstyle>color:Blue</previewstyle>
  <arrangement>
     <autogroup>0</autogroup>
     <collapseclient/>
  </arrangement>
  <column>
     <name>HREF</name>
     <prop>DAV:href</prop>
     <checkbox>1</checkbox>
```

```
  </column>
  <column>
     <maxrows>4294901760</maxrows>
     <heading>Organization</heading>
     <prop>urn:schemas:contacts:sn</prop>
     <type>string</type>
     <width>987</width>
     <style>padding-left:3px;;text-align:left</style>
     <editable>1</editable>
     <userheading>Organization</userheading>
  </column>
  <orderby>
     <order>
        <heading>Organization</heading>
        <prop>urn:schemas:contacts:sn</prop>
        <type>string</type>
        <userheading>Organization</userheading>
        <sort>asc</sort>
     </order>
  </orderby>
  <multiline>
     <width>0</width>
  </multiline>
</view>]]>
  </str>
</res_root>
```

XML Code to Customize the SalesBook Control

This topic describes the code of the lookup_view_defs.xml file. It includes the following topics:

- "Example Code of the lookup_view_defs.xml File" on page 278
- "Array Tag of the lookup_view_defs.xml File" on page 278
- "Lookup View Definition Tag of the lookup_view_defs.xml File" on page 279

For more information, see "Customizing the SalesBook Control" on page 152.

Example Code of the lookup_view_defs.xml File

This topic gives one example of code that you can use in the lookup_view_defs.xml file. You might use this feature differently, depending on your business model. The following code is an example of the lookup_view_defs.xml file:

```
<res_root>

    <array key="all_lookup_types">

        <item value="Account"></item>

        <item value="Contact"></item>

        <item value="Opportunity"></item>

    </array>

    <lookup_view_def key="lookup:contacts">

        <display name="Contacts"></display>

       <filter dasl="[http://schemas.microsoft.com/mapi/proptag/0x001A001E] &gt;=
'IPM.Contact.SBL.Contact' AND [http://schemas.microsoft.com/mapi/proptag/
0x001A001E] &lt;= 'IPM.Contact.SBL.Contact'"></filter>

        <view id="contacts:salesbook"></view>

       <quick_lookup dasl_format="[http://schemas.microsoft.com/mapi/id/{00062004-
0000-0000-C000-000000000046}/8005001E] = '%s'"></quick_lookup>

        <type id="Contact"></type>

    </lookup_view_def>

</res_root>
```

Array Tag of the lookup_view_defs.xml File

The array tag defines a set of types that is available for the SalesBook control. The user cannot use the SalesBook control to choose a certain object type until you describe this type in the array tag. Also, you must specify the type ID as a value attribute of the item tag.

The following code is an example of the array tag:

```
<array key="all_lookup_types">
```

```
        <item value="Account"></item>
        <item value="Contact"></item>
        <item value="Opportunity"></item>
    </array>
```

Lookup View Definition Tag of the lookup_view_defs.xml File

The lookup_view_def tag describes the configuration for the SalesBook control. You can define as many configurations as you require. The key attribute defines the unique ID, or name, for this configuration.

The lookup_view_def tag includes the following tags:

- **display.** The name attribute of the display tag defines the name of this configuration. Siebel CRM Desktop displays it as a list value in the top, right corner of the SalesBook control.
- **filter.** The dasl attribute of the filter tag describes the dasl filter that Siebel CRM Desktop applies to all objects that the array tag describes. The user can only view the objects that match this filter in the SalesBook control.
- **view.** The id attribute of the view tag defines the view that Siebel CRM Desktop applies to the list in the SalesBook control. You must describe this view ID in the views.xml file.
- **quick_lookup.** The dasl_format attribute of the quick_lookup tag defines the dasl filter that Siebel CRM Desktop applies to the quick search feature of the SalesBook control. To simplify finding a field, this feature allows the user to enter any text to filter records. The user enters text in the top, left edit box on a SalesBook form.

  The following example code allows the user to view records where the File As field is the same as the string:

  ```
  <quick_lookup dasl_format="[http://schemas.microsoft.com/mapi/id/{00062004-
  0000-0000-C000-000000000046}/8005001E] = '%s'"></quick_lookup>
  ```

  where:

  - The *File As* field is ([http://schemas.microsoft.com/mapi/id/{00062004-0000-0000-C000-000000000046}/8005001E]).
  - The *quick search* is entered as ('%s').
- **type.** The ID attribute of the type tag defines the type of object that Siebel CRM Desktop creates if the user clicks New in a SalesBook control. If you do not define this attribute, then the user cannot create a new object.

XML Code for Meta Information

This topic describes the code of the siebel_meta_info.xml file. It includes the following topics:

- "SiebelMetaInfo Tag of the siebel_meta_info.xml File" on page 280
- "Common_settings Tag of the siebel_meta_info.xml File" on page 280
- "Object Tag of the siebel_meta_info.xml File" on page 281
- "Field Tag of the siebel_meta_info.xml File" on page 282
- "Extra_command_options Tag of the siebel_meta_info.xml File" on page 284
- "Open_with_url_tmpl Tag of the siebel_meta_info.xml File" on page 284
- "Picklist Tag of the siebel_meta_info.xml File" on page 285
- "Master_filter_expr Tag of the siebel_meta_info.xml File" on page 285

For more information, see "Customizing Meta Information" on page 152.

SiebelMetaInfo Tag of the siebel_meta_info.xml File

The SiebelMetaInfo tag is a root tag. It does not contain attributes.

Common_settings Tag of the siebel_meta_info.xml File

The common_settings tag does not contain tags. However, it can contain subtags that you can use to specify common options for the Web Service Connector. Siebel CRM Desktop supports the following subtags:

- **max_commands_per_batch.** Defines the maximum number of commands that Siebel CRM Desktop can place in a single batch. If Siebel CRM Desktop cannot interpret the value of this tag as a positive integer value, then it does not apply any restrictions on the number of commands.
- **max_ids_per_command.** Defines the maximum number of object IDs that Siebel CRM Desktop can specify in a search specification for each independent request when a user performs a query by ID. It is the maximum number of record IDs that can be related to the parent record.

For example:

