



Siebel CRM Desktop for IBM Notes Administration Guide

Version 3.4, Rev. A
January 2018

ORACLE®

Copyright © 2005, 2018 Oracle and/or its affiliates. All rights reserved.

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

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

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

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

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

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

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

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

Documentation Accessibility

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

Access to Oracle Support

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

Contents

Chapter 1: What's New in This Release

Chapter 2: Overview of Siebel CRM Desktop for IBM Notes

Benefits of Using Siebel CRM Desktop 15

Scenarios for Using Siebel CRM Desktop 15

 Scenario for Working with an Opportunity 16

 Scenario for Managing Contact Information 16

 Scenario for Managing Account Information 17

 Scenario for Creating a Relationship Between an Email and an Opportunity 17

 Scenario for Managing an Opportunity 18

Overview for Using This Book 18

Chapter 3: How Siebel CRM Desktop Works

Overview of How Siebel CRM Desktop Works 21

 Extensions to the IBM Notes User Interface 21

 Infrastructure That Siebel CRM Desktop Uses 22

 Architecture That Siebel CRM Desktop Uses 23

How Siebel CRM Desktop Uses the Siebel Enterprise 26

 Siebel Enterprise Components That Siebel CRM Desktop Uses 27

 About the Web Service API 28

 About the PIM Client Sync Service Business Service 29

 About the EAI Siebel Adapter Business Service 29

 About Integration Objects 30

 User Details Business Component 31

 About Authentication and Session Management 31

Metadata That Siebel CRM Desktop Uses 32

 Relationships Between Users, Responsibilities, Customization Packages, and Metadata Files 32

 About the Customization Package 33

 About Metadata Files 34

 About Metadata Administration 35

Chapter 4: How Siebel CRM Desktop Handles Siebel CRM Data

| | |
|--|----|
| How Siebel CRM Desktop Handles Activities | 37 |
| Overview of How Siebel CRM Desktop Handles Activities | 37 |
| How Activities Are Created or Modified | 39 |
| How Siebel CRM Desktop Processes Activities | 39 |
| How Siebel CRM Desktop Resolves Participants and Email Recipients of Activities | 41 |
| How Siebel CRM Desktop Displays Activities in IBM Notes | 43 |
| How Siebel CRM Desktop Sets the Primary Employee of Activities | 43 |
| How Siebel CRM Desktop Handles Attachments | 45 |
| How Siebel CRM Desktop Handles Shared Activities | 45 |
| How the Origin of an Activity Affects Handling | 46 |
| How Siebel CRM Desktop Handles a IBM Notes Meeting That Includes Multiple Attendees | 47 |
| How Siebel CRM Desktop Handles a Shared IBM Notes Calendar Entry That Is Declined | 48 |
| How Siebel CRM Desktop Handles IBM Notes Calendar Data | 48 |
| How Siebel CRM Desktop Handles IBM Notes Calendar Entries That Users Save, Change, or Delete | 49 |
| How Siebel CRM Desktop Handles Siebel CRM Activities That Users Save, Modify, or Delete | 49 |
| How Siebel CRM Desktop Handles a Calendar Entry | 50 |
| How Siebel CRM Desktop Handles a Repeating Calendar Entry | 58 |
| How Siebel CRM Desktop Handles IBM Notes To Do items | 59 |
| How Siebel CRM Desktop Handles IBM Notes Email Messages | 59 |
| How CRM Desktop Displays Data That Is Not Directly Visible | 60 |
| How a User Can Link Siebel CRM Records to IBM Notes Records | 61 |
| How Siebel CRM Desktop Handles Items If the User Removes the CRM Desktop Add-In | 62 |

Chapter 5: How Siebel CRM Desktop Synchronizes Data

| | |
|---|----|
| How Siebel CRM Desktop Synchronizes Data Between the Client and the Siebel Server | 63 |
| How Siebel CRM Desktop Synchronizes Data During the Initial Synchronization | 63 |
| How Siebel CRM Desktop Synchronizes Data During an Incremental Synchronization | 64 |
| How Siebel CRM Desktop Synchronizes Siebel CRM Data | 67 |
| How Siebel CRM Desktop Manages Synchronization Duration | 68 |
| Situations Where Siebel CRM Desktop Reinstalls the Data Structure | 68 |
| Factors That Determine the Data That Siebel CRM Desktop Synchronizes | 70 |

| | |
|--|----|
| How Siebel CRM Desktop Handles Synchronization Duplicates and Errors | 73 |
| How Siebel CRM Desktop Avoids Duplicate Data | 73 |
| How Siebel CRM Desktop Handles Synchronization Errors | 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 | 78 |
| Administering Server Variables | 80 |
| Overview of Installing the CRM Desktop Add-In | 81 |
| Changes That Siebel CRM Desktop Makes During Installation | 82 |
| Process of Installing the CRM Desktop Add-In | 84 |
| Preparing Your Environment for Installation | 85 |
| Installing the CRM Desktop Add-In | 87 |
| Options for Installing the CRM Desktop Add-In | 88 |
| Customizing the First Run Assistant | 89 |
| Installing Siebel CRM Desktop in the Background | 97 |
| Using the Windows Command Line to Set Optional Parameters | 98 |