```
<common_settings>
   <max_commands_per_batch>50</max_commands_per_batch>
   <max_ids_per_command>50</max_ids_per_command>
</common_settings>
```

Object Tag of the siebel_meta_info.xml File

You can use the object tag to define an object type that Siebel CRM Desktop supports. Table 35 describes the tags that Siebel CRM Desktop supports.

Table 35. Tags of the Object Tag of the siebel_meta_info.xml File

Tag	Type	Description
EnableGetIDsBatching	Binary: yes or no	Allows or disallows batching for IDs for each command. The following values are valid: ■ **False.** Disallows ID batching for a get command. ■ **True.** Allows ID batching for a get command.
IntObjName	String	Name of the integration object that Siebel CRM Desktop uses for requests.
IsAssociation	Binary: yes or no	Indicates if this type of object is an association object.
IsCascadeDelete	Binary: yes or no	Not currently used.
IsTopLevel	Binary: yes or no	Indicates if a request for this type of object must be wrapped in a request for an object of some parent type.
Label	String	Label that Siebel CRM Desktop uses for this type of object in the user interface.
LabelPlural	String	Plural label that Siebel CRM Desktop uses for this type of object in the user interface.
SiebMsgXmlCollectionElemName	String	Name of collection XML element that Siebel CRM Desktop uses in a Siebel message.
SiebMsgXmlElemName	String	Name of the XML element that Siebel CRM Desktop uses in a Siebel message.
SyncFrequency	Numeric	Identifies how often Siebel CRM Desktop synchronizes the type, measured in seconds. If you set SyncFrequency to 0, then Siebel CRM Desktop synchronizes the type during every synchronization session. If you enter a positive integer, then Siebel CRM Desktop queries the Siebel Server for the records of this type in the time interval you define, starting from the time the type was last queried.
TypeId	String	Unique ID of this type of object.

Table 35. Tags of the Object Tag of the siebel_meta_info.xml File

Tag	Type	Description
UpsertBusObjCacheSize	Numeric	Request attribute that defines the preferred cache size for each upsert operation for each object type. The Siebel API uses this information. The following values are valid: ■ **5.** Default value for all types. ■ **0.** Special value that you can use to resolve a problem that might exist with primaries.
ViewMode	String	Default ViewMode for this type of object.

Field Tag of the siebel_meta_info.xml File

You can use the field tag to define an object field. You must nest the field tag in the definition of an object type. Table 36 describes the tags that Siebel CRM Desktop supports.

Table 36. Tags of the Field Tag of the siebel_meta_info.xml File

Tag	Type	Description
DataType	String	Indicates the data type. The Web Service Connector uses the value for this attribute to perform data conversion.
HasPicklist	Binary: yes or no	Indicates if this field is a bounded list.
IOElemName	String	Name of the XML element that Siebel CRM Desktop uses in Siebel messages for values from this field.
IsCompositeId	Binary: yes or no	Not currently used.
IsFake	Binary: yes or no	If the value for this tag is yes, then Siebel CRM Desktop does not use the value from this field in any requests to the API.
IsFilterable	Binary: yes or no	Indicates if this field is available to choose a filter expression on the control panel.
IsMVGField	Binary: yes or no	Not currently used.
IsNullable	Binary: yes or no	The Synchronization Engine uses this tag to break a circular reference.
IsPartOfUserKey	Binary: yes or no	Indicates if this field is part of a user key.

Table 36. Tags of the Field Tag of the siebel_meta_info.xml File

Tag	Type	Description
IsPrimaryKey	Binary: yes or no	Indicates if Siebel CRM Desktop uses the value from this field as the primary key for the object. Only one field on an object type can be marked as the primary key.
IsReadonly	Binary: yes or no	Not currently used.
IsRefObjId	Binary: yes or no	Indicates if Siebel CRM Desktop uses this tag as a foreign key field. The Web Service Connector and the Synchronization Engine use this tag.
IsRequired	Binary: yes or no	Not currently used.
IsTimestamp	Binary: yes or no	Indicates if Siebel CRM Desktop uses the value from this field as an object timestamp. You can define only one timestamp for each type of object.
Label	String	The label that displays in the user interface.
Name	String	Unique name of the field. If the IsFake tag for this field is set to no, and if this field is not present in requests to the Siebel Server, then this name must be identical to the field name of the Integration Component from the API.
OrderNumber	Numeric	Indicates the order number of this field. The Web Service Connector uses this tag internally. If several fields are defined that hold a reference to a parent record, then their order numbers must reflect the nesting order of the parent types. This situation occurs if an association exists between two parent types where one of the parent types is nested in the other parent type.
PicklistCollectionType	String	Type of list items.
PicklistIsStatic	Binary: yes or no	Indicates if the associated list is static or dynamic.
PicklistTypeId	String	Name of the type of list object. You must use the picklist tag to define the list in the siebel_meta_info.xml file.
RefObjIsParent	Binary: yes or no	Indicates if the object type that is referenced is a parent. The Web Service Connector uses this tag connector to build the hierarchy for the object type.
RefObjTypeId	String	The name of the object type to which this field refers. This object type must be defined in the siebel_meta_info.xml file.

Extra_command_options Tag of the siebel_meta_info.xml File

You can use the extra_command_options tag in the definition of an object type to specify extra options that Siebel CRM Desktop passes to a command element on each request. You can use the option subtag with the following tags to specify each tag:

- **Name.** Name of the extra command attribute.
- **Value.** Value of the extra command attribute.

Open_with_url_tmpl Tag of the siebel_meta_info.xml File

You can use the open_with_url_tmpl tag in the definition of an object type. You use this tag to specify a template that Siebel CRM Desktop uses to open records of this object type in the Siebel Web Client. For more information, see "Defining the URL That Siebel CRM Desktop Uses to Open the Siebel Web Client" on page 113.

The open_with_url_tmpl tag includes the following format:

```
<open_with_url_tmpl>
   <![CDATA[
   "URL template"
   ]]>
</open_with_url_tmpl>
```

Siebel CRM Desktop uses macros in the CDATA section for the attributes that Siebel CRM Desktop uses in the open_with_url_tmpl template. Each attribute includes the following format:

```
:[:(attribute):]
```

A series of attributes in the command uses the following format:

```
:[:(protocol):]://:[:(hostname):]:[:(port):]:[:(own_id):]
```

where:

- *protocol* is automatically replaced with the URL protocol. The value is http or https.
- *hostname* is automatically replaced with the name of the Siebel Server.
- *port* is automatically replaced with the port number of the Siebel Server.
- *own_id* is automatically replaced with the object ID that Siebel CRM Desktop uses to open the Siebel Web Client.

For each macro in the command, Siebel CRM Desktop replaces the macro with the actual value at runtime. The definition of the template must be enclosed in a CDATA construct.

For example, you can use the following code for account objects:

```
<open_with_url_tmpl>
    :[:(protocol):]://:[:(hostname):]::[:(port):]/sales_enu/
    start.swe?SWECmd=GotoView&SWEView=Account+List+View&SWERF=1&SWEBU=1&SWEApplet0=
    Account+Entry+Applet&SWERowId0=:[:(own_id):]
    ]]>
</open_with_url_tmpl>
```

Picklist Tag of the siebel_meta_info.xml File

You can use the picklist tag to define a static list. Table 37 describes the tags that Siebel CRM Desktop supports.

Table 37. Tags of the Picklist Tag of the siebel_meta_info.xml File

Tag	Type	Description
TypeID	String	Unique name for the list.
SrcObjectTypeId	String	Name of the object type that Siebel CRM Desktop uses to retrieve items for this list. This type must be defined in the siebel_meta_info.xml file.
CollectionTypeFldName	String	Name of the field on the original object that contains the type for the list items.
ValueFldName	String	Name of the field on the original object from which Siebel CRM Desktop retrieves values for the list items.
LabelFldName	String	Name of the field on the original object from which Siebel CRM Desktop retrieves labels for the list items.
LangFldName	String	Name of the field on the original object that contains the language code.

Master_filter_expr Tag of the siebel_meta_info.xml File

You can nest the master_filter_expr tag in the definition of a list. You can use this tag to define a filter expression that Siebel CRM Desktop applies to any request for items of this list. You must enclose the value for this tag with a CDATA section and display the values for the search specification attributes of the target object type.

C Additional Code in the Customization Example

This appendix includes the more lengthy XML code that the customization example requires. It includes the following topics:

- XML Code That Defines a Set of Custom Fields on page 287
- XML Code That Defines a Many-To-Many Relationship on page 289
- XML Code That Defines a List on page 291
- XML Code That Creates Cells on page 293

For more information, see "Process of Customizing Objects in Siebel CRM Desktop" on page 166.

XML Code That Defines a Set of Custom Fields

To define a set of custom fields, you add the following code to the type tag of the siebel_basic_mapping.xml file:

```
<type id="ChannelPartner" hidden_folder="true" folder_type="10"
display_name="CHPT">

 <form message_class="IPM.Contact.SBL.Channel_Partner" display_name="Channel
Partner" icon="type_image:User:16"></form>