Chapter 7: Administering Siebel CRM Desktop

| | |
|--|-----|
| Controlling the Behavior of Siebel CRM Desktop | 103 |
| Using the Windows Registry to Control Siebel CRM Desktop | 103 |
| Using the Metadata to Control Siebel CRM Desktop | 105 |
| Controlling How Siebel CRM Desktop Handles CRM Data | 108 |
| Controlling How Siebel CRM Desktop Assigns Calendar Entry Owners | 108 |
| Controlling How Siebel CRM Desktop Handles Email Attachments | 109 |
| Controlling the Maximum Size of an Attachment | 110 |
| Removing Siebel CRM Desktop | 113 |
| Removing the CRM Desktop Add-In for a Single User | 113 |
| Removing the CRM Desktop Add-In for Multiple Users | 114 |
| Administering Logging | 115 |
| Log Files You Can Use with Siebel CRM Desktop | 116 |
| Assigning Logging Profiles for Siebel CRM Desktop | 117 |
| Creating Custom Logging Profile | 119 |
| Creating Installation Log Files for Siebel CRM Desktop | 120 |
| Administering Logging on the Siebel Server | 121 |
| Using Script to Modify Logging Levels | 121 |

| | |
|---|-----|
| Troubleshooting Problems That Occur with Siebel CRM Desktop | 122 |
| Troubleshooting Problems That Occur When Siebel CRM Desktop Connects to the Siebel Server | 123 |
| Troubleshooting Problems That Occur During Synchronization | 124 |

Chapter 8: Controlling Synchronization

| | |
|---|-----|
| Controlling Synchronization Filters | 127 |
| Controlling the Object Types That Siebel CRM Desktop Displays in the Filter Records Tab | 127 |
| Controlling the Synchronization Exceptions Button In the Filter Records Tab | 128 |
| Controlling the Date Range in the Filter Records Tab | 129 |
| Controlling the Fields That Display in a Filter | 130 |
| Controlling Synchronization Time, Day, and Size | 131 |
| Overview of Controlling Synchronization Frequency | 131 |
| Controlling the Synchronization Intervals That Display in the Synchronization Tab | 132 |
| Controlling the Time and Day When Synchronizations Occur | 133 |
| Controlling the Size and Type of Synchronized Records | 135 |
| Synchronizing All Changes or Only Local Changes | 136 |
| Controlling the Number of Records That Synchronize | 137 |
| Configuring Siebel CRM Desktop to Disregard Erroneous Data That Users Modify | 138 |
| Controlling the Number and Size of Batch Requests | 139 |
| Controlling Other Configurations That Affect Synchronization | 140 |
| Configuring How CRM Desktop Gets Updates That Occur During Synchronization | 140 |
| Configuring CRM Desktop to Synchronize Private Activities | 141 |
| Controlling the View Mode During Synchronization According to Object Type | 143 |
| Controlling How Siebel CRM Desktop Deletes Records During Synchronization | 145 |
| Resolving Synchronization Conflicts | 148 |
| Overview of Synchronization Conflicts | 148 |
| Configuring Siebel CRM Desktop to Resolve Synchronization Conflicts | 150 |
| Examples of Auto Resolver Rules | 151 |

Chapter 9: Customizing Siebel CRM Desktop

| | |
|--|-----|
| Overview of Customizing Siebel CRM Desktop | 153 |
| Customizing Field Mapping | 154 |
| Customizing Synchronization | 154 |
| Customizing Forms | 155 |
| Customizing Dialog Boxes | 155 |
| Customizing Views | 156 |
| Hiding Custom Views | 156 |
| Customizing the SalesBook Control | 157 |
| Customizing Meta Information | 157 |

| | |
|--|-----|
| Customizations That Oracle Does Not Support | 158 |
| Preparing the Development Environment | 161 |
| Using Siebel Tools | 163 |
| Customizing Field Behavior | 164 |
| Displaying Siebel CRM Fields | 164 |
| Hiding Siebel CRM Fields | 167 |
| Making Fields Read-Only | 168 |
| Adding Default Values to Fields | 169 |
| Adding Postdefault Values to Fields | 172 |
| Updating One Field If the User Modifies Values In Another Field | 173 |
| Creating Calculated Fields | 175 |
| Customizing UI Behavior | 179 |
| Customizing the Product Name | 179 |
| Customizing the Email Address of the Support Team | 179 |
| Controlling How Siebel CRM Desktop Pins Objects | 181 |
| Controlling How Siebel CRM Desktop Handles Data That Is Not Directly Visible | 181 |
| Preventing Users from Deleting Records | 184 |
| Preventing Users from Deleting Records According to Conditions | 185 |
| Configuring CRM Desktop to Automatically Add Deleted Items to the Exclusion List | 185 |
| Making Forms Read-Only | 186 |
| Localizing Strings | 188 |
| Validating the Data That Users Enter | 190 |
| Preparing to Use Validation | 191 |
| Making Sure Users Enter Information in a Field | 191 |
| Making Sure Users Enter Unique Values | 192 |
| Making Sure Users Do Not Exceed the Maximum Number of Characters | 193 |
| Creating Custom Validations | 194 |
| Process of Adding Custom Objects | 196 |
| Creating the Custom Object | 196 |
| Defining Synchronization for Custom Objects | 202 |
| Adding Custom Views in IBM Notes | 203 |
| Defining the User Interface | 204 |
| Adding Custom Logic | 205 |
| Defining the Toolbar | 206 |
| Removing Customizations | 207 |
| Troubleshooting Problems That Occur When You Customize Siebel CRM Desktop | 208 |
| Chapter 10: Customizing Picklists | |
| Overview of Customizing Picklists | 209 |