  <field id="Name">

   <reader>

    <mapi_std>

     <mapi_tag id="0x3A110000"></mapi_tag>

     <convertor><string/></convertor>

    </mapi_std>

   </reader>

   <writer>

    <outlook_std>

     <outlook_field id="LastName"></outlook_field>

     <convertor><string/></convertor>

    </outlook_std>
```

```
    </writer>
    <reader>
     <mapi_std>
      <mapi_tag id="0x3A060000"></mapi_tag>
      <convertor><string/></convertor>
     </mapi_std>
    </reader>
    <writer>
     <outlook_std>
      <outlook_field id="FirstName"></outlook_field>
      <convertor><string/></convertor>
     </outlook_std>
    </writer>
   </field>
   <field id="Location">
    <reader>
     <mapi_user>
      <user_field id="sbl Location" ol_field_type="1"></user_field>
      <convertor><string/></convertor>
     </mapi_user>
    </reader>
    <writer>
     <outlook_user>
      <user_field id="sbl Location" ol_field_type="1"></user_field>
      <convertor><string/></convertor>
     </outlook_user>
    </writer>
   </field>
  </type>
```

XML Code That Defines a Many-To-Many Relationship

You must define the Microsoft Outlook folder that stores these objects. However, for this example, it is not desirable to display these objects to the user. Therefore, you use a hidden folder in Microsoft Outlook that you define in the code as the following:

```
predefined_folder="1"
```

Because a form is not required to display these objects, you do not define a form. You add the following code to the siebel_basic_mapping.xml file:

```
<type id="Opportunity.Channel_Partner.Association" hidden_folder="true"
folder_type="10" display_name="OCHP">

    <form message_class="IPM.Contact.SBL.OpportunityChannel_PartnerAssociation"
display_name="OpportunityChannel_PartnerAssociation"
icon="type_image:Generic:16"></form>

    <field id="OpportunityId">

        <reader>

            <mapi_user>

                <user_field id="sbl OpportunityId" ol_field_type="1"></user_field>

                <convertor><binary_hexstring/></convertor>

            </mapi_user>

        </reader>

        <writer>

            <outlook_user>

                <user_field id="sbl OpportunityId" ol_field_type="1"></user_field>

                <convertor><binary_hexstring/></convertor>

            </outlook_user>

        </writer>

    </field>

    <field id="ChannelPartnerId" ver="2">

        <reader>

            <mapi_user>

              <user_field id="sbl ChannelPartnerId" ol_field_type="1"></user_field>

                <convertor><binary_hexstring/></convertor>
```

```
            </mapi_user>
        </reader>
        <writer>
            <multiwriter>
                <outlook_user>
                    <user_field id="sbl ChannelPartnerId" ol_field_type="1"></
user_field>
                    <convertor><binary_hexstring/></convertor>
                </outlook_user>
                <link_fields>
                    <field from="Name" to="PartnerName"></field>
                    <field from="Location" to="PartnerLocation"></field>
                </link_fields>
            </multiwriter>
        </writer>
    </field>
    <field id="PartnerName">
        <reader>
            <mapi_user>
                <user_field id="sbl PartnerName" ol_field_type="1"></user_field>
                <convertor><string/></convertor>
            </mapi_user>
        </reader>
        <writer>
            <outlook_user>
                <user_field id="sbl PartnerName" ol_field_type="1"></user_field>
                <convertor><string/></convertor>
            </outlook_user>
        </writer>
    </field>
```

```
        <field id="PartnerLocation">
            <reader>
                <mapi_user>
                   <user_field id="sbl PartnerLocation" ol_field_type="1"></user_field>
                    <convertor><string/></convertor>
                </mapi_user>
            </reader>
            <writer>
                <outlook_user>
                   <user_field id="sbl PartnerLocation" ol_field_type="1"></user_field>
                    <convertor><string/></convertor>
                </outlook_user>
            </writer>
        </field>
    </type>
```

XML Code That Defines a List

To define a list that drops down, you add the following code to the siebel_basic_mapping.xml file on the activity object:

```
    <type id="ActionTypePicklist" predefined_folder="1">
            <form message_class="IPM.Contact.SBL.ActionTypePicklist"></form>
            <field id="Label">
               <reader class="mapi_user">
                  <user_field id="sbl picklistLabel" ol_field_type="1"></user_field>
                  <convertor class="string"></convertor>
               </reader>
               <writer class="Microsoft Outlook_user">
                  <user_field id="sbl picklistLabel" ol_field_type="1"></user_field>
                  <convertor class="string"></convertor>
               </writer>
```

```
</field>
<field id="Value">
   <reader class="mapi_user">
      <user_field id="sbl picklistValue" ol_field_type="1"></user_field>
      <convertor class="string"></convertor>
   </reader>
   <writer class="Microsoft Outlook_user">
      <user_field id="sbl picklistValue" ol_field_type="1"></user_field>
      <convertor class="string"></convertor>
   </writer>
</field>
<field id="SortOrder">
   <reader class="mapi_user">
      <user_field id="sbl SortOrder" ol_field_type="3"></user_field>
      <convertor class="integer"></convertor>
   </reader>
   <writer class="Microsoft Outlook_user">
      <user_field id="sbl SortOrder" ol_field_type="3"></user_field>
      <convertor class="integer"></convertor>
   </writer>
</field>
<field id="IsDefault">
   <reader class="mapi_user">
      <user_field id="sbl IsDefault" ol_field_type="6"></user_field>
      <convertor class="bool"></convertor>
   </reader>
   <writer class="Microsoft Outlook_user">
      <user_field id="sbl IsDefault" ol_field_type="6"></user_field>
      <convertor class="bool"></convertor>
   </writer>
```