| | |
|--|-----|
| Modifying the Values That Predefined Static Picklists Display | 212 |
| Modifying the Values That Predefined Lists of Values Display | 215 |
| Process of Creating Predefined Picklists | 215 |
| Identifying Predefined Picklist Objects in Siebel CRM | 215 |
| Creating an Integration Object for the Contact Method Picklist | 217 |
| Extending an Integration Object for the Contact Method Picklist | 218 |
| Adding Fields to the Customization Package | 220 |
| Customizing the Physical Layout for the Picklist | 227 |
| Publishing and Testing Picklists | 228 |
| Process of Creating Custom Static Picklists | 228 |
| Modifying Siebel CRM Objects to Support Static Picklists | 229 |
| Adding Fields to the Metadata to Support Static Picklists | 230 |
| Adding Fields to the Basic Mapping to Support Static Picklists | 231 |
| Modifying the Basic Mapping to Store Values for Static Picklists | 232 |
| Modifying the Form to Support Static Picklists | 233 |
| Uploading and Testing Your Static Picklist | 233 |
| Creating Static Picklists That Use Long Values | 234 |
| Process of Creating Dynamic Picklists | 235 |
| Modifying Siebel CRM Objects to Support Dynamic Picklists | 235 |
| Modifying the Metadata, Basic Mapping, and Forms to Support Dynamic Picklists | 237 |
| Process of Creating Dynamic Picklists That Use Custom Objects | 239 |
| Modifying the Business Component | 239 |
| Creating an Integration Object | 240 |
| Modifying Siebel CRM Desktop to Support the New Integration Object | 243 |
| Modifying the Remaining Siebel CRM Desktop Objects | 245 |
| Process of Creating Dynamic Picklists That Use a SalesBook Control | 247 |
| Modifying Siebel CRM Objects to Support a Dynamic Picklist That Uses a SalesBook Control | 248 |
| Modifying the Metadata | 249 |
| Modifying the Basic Mapping and Connector Configuration | 250 |
| Modifying the Business Logic and Testing Your Work | 252 |
| Process of Creating Hierarchical Picklists | 255 |
| Modifying Siebel CRM Objects to Support Hierarchical Picklists | 256 |
| Modifying the Metadata to Support Hierarchical Picklists | 258 |
| Modifying the Basic Mapping and Forms to Support Hierarchical Picklists | 259 |
| Linking Fields and Testing Your Hierarchical Picklist | 261 |
| Configuring Lists of Values to Support Multiple Languages | 264 |

Chapter 11: Customizing Multi-Value Groups

- Process of Creating MVG Fields 269
 - Identifying Predefined MVG Objects in Siebel CRM 269
 - Process of Making Siebel CRM Data Available to Add an MVG 271
 - Process of Modifying the Customization Package to Add an MVG 276
 - Publishing and Testing a Custom MVG Field 279
 - Example Code You Use to Add an MVG 280

Chapter 12: Customizing Authentication

- Overview of Customizing Authentication 285
 - Authentication That Comes Predefined with Siebel CRM Desktop 286
 - Types of Authentication That You Can Use With CRM Desktop SSO 287
 - Single Sign On Services That CRM Desktop SSO Supports 290
- Installing CRM Desktop SSO 290
 - Setting Windows Registry Keys to Enable CRM Desktop SSO 292
 - Options for Installing CRM Desktop SSO 293
 - Removing or Upgrading CRM Desktop SSO 296
- About CRM Desktop SSO Architecture 297
 - Architecture That CRM Desktop SSO Uses 298
 - Flow That CRM Desktop SSO Uses During Authentication 299
 - Flow That the CRM Desktop SSO DLL Uses 301
 - Architecture That an SSO Session Uses 303
 - How CRM Desktop SSO Handles Errors 306
 - Modifying SSO JavaScript 307
- CRM Desktop SSO Objects You Can Customize 308
 - SSO Client Object 308
 - Logger Object 314
 - Settings Cache Object 314
 - Settings Object 315
 - Request Object 315
 - Response Object 315
 - Content Object 316
 - Header Object 317
 - Credentials Object 318
 - Interactive State Object 319
 - Dialog Object 320
 - Example Code That Customizes CRM Desktop SSO 320

Appendix A: Reference Information for Siebel CRM Desktop

- Registry Keys You Can Use with Siebel CRM Desktop 325

| | |
|---|-----|
| Registry Keys That Affect Siebel CRM Desktop Behavior | 325 |
| Registry Keys That Affect Credentials | 329 |
| Registry Keys That Affect CRM Desktop SSO | 330 |
| Parameters You Can Use with Log Files | 335 |
| Filters in the CRM Desktop Filter - Edit Criterion Dialog Box | 344 |
| Threshold That Siebel CRM Desktop Uses to Display the Confirm Synchronization Tab | 346 |
| Files That the Metadata and Customization Package Contains | 348 |
| Files in the Metadata | 348 |
| Files in the Customization Package | 355 |
| IBM Notes Field Types and Equivalent Converter Classes | 358 |

Appendix B: How Siebel CRM Desktop Maps Fields Between Siebel CRM Data and IBM Notes Data

| | |
|--|-----|
| How Siebel CRM Desktop Maps Fields Between Siebel Activities and IBM Notes Calendar | 363 |
| How Siebel CRM Desktop Maps Fields Between Siebel Activities and IBM Notes To Do Items | 366 |
| How Siebel CRM Desktop Maps Fields Between Siebel CRM Activities and IBM Notes Emails | 371 |
| How Siebel CRM Desktop Transforms Objects Between Siebel CRM Data and IBM Notes Data | 372 |
| How Siebel CRM Desktop Transforms a Calendar Entry That Does Not Repeat | 373 |
| How Siebel CRM Desktop Transforms a Repeating Calendar Entry That Matches a Siebel Repeating Pattern | 374 |
| How Siebel CRM Desktop Transforms a Repeating Calendar Entry That Does Not Match Siebel Repeating Patterns | 375 |
| How Siebel CRM Desktop Transforms Siebel CRM Activities That Do Not Repeat | 376 |
| How Siebel CRM Desktop Transforms Siebel CRM Activities That Repeat | 377 |
| How Siebel CRM Desktop Maps Fields Between a Siebel Calendar Entry and a IBM Notes Calendar Entry | 379 |

Appendix C: XML Files Reference

| | |
|--|-----|
| Getting Information About Tags of the Metadata Files | 381 |
| XML Code That Maps a Field | 381 |
| Example Code of the Siebel Basic Mapping File | 382 |
| Type Tag of the Siebel Basic Mapping File | 383 |
| Field Tag of the Siebel Basic Mapping File | 383 |
| Writer Tag of the Siebel Basic Mapping File | 383 |

| | |
|--|-----|
| XML Code That Customizes Platform Configuration | 384 |
| XML Code That Customizes Synchronization | 384 |
| Example Code of the Connector Configuration File | 385 |
| Types Tag of the Connector Configuration File | 385 |
| Type Tag of the Connector Configuration File | 385 |
| View Tag of the Connector Configuration File | 386 |
| Synchronizer Tag of the Connector Configuration File | 386 |
| Links Tag of the Connector Configuration File | 386 |
| Natural Key Tag of the Connector Configuration File | 387 |
| Filter Presets Tag of the Connector Configuration File | 388 |
| | 389 |

Glossary

Index

1

What's New in This Release

What's New in Siebel CRM Desktop for IBM Notes Administration Guide, Version 3.4, Rev. A

This guide has been updated to correct or remove obsolete product and component terms.