```
        </field>
      </type>
```

XML Code That Creates Cells

To define part of the layout of the form, you add the following code to the forms_12.xml file:

```
<stack layout="horz" spacing="3">
<!-- left side captions -->
  <cell size="105">
     <stack spacing="5" layout="vert" padding="4">
          <cell size="22">
             <control id="0x20002" class="static" tab_order="1">
                <text>Description:</text>
             </control>
          </cell>
          <cell size="22">
             <control id="0x20004" class="static" tab_order="3">
                <text>Type:</text>
             </control>
          </cell>
          <cell size="22">
             <control id="0x20008" class="static" tab_order="5">
                <text>Priority:</text>
             </control>
          </cell>
          <cell size="22">
             <control id="0x20010" class="static" tab_order="7">
                <text>Owner:</text>
             </control>
          </cell>
          <cell size="22">
```

```
            <control id="0x20024" class="static" tab_order="9">
               <text>Account:</text>
            </control>
         </cell>
         (……..)
      </stack>
   </cell>
<!-- left side fields -->
  <cell>
     <stack layout="vert" spacing="5">
        <cell size="22">
           <control class="edit" id="0x20003" tab_order="2">
              <field value="string">Description</field>
           </control>
        </cell>
        <cell size="22">
           <control class="combobox" id="Type" tab_order="4">
              <source type="ActionTypePicklist" field="Value"
format=":[:(Label):]"></source>
              <field>Type</field>
           </control>
        </cell>
        <cell size="22">
           <control class="combobox" id="0x20009" tab_order="6">
              <source type="ActionPriorityPicklist" field="Value"
format=":[:(Label):]"></source>
              <field>Priority</field>
           </control>
        </cell>
        <cell size="22">
           <control class="lookup" id="Owner" tab_order="8">
```

```
            <source type="Employee" format=":[:(Login Name):]"
resource_id="lookup:employees"></source>

            <field>Primary Owner Id</field>

         </control>

      </cell>

      <cell size="22">

         <control id="0x20025" tab_order="10" class="lookup">

            <source type="Account" format=":[:(Name):]"
resource_id="lookup:accounts"></source>

            <field>Account Id</field>

         </control>

      </cell>

      (……..)