2

Overview of Siebel CRM Desktop for IBM Notes

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

- [Benefits of Using Siebel CRM Desktop on page 15](#)
- [Scenarios for Using Siebel CRM Desktop on page 15](#)
- [Overview for Using This Book on page 18](#)

Benefits of Using Siebel CRM Desktop

Oracle's Siebel CRM Desktop for IBM Notes (Siebel CRM Desktop) is an application that integrates Siebel CRM processes and data in IBM Notes. It allows the user to do typical CRM work directly in a native IBM Notes environment. It centralizes essential business information in the familiar IBM Notes environment. This centralization complements the existing capabilities that a Siebel Business Application provides. The user can use Siebel CRM Desktop to do the following work:

- **Manage Siebel CRM data.** Manage Siebel CRM data and link this data to CRM records directly in IBM Notes. The user can manage a calendar entry, emails, contacts, accounts, activities, opportunities, and so on directly in IBM Notes.
- **Synchronize data.** Perform bidirectional, incremental synchronization between Siebel CRM Desktop in the IBM Notes client and the Siebel Server. This synchronization helps to keep data up-to-date and consistent.
- **Work while disconnected.** Perform work even if disconnected from the corporate network.

Your organization can realize the following benefits:

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

Scenarios for Using Siebel CRM Desktop

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

- [Scenario for Working with an Opportunity on page 16](#)
- [Scenario for Managing Contact Information on page 16](#)

- [Scenario for Managing Account Information on page 17](#)
- [Scenario for Creating a Relationship Between an Email and an Opportunity on page 17](#)
- [Scenario for Managing an Opportunity on page 18](#)

The scenarios in this topic give examples of how you might use Siebel CRM Desktop. You might use Siebel CRM Desktop differently, depending on your business model.

Scenario for Working with an Opportunity

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 IBM Notes to assign the To Do item to a sales representative:

- Creates a new IBM Notes To Do item
- Creates a relationship between an opportunity and the To Do item and completes other To Do item details
- Sends the To Do item to the assigned owner

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

Scenario for Managing Contact Information

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. This company is a competitor of Company Y. The sales representative also encounters an old college friend and they trade contact information.

The sales representative creates a new contact in IBM Notes 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 CEO's business card and then attaches the scanned image to the contact. The representative must share this contact with colleagues, so the representative clicks the Share Bar. Siebel CRM Desktop then marks the contact as shared and toggles the Share Bar to indicate that the contact is shared.

In the same Contacts folder, the sales representative creates a new contact for the old college friend. The default option does not share the contact with the Siebel Server and 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 IBM Notes 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 uses the Siebel Web Client to update the contact information for the VP. When the sales representative synchronizes, Siebel CRM Desktop displays this updated information directly in IBM Notes.

Scenario for Managing Account Information

Company Z is one of the smaller accounts that a sales representative manages. This company recently relocated and the representative must update the address details that are related to 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 sends an email to the account team. The representative uses the Email to Account Team button in IBM Notes to include all account team members. 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 CRM Desktop displays the names of contacts who already exist. The representative chooses one of these contacts and then clicks Add to create a relationship. As an alternative, the representative can click the SalesBook icon and then choose an existing contact or create a new contact.

Scenario for Creating a Relationship Between an Email and an Opportunity

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, so the representative shares this email with the Siebel Server and then relates it with the opportunity. This work makes sure that the entire sales team who is working on this 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 includes a relationship 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 that the email describes. To create a relationship between the Big Deal opportunity and the email, the representative types the name in the Opportunity control. If the external contact is related to at least one opportunity, then 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 CRM Desktop links the email to the opportunity.

Scenario for Managing an Opportunity

After meeting with a contact at a key strategic account, a sales representative learns of an upcoming request for proposal 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 IBM Notes.

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

If the representative saves these details in IBM Notes and then synchronizes with the Siebel Server, then Siebel CRM makes the details available to other users who can access the account and contacts.

Overview for Using This Book

This book uses the following terms, unless noted otherwise:

- The *client* is IBM Notes with the Siebel CRM Desktop plug-in installed.
- A *user* is a person who uses the client.
- The *server* is the Siebel Server.
- An *administrator* is anyone who uses an administrative screen in the client to configure CRM Desktop. The Administration - Server Configuration screen is an example of an administrative screen.

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

This is computer font

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

Named Method *n*: *method_name*

The following is an example of this code:

Named Method 2: WriteRecord

A *predefined object* is an object that comes already defined with Siebel CRM. The objects that Siebel Webtools displays in the Object List Editor immediately after you install the product, but before you make any customizations, are predefined objects.

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

Getting Help From Oracle

For help with using object types, 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. You can also contact your Oracle sales representative to request assistance from Oracle's Professional Services.