   </stack>

 </cell>

</stack>
```

Glossary

access control
The set of Siebel CRM mechanisms that control the records to which the user possesses access and which operations the user can perform on the records.

account
A financial entity that represents the relationships between a company and the companies and people with whom the company does business.

account team
Users who possess access to the account record. A user who is assigned to the account is a member of the account team.

ActiveX
A loosely defined set of technologies developed by Microsoft for sharing information among different applications.

ActiveX control
A specific way to implement ActiveX technology. It denotes reusable software components that use the component object model (COM) from Microsoft. ActiveX controls provide functionality that is encapsulated and reusable to programs. They are typically, but not always, visual in nature.

activity
Task that a user must track. Examples include a to-do, email sent to a contact, or an appointment with a contact.

activity (Siebel CRM)
An object in the Action business component of the Siebel data model that organizes, tracks, and resolves a variety of work, from finding and pursuing an opportunity to closing a service request. An Activity also captures an event, such as scheduling a meeting or appointment that occurs at a specific time and displays in the calendar.

activity template (Siebel CRM)
An activity that is defined in an activity template. While the activity for a template is stored in the same object as a transactional activity, this document uses a different term. A template activity essentially behaves like reference data and contains a subset of the attributes for an activity, plus some more attributes that are only relevant to being part of a template.

appointment (Microsoft Outlook)
A record in the Microsoft Outlook calendar or Siebel Web application calendar that reserves time to perform something, such as an appointment to schedule a meeting with a customer or to reserve time to complete work in a given time frame.

attendee (Microsoft Outlook)
A person included in the appointment, such as an organizer or a participant.

authentication
Process of verifying the identity of a user.

business component
A logical representation of one or more Siebel tables that usually contains information for a particular functional area, such as opportunity, account, contact, or activity. A business component can be included in one or more business objects.

business object
A logical representation of CRM entities, such as accounts, opportunities, activities, and contacts, and the logical groupings and relationships among these entities. A business object uses links to group business components into logical units. The links provide the one-to-many relationships that govern how the business components interrelate in this business object. For example, the opportunity business object groups the opportunity, contact, and activities business components.

business object (activity)
The object that is the parent of or related to the activity. For example, a service request, opportunity, marketing campaign, order orchestration process, and so forth.

business object (interaction)
The object that is the focus of the communication between the customer and the organization. For example, a service request, opportunity, contract, and so forth.

child business component
A business component that represents the many in the one-to-many relationship between two business components in a parent-child relationship.

child record
An instance of the child business component.

client computer
The computer that the Siebel CRM Desktop user uses. This is the computer on which you install the Siebel CRM Desktop add-in.

consumer
In Siebel CRM, a consumer is a person with a party of usage type Customer. In Microsoft Outlook, a consumer is visible from the Contacts folder and is flagged with the Customer check box.

contact
A person with whom a user might be required to phone or email to pursue a selling relationship. Various business objects can refer to a contact, and this does not require a relationship between the customer and contact. In Siebel CRM, a contact attribute in the context of a business object is a party that might or might not have a relationship defined. In Microsoft Outlook, a contact attribute in the context of a business object is the same as the Contact folder. Therefore, a contact can be a consumer and can also be an employee of an organization.

contact points
Methods of contacting a contact other than through a postal address, such as such as email, telephone, and fax.

Global ID

An attribute on an appointment record in Microsoft Outlook that the user can use to correlate shared appointments between meeting attendees. Meeting attendees in Microsoft Outlook include their own copy of the appointment, but all copies include the same value for the global ID.

CRM (Customer Relationship Management)

A software application that helps a business track customer interactions.

CRM contact

A contact who uses a user interface where the interface uses a CRM style and is shared with CRM.

Siebel CRM Desktop add-in

The technology for Siebel CRM Desktop that resides on the client computer that is provided in the form of a Microsoft Outlook add-in. The Microsoft Outlook add-in performs important work, including storing and displaying Siebel CRM data in native Microsoft Outlook, and synchronizing PIM and nonPIM data with the Siebel Server.

See also Microsoft Outlook add-in.

current view

The Microsoft Outlook view that displays content from the Microsoft Outlook folder that is currently chosen.

custom view

A view that a user creates to control the amount of detail that displays in a particular folder. The user can create a filter or change the order of the columns and how the columns are arranged in the new custom view.

customer

A party with whom a user maintains a selling relationship. This party can be an organization or a person. Various business objects can refer to a customer. In Siebel CRM, a customer attribute in the context of a business object can be a person or an organization that includes the party usage type of Customer. In Microsoft Outlook, a customer attribute in the context of a business object can be an organization or a contact that is flagged as a consumer.

customer team

A group of several employees from the deploying organization or partners who actively work with a customer, including nonsales personnel, such as product marketing, partners, or customer service. The customer team provides the ability to control the visibility of the customer information by associating a person with a business object.

customization

The process of changing the definition of the Siebel CRM Desktop application.

customization package

A logical collection of metadata files that are associated with a particular responsibility. A customization package is deployed to the client computer.

cyclical synchronization

A potential synchronization problem when two or more synchronizations form a circular loop. A cyclical synchronization occurs when a single transaction repeatedly loops between servers.

data synchronization

The process of checking for differences between two or more different sets of data, then updating the data sets so that the data in each set is consistent.

DHTML

Dynamic HTML, a combination of technologies that you use to create dynamic Web sites. It can be a combination of HTML 4.0, Style Sheets, and JavaScript.

Dynamic HTML (DHTML)

See DHTML.

encryption

The method of encoding data for security purposes.

form

A generic concept that Microsoft Outlook uses to present information about a single record and data related to that record in a form layout. Each control in the form is a separate attribute or collection of related data. A form can also support different tabs so that details of a child record can be displayed as separate lists.

hash value

A fixed-size string that is obtained as a result of cryptographic transformation from a cryptographic hash function.

homepage

A user interface component in Microsoft Outlook that displays a collection of information from Microsoft Outlook and CRM applications, and potentially external Web content that is embedded.

household

Provides a way to group consumers.

inbound Web service

A Web service that the Siebel Server makes available.

integration object instance

Data that is organized in the format or structure of the integration object. It is also referred to as a Siebel message object.

interaction (Siebel CRM)

The tracking of customer communications with an organization in the context of the channels through which that communication occurs and the business objects to which they refer. An interaction can take the form of a phone call, email, chat request, Web collaboration, or communication through another channel. An interaction in Siebel CRM Desktop provides an historical view of the communication that occurred. For example, Sales uses interactions to capture communications with a customer during the

sales cycle. To pursue an opportunity, a user can log a call as an interaction that the representative made.

installation package

An installation executable that includes the application binaries and any necessary instructions for completing the customization package installation in Microsoft Outlook. It also includes details that are required to connect the application server for the initial synchronization.

lead

An unqualified sales opportunity that often represents the first contact in the opportunity management process. After a lead is qualified it can be converted to an opportunity.

list view

A generic concept in a PIM application that presents information in a list. Each row in the list is a separate record and each column in the list is a separate field in the record.

lookup control

A control that is available in Microsoft Outlook that allows the user to view records in a list, then choose one or more records to associate with the current item. To identify a subset of data from which to choose the data, a lookup control typically includes the capability to specify a search condition.

meeting

An appointment in Microsoft Outlook that includes at least one participant.

metadata files

XML files that hold information on how the user experience must be shaped. The Siebel CRM Desktop add-in uses metadata files to perform field mapping with the user interface, look ups in the user interface, application object mapping, and general representation of the user interface.

offline

A mode in which the user uses Siebel CRM Desktop but does not possess access to the Siebel Server. When in offline mode, Siebel CRM Desktop uses data in the local data to perform operations. Synchronization is delayed until the user is online.

online

A mode in which the user uses Siebel CRM Desktop while connected to the Siebel Server and synchronizes data with the Siebel Server at regular intervals, or when the user performs an update. Similar to offline mode, when in online mode Siebel CRM Desktop uses data in the local data store to perform operations.

opportunity

A qualified sales engagement that represents potential revenue where a sales representative is willing to officially commit to the pipeline and to include revenue in the sales forecast. The sales representative monitors the opportunity life cycle. This representative might be compensated depending on the results of cumulative sales and potentially how well the representative maintains details about the opportunity.

organization team

Includes the sales groups who possess ownership of the associated prospect, customer, or products with the opportunity, or who are involved for a certain size of deal or with a specific sales stage, and partner organizations that can help close the deal.

organizer
In Microsoft Outlook, the person who created the appointment.

Microsoft Outlook data
Data that is created in the native Microsoft Outlook application.

Microsoft Outlook folder
A folder in Microsoft Outlook that contains a collection of data, such as email messages in the Inbox folder, or sent email messages in the Sent Items folder. In the context of this book, a Microsoft Outlook folder might also contain Siebel CRM data.

Microsoft Outlook object
An entity that is native to Microsoft Outlook. Examples of Microsoft Outlook objects include an email, appointment, contact, and so forth.

Microsoft Outlook add-in
A program that performs important work, including storing and displaying Siebel CRM data in native Microsoft Outlook and synchronizing PIM and nonPIM data with the Siebel Server.

See also PIM; Siebel Server.

Microsoft Outlook portlet
A portlet that uses data in a Microsoft Outlook folder that includes a custom view filter. The Microsoft Outlook portlet includes ActiveX characteristics.

Microsoft Outlook standard view
A default Microsoft Outlook view that exists without Siebel CRM Desktop. A Microsoft Outlook view provides different ways of viewing the same information in a folder by placing the information in different arrangements and formats.

parent business component
A business component that provides the one in a one-to-many relationship between two business components in a parent-child relationship.

parent record
An instance of the parent business component.

parent-child relationship
The relationship between the parent business component and the child business components that are related to the parent.

participant
In native Microsoft Outlook, the person who is invited to the meeting.

participant of interaction
The people who participate in an interaction. The participant can include an internal representative of the organization, such as a resource, agent, sales representative, and so forth. The participant can also include an external representative, such as a customer, contact of a customer, account, or a site. In a help desk or in an employee self-service application, a participant can be an employee.

personalization
The process where the user tailors the user interface and behavior of Microsoft Outlook.

PIM
Personal Information Manager. An application that typically helps a user to manage a list of contacts, calendar entries, email, and so forth. Microsoft Outlook, Google email, and Thunderbird are examples of PIMs.

personal information manager (PIM)
See PIM.

PIM data
Personal information that refers to data that is stored in native Microsoft Outlook that relates to contacts, appointments, and so forth.

portlet
A user interface component that is managed and displayed in the home page. The home page is composed of multiple portlets.

position
An entity in the Siebel data model. The position of the user determines which records are visible to the user and what operations are permissible on the records in a given Microsoft Outlook view.

property set
A logical memory structure that Siebel CRM Desktop uses to pass data between business services. Siebel EAI data is represented in the property set.

recipient
The person who receives an email.

record
A specific instance of the business component, also known as a CRM record, or an object in native Microsoft Outlook, also known as a Microsoft Outlook record.

responsibility
An entity in the Siebel data model that determines which views the user can access in Microsoft Outlook. For example, the responsibility of the sales representative allows the user to access the My Opportunities view, whereas the responsibility of the Siebel application developer allows the user to access administration views. A Siebel application developer or system administrator defines the responsibilities.