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 21](#)
- [How Siebel CRM Desktop Uses the Siebel Enterprise on page 26](#)
- [Metadata That Siebel CRM Desktop Uses on page 32](#)

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 IBM Notes User Interface on page 21](#)
- [Infrastructure That Siebel CRM Desktop Uses on page 22](#)
- [Architecture That Siebel CRM Desktop Uses on page 23](#)

Extensions to the IBM Notes User Interface

Siebel CRM Desktop is a composite application that displays Siebel CRM sales data in IBM Notes. To store and display Siebel CRM data, the IBM Notes add-in framework deploys Siebel CRM Desktop to IBM Notes and extends the IBM Notes data model and user interface. Extensions to the IBM Notes user interface allow the user to display Siebel CRM data. The following are some examples of these extensions:

- Custom Actionbar buttons.
- Custom menu items.
- Custom forms that display Siebel CRM data.
- Custom controls that are embedded in IBM Notes forms that display Siebel CRM data.
- Personalization options dialog box.

IBM Notes uses these extensions to allow the user to do a variety of work. The following are some examples of work that the user can do:

- Create new Siebel CRM data in IBM Notes.
- Mark the IBM Notes item to share with the Siebel Server data and related sales data. As an option, it can share or unshare a calendar entry, To Do item, contact, or email message.
- View and edit sales data.

- Start a standard IBM Notes action, such as sending an email or scheduling a meeting in the context of a sales item.

The following are some examples of validation that Siebel CRM Desktop can do when the user enters data:

- Make sure the data type is valid for a given field.
- Make sure each required field includes information.
- Make sure some 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.

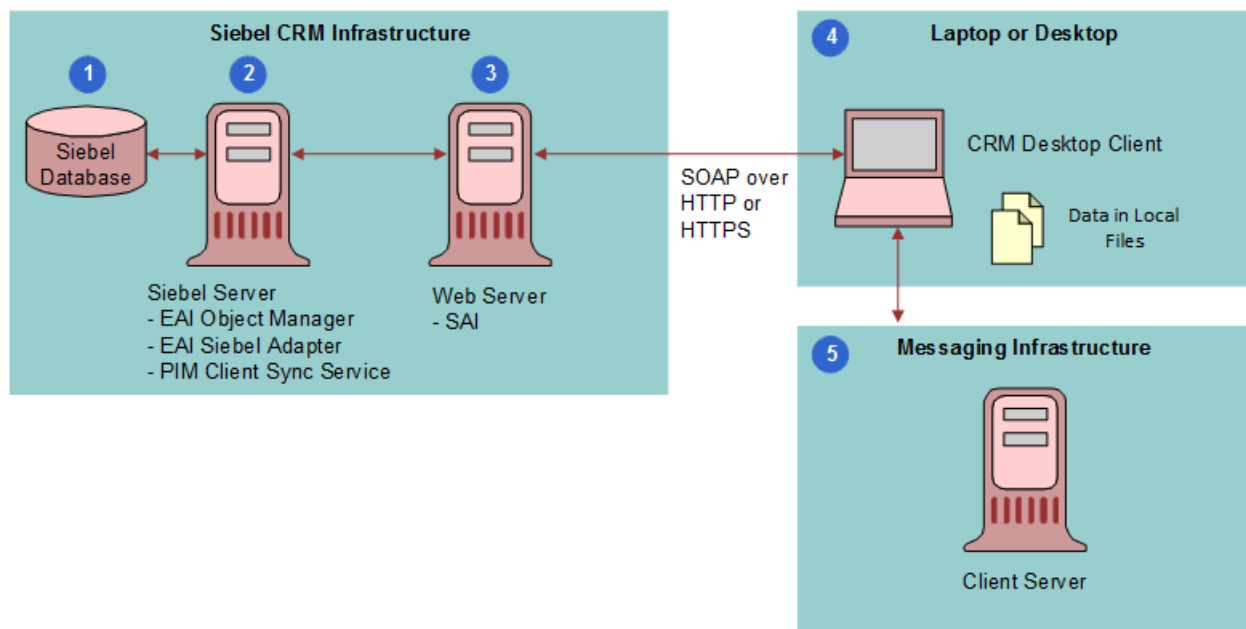


Figure 1. Infrastructure That Siebel CRM Desktop Uses

Explanation of Callouts

The infrastructure that Siebel CRM Desktop uses includes the following items:

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

- 2 **Siebel Server.** Runs the server components for CRM Desktop and manages synchronization sessions with the Siebel Application Interface. Hosts the processes that CRM Desktop requires to support synchronization and handles incoming synchronization requests from the client through the Siebel Application Interface. For more information, see [“How Siebel CRM Desktop Uses the Siebel Enterprise” on page 26](#).
- 3 **Siebel Application Interface.** Acts as the Web server 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 28](#).
- 4 **Laptop or Desktop.** The computer where CRM Desktop is installed. Oracle implements CRM Desktop as the IBM Notes add-in. CRM Desktop deploys a binary add-in during installation that supports integration with IBM Notes, custom user interface capabilities, and the Synchronization Engine. CRM Desktop metadata is available in IBM Notes after you download and install the customization package from the Siebel Server. Siebel CRM data is available on the client computer after the first synchronization finishes. The customization package includes features that allow synchronization and customization. For more information, see [“Customizing the First Run Assistant” on page 89](#).
- 5 **Messaging Infrastructure.** Handles local email, calendar, contacts, and To Do items. The messaging infrastructure provides support for mobile access and Web access to information and for storing data. For more information, see [“How Siebel CRM Desktop Stores Siebel CRM Data” on page 25](#).

Architecture That Siebel CRM Desktop Uses

Figure 2 illustrates the architectural components that Siebel CRM Desktop uses.

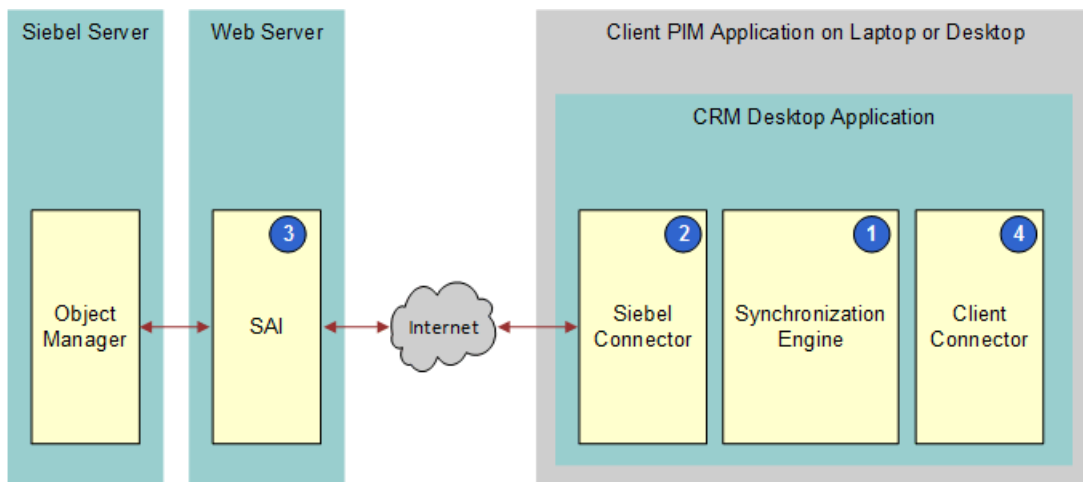


Figure 2. Architecture That Siebel CRM Desktop Uses

Explanation of Callouts

CRM Desktop uses the following architecture components:

- 1 **Synchronization Engine.** Starts the synchronization process. Determines the changes that 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, it submits requests to the connector and then, to determine the required data changes, it processes the replies. It works with the IBM Notes connector to make the necessary data changes in the data storage in IBM Notes.
- 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 **Siebel Application Interface (SAI).** Brokers requests from the Siebel Connector to the Siebel Server.
- 4 **Client Connector.** Allows the Synchronization Engine to access the data storage in IBM Notes. Supports queries, inserts, updates, and deletes of data in this data storage.

Overview of How Siebel CRM Desktop Synchronizes Data

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

The user must do the first synchronization with the Siebel Server to make Siebel CRM data available in IBM Notes. *First Run Assistant* is a wizard that guides the user through the setup of the CRM Desktop add-in in IBM Notes. It displays when the user starts IBM Notes for the first time after the CRM Desktop add-in is installed. It starts the first synchronization.

NOTE: You must restart the IBM Notes client after installing the CRM Desktop add-in. The Personal Address Book and Mail databases should be reopened after applying customizations or after the first synchronization, if they were opened during customizations.

The user can choose among several preferences and then start the first synchronization while using the First Run Assistant. CRM Desktop does the following work:

- 1 Connects IBM Notes to the Siebel Server and then authenticates the user.
- 2 Displays a prompt to the user that describes the location where CRM Desktop will apply the IBM Notes customization.
- 3 Determines the configuration that the user can access. A relationship with a responsibility that is related to a customization package determines this access. For more information, see [“Relationships Between Users, Responsibilities, Customization Packages, and Metadata Files” on page 32](#).
- 4 Downloads and applies the configuration.
- 5 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 63](#).

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 (Enterprise Application Integration) runtime repository objects, such as integration objects and integration components. These objects acquire data through their relations with business objects and business components.

About Siebel CRM Desktop and IBM Notes Data

IBM Notes *data* is data that the user creates in the native IBM Notes application. Examples include a calendar entry or To Do item. *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 Business 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. CRM Desktop uses native IBM Notes data files, so IBM Notes displays Siebel CRM data through native IBM Notes user interface elements, such as lists and forms. IBM Notes can display this data simultaneously with other IBM Notes data while using the same user interface concept, such as a mailbox folder. The user can choose a folder that displays Siebel CRM data and can also view IBM Notes data in the IBM Notes list view.

Siebel CRM Desktop displays Siebel CRM data in the IBM Notes documents that open according to the design element that it associates with the document.

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

How Siebel CRM Desktop Stores Siebel CRM Data

Figure 3 illustrates how Siebel CRM Desktop stores Siebel CRM data.

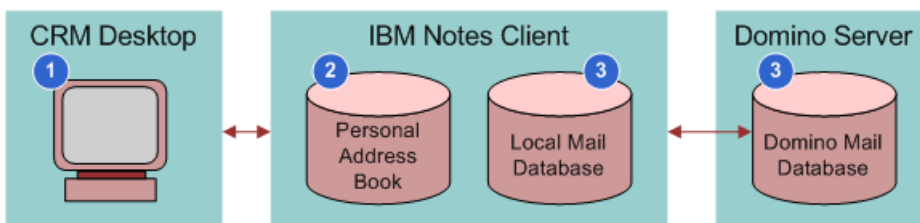


Figure 3. How Siebel CRM Desktop Stores Siebel CRM Data

Explanation of Callouts

Siebel CRM Desktop stores Siebel CRM data in the following way:

- 1 You install CRM Desktop as an add-in with IBM Notes on the client computer.

- 2 CRM Desktop stores data in the Personal Address Book (PAB) database. This data can include the following items. It can also include other data:
 - Custom objects.
 - Design elements that CRM Desktop uses to display custom objects.
 - Predefined Notes forms.
 - Lookup data, such as lists of values and currencies.
 - Relational data, such as contact and account intersection records.
- 3 The mail database stores tasks, appointments, and email messages. This database can reside locally on the Notes client or on the Domino Server. CRM Desktop does not interfere with communication between Notes and the Domino Server. Notes synchronizes with the Domino Server the same way it does if you do not install CRM Desktop.

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 27](#)
- [About the Web Service API on page 28](#)
- [About the PIM Client Sync Service Business Service on page 29](#)
- [About the EAI Siebel Adapter Business Service on page 29](#)
- [About Integration Objects on page 30](#)
- [User Details Business Component on page 31](#)
- [About Authentication and Session Management on page 31](#)

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 27](#).