sales team
The users who possess access to an opportunity record. A user who creates the opportunity record is automatically part of the sales team. Other users can also be assigned to the sales team so that they can collaborate on the opportunity.

side pane
A user interface component that is available in native Microsoft Outlook that is analogous to a task pane or action pane in the Siebel Web Client. This region of the user interface is typically available on the right

side of the user interface. It displays a collection of data and actions on which the user can interact that are appropriate for the context in which the user accesses data.

Siebel application

An application that is part of Siebel Business Applications, such as Call Center.

Siebel CRM data

Business data that is created in the Siebel CRM Desktop add-in, data that is created in the client of a Siebel application, such as Siebel Call Center, or data that resides in the Siebel database on the Siebel Server. Examples include an opportunity, account, or activity.

Siebel CRM Desktop

A solution provided by Oracle that includes modifications to the standard Microsoft Outlook capabilities that allows the user to work with CRM records and business processes from the Siebel CRM Desktop user interface.

Siebel Server

The server that runs the Siebel Server software. The Siebel Server processes business logic and data access for Microsoft Outlook.

Siebel Web services framework

Provides access to an existing Siebel business service or workflow process as a Web service to be consumed by an external application.

SOAP

Simple Object Access Protocol, a protocol that allows a user or program to interact with Web services by exchanging XML messages that conform to SOAP.

Simple Object Access Protocol (SOAP)

See SOAP.

standard Microsoft Outlook

The native Microsoft Outlook application without the Siebel CRM Desktop add-in.

synchronization

A process that exchanges transactions between Oracle's Siebel CRM Desktop for Microsoft Outlook and Microsoft Outlook. This synchronization makes sure that CRM data is the same on the Siebel Server and in Microsoft Outlook.

synchronization filter

Criteria that are considered during data synchronization so that certain records are included and other records are excluded from processing during synchronization.

Task (Microsoft Outlook)

A part of a set of actions that accomplish a job, solve a problem or completes an assignment. In the native Microsoft Outlook application, a task is a collection of simple business objects on the user level. A task can be used as a reminder and also as a tracking tool for an effort that is scheduled compared to an actual effort.

task (Siebel CRM)

A logical unit of work that is performed by a user to finish a business operation. From the perspective of the Microsoft Outlook user, a task is the view representation of a logical unit of work that the user must perform. The task is presented in the Siebel application as a link that can be clicked in the task pane. The view, or series of views, where the user performs this unit of work is then displayed. It is part of the Task UI solution.

Web services

Self-contained, modular applications that can be described, published, located, and called over a network. Web services perform encapsulated business functions, ranging from a simple request-reply to full business process interactions. Web services combine development that uses components and Internet standards and protocols, which include HTTP, XML code, and SOAP.

See also SOAP.

Index

A

activities
- about 35
- how it is handled when it is recurrent 55
- how origin affects handling 44

administration
- customization package 82
- metadata 105
- metadata files 78
- server variables 84

appointments
- as an activity 35
- field mapping 244
- handling of 49
- how handled when CRM Desktop is removed 125
- invitee list handling for 53
- repeated 52

attachments
- configuration for in XML 274
- configuration of attachment settings in XML 274
- custom button for 177
- field mapping 237
- handling of 44
- usage with a customization package 127

B

back up
- manual export 62, 125

C

calendar items
- activities 35
- creation of 39
- field mapping 229
- how handled when saved or changed 48

Client
- file storage on 86
- installation file 85
- role in architecture 19

components
- architecture 25
- CRM Desktop 20
- customization 30
- logical 21
- synchronization 25

conflict handling
- during synchronization 71

connector_configuration.xml
- about 149
- example code 251
- usage with duplicate record prevention 73

CRM Desktop
- about 22
- architecture diagram, basic 20, 67
- architecture diagram, synchronization 25
- role in architecture 20, 67

CRM Desktop add
- removing for multiple users 126

CRM Desktop add-in
- installing 89
- installing for a single user 91
- item handling when removed 57
- network and infrastructure requirements 89
- removing for a single user 125

customization package
- about 26, 32, 144
- administering 82, 127
- affect of changing 64
- creating and publishing 83
- error handling 133
- items included during initial downloaded 60
- modifying 166
- MVG configuration 151
- registering and obtaining 94
- relation with responsibilities 33
- relations with other customization objects 31
- role in the architecture 22
- unpublishing 127
- XML code 245
- XML files included in 145

D

dialogs.xml
- about 151
- code description 275

E

Email
- data handling of email
  - how an email is handled 56

email
address processing 53
as an activity 35
field mapping 236
error handling
during synchronization 133
tracing and logging 132

F

field mapping
appointments 244
attachments 237
recurrent activity 238
Siebel CRM activity and Outlook calendar 229
Siebel CRM Activity and Outlook email 236
First Run Assistant
about 60
forms.xml
about 150, 247
code description 258
correct usage of 178
usage when mapping a field 148
usage with a dialog box 151

I

installation
CRM Desktop add-in 91
installation file
usage with client 85

L

lookup_view_defs.xml
about 152
code description 277, 279

M

metadata files
administering 78
connector_configuration.xml 73, 145, 149, 251
customization of 143
definition of 33
dialogs.xml 145, 151, 275
downloading of 64
forms_xx.xml 145, 148, 150, 151, 178, 247, 258, 267
list of metadata files 144
lookup_view_defs.xml 146, 152
lookup_view_defs.xml 277, 279
overview of administering 127
relations with customization package 33
role in customization package 26
siebel_basic_mapping.xml 146, 148, 166, 171, 245, 287, 289, 291
siebel_meta_info.xml 147, 152
toolbars.xml 147, 151, 184, 273
uploading 78
views.xml 147, 152, 276
MVG
customization of 151

P

parameters
options for setting client installation 101

R

repeated appointments 52
responsibilities
relations with other customization objects 31

S

Siebel Server
administering 84
siebel_basic_mapping.xml
about 148
code description 245
code example 287, 289, 291
using to create a user interface 171
using to define a custom object 166
siebel_meta_info.xml
about 152
synchronization
appointment handling 49
attachment handling 44
conflict handling 71
customizing 149
cyclical synchronization prevention 71
defining for a custom object 169
duplicate record prevention 74
engine in logical architecture 22
error handling during 133
of metadata during initial download 60
of metadata during subsequent downloads 64
XML code for 251

T

toolbars.xml
about 151
code description 273
using to define a toolbar 184

U

upgrading
upgrading CRM Desktop client 126

V

views.xml
about 152
code description 276