Siebel Enterprise Components That Siebel CRM Desktop Uses

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

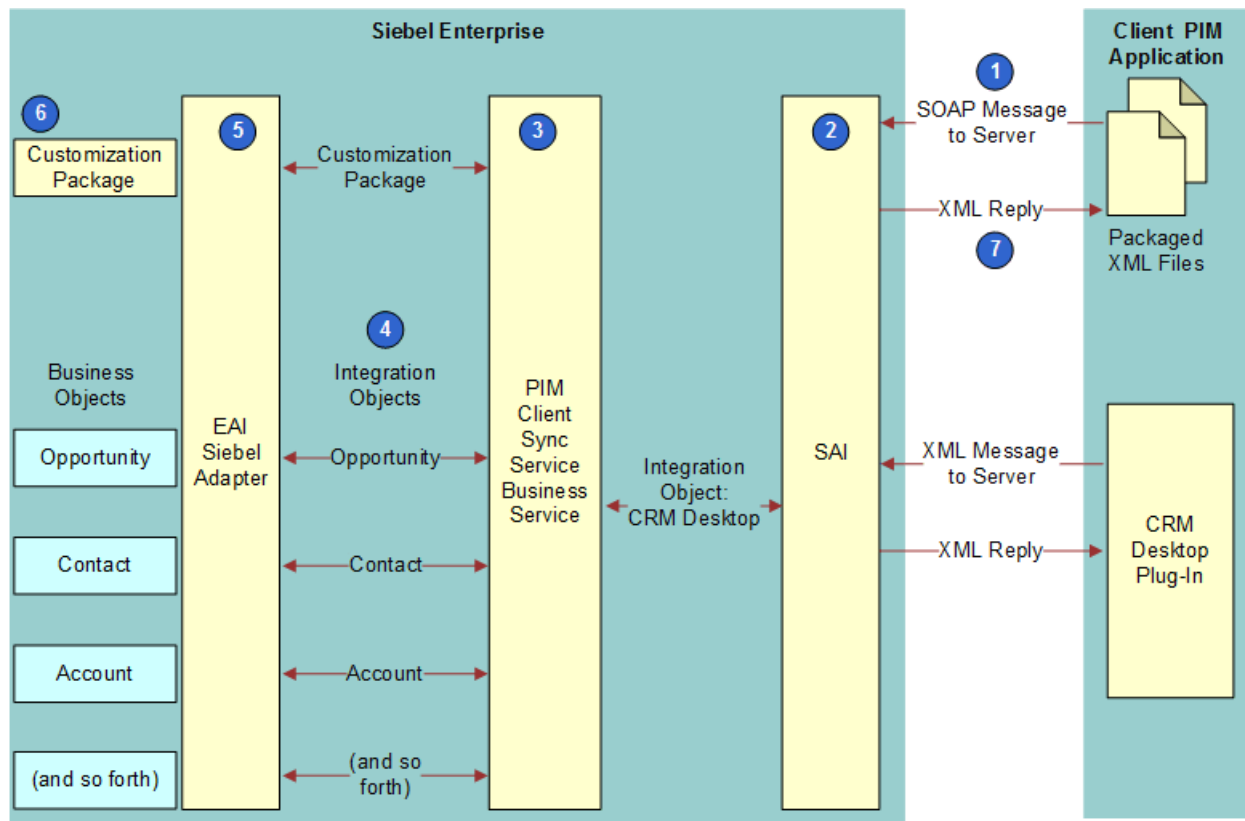


Figure 4. Siebel Enterprise Components That Siebel CRM Desktop Uses

Explanation of Callouts

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 it sends in a SOAP envelope.
- 2 Siebel Application Interface.** To handle requests from IBM Notes for data and metadata synchronization, 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 IBM Notes.

- Uses the configured endpoint to connect to the EAI Object Manager server component or another server component that provides access to the Web services.

For more information, see [“About the Web Service API” on page 28](#), and *Configuring Siebel Business Applications*.

- 3 PIM Client Sync Service Business Service.** Handles the data synchronization request from the Synchronization Engine and then passes this 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 record counts and resolving calendar entry attendees that reside on the Siebel Server. For more information, see [“About the PIM Client Sync Service Business Service” on page 29](#).
- 4 Integration Objects.** Requests processes through the EAI Siebel Adapter business service. It uses integration objects to describe the underlying Siebel data structure. It specifies objects and fields that are available for synchronization. For more information, see [“About Integration Objects” on page 30](#).
- 5 EAI Siebel Adapter.** Sends requests to the data and metadata synchronization Web services. These services use the Execution method and various operations to process the requests through the EAI Siebel Adapter business service. Example operations include querypage, insert, update, and delete. For more information, see [“About the EAI Siebel Adapter Business Service” on page 29](#).
- 6 Customization package.** A collection of XML, JavaScript, DXLfiles that describe the CRM Desktop add-in that runs in IBM Notes. 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 33](#).
- 7 XML Reply to IBM Notes.** Sends an XML message that contains the metadata that CRM Desktop requests in [Step 1](#). It sends this message to IBM Notes over HTTP or HTTPS.

For more information, see [“How Siebel CRM Desktop Synchronizes Data Between the Client and the Siebel Server” on page 63](#).

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. IBM Notes communicates with the Web Service API on the Siebel Server. This Web Service API references the PIM Client Sync Service business service. Communication between IBM Notes and the Siebel Server occurs through SOAP packages that include 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 IBM Notes. For more information, see [“About Integration Objects” on page 30](#).
- PIM Client Sync Service business service that supports synchronization with IBM Notes.
- 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 object attributes 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 IBM Notes and contains methods that handle calls from CRM Desktop. Each method is generic and is not tightly coupled to a specific Siebel object. A Web service provides access to the business service methods. CRM Desktop accesses the Web service through SOAP messages. The business service receives and sends the custom integration object as method parameters of the business service.

The PIM Client Sync Service business service does the following work:

- 1 Parses the input hierarchy of the object instance.
- 2 Retrieves commands from the Siebel message. These commands 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 integration object instance.

To send the information back to IBM Notes through the web service interface, the PIM Client Sync Service business service does this step.

For more information, see [“About Integration Objects” on page 30](#).

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. The following work occurs:

- 1 The EAI Siebel Adapter business service does the following work:
 - a Takes, as input, an XML document or a property set that conforms to the definition of an integration object in Siebel CRM.
 - b Queries, inserts, updates, deletes, or synchronizes data with the Siebel business object layer.
- 2 The PIM Client Sync Service web service calls the PIM Client Sync Service business service.
- 3 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 30](#).

About Integration Objects

An *integration object* is an object that includes the contents of the messages that Siebel CRM Desktop exchanges between the Siebel Server and IBM Notes.

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. This 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. If CRM Desktop sends an integration object instance from IBM Notes, then Siebel CRM encapsulates this object in an element of the Siebel message. Each of these messages includes 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:

- You must not create a new integration object for accounts. Instead, you can extend the CRMDesktopAccountIO integration object.
- You must create a new integration object for Channel Partner. For more information, see [“Creating an Integration Component for the Channel Partner MVG” on page 273](#).

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

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

CRM Desktop uses the following integration objects to meet other integration requirements:

- **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 this 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 allows CRM Desktop to access the following fields in IBM Notes:

- 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, 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, CRM Desktop performs formal authentication before it runs the Web service operation. If the authentication succeeds, then the operation proceeds and CRM Desktop places a special SessionID token in the SOAP header of the Web service reply.

When IBM Notes includes the SessionID in subsequent Web service requests, CRM Desktop uses this SessionID to restore cached session information. This configuration bypasses the substantially more expensive process of running the authentication again. If presented with the SessionID and a valid set of user credentials, then 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 Siebel CRM Desktop Uses

This topic describes the structure of the metadata that Siebel CRM Desktop uses and how you can modify it to support a customization. It includes the following topics:

- [Relationships Between Users, Responsibilities, Customization Packages, and Metadata Files on page 32](#)
- [About the Customization Package on page 33](#)
- [About Metadata Files on page 34](#)
- [About Metadata Administration on page 35](#)

For more information, see [“Overview of Customizing Siebel CRM Desktop” on page 153](#).

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

The responsibility that a customization package references creates a relationship between a Siebel CRM Desktop user and a CRM Desktop configuration. If the package is activated and published, then a user that the responsibility references can download the configuration that the package defines. This configuration is a collection of metadata files that CRM Desktop stores on the Siebel Server and downloads to IBM Notes during synchronization.

Figure 5 includes 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 to several customization packages, P1, P2, and P3.

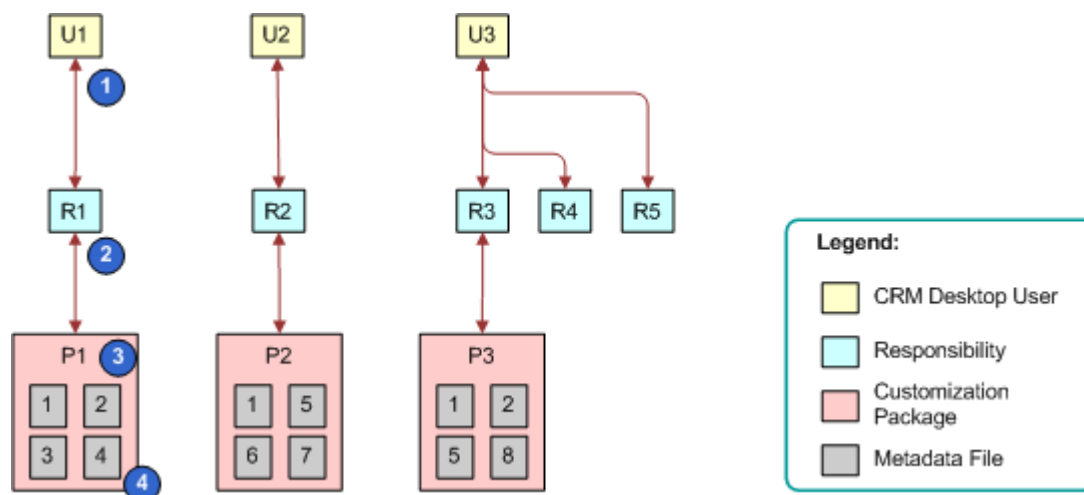


Figure 5. Example of Relationships Between Users, Responsibilities, Customization Packages, and Metadata Files

Explanation of Callouts

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. CRM Desktop creates a relationship between these files and a responsibility. For more information, see [“About the Customization Package” on page 33](#).
- 4 **Metadata File.** A description of CRM Desktop that CRM Desktop deploys to IBM Notes as XML code or JavaScript files. For more information, see [“About Metadata Files” on page 34](#).

How Siebel CRM Desktop Allows Users to Access Siebel CRM Data

Siebel CRM Desktop allows the user to access Siebel CRM data in IBM Notes in the following ways:

- **Through responsibility.** Similar to how a view allows the user to access data in the Siebel Web Client, CRM Desktop uses a responsibility to create a relationship between the user and a customization package. This relationship identifies the application metadata that CRM Desktop sends to the user through metadata synchronization. The metadata defines the objects that CRM Desktop synchronizes with IBM Notes. For more information, see [“Guidelines for Assigning Responsibilities to Customization Packages” on page 79](#).
- **Through synchronization filters.** The customization package includes metadata files that specify the data access control and the filters to apply when 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 related to the sales team that Siebel CRM assigns to the user.

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

About the Customization Package

A *customization package* is a collection of XML metadata files, .dxl files, and JavaScript files that Siebel CRM Desktop uses with a responsibility. CRM Desktop deploys a customization package when the user synchronizes the metadata. This synchronization identifies the customization package that is available to the user. You can modify the metadata files to customize your deployment. The following items are some examples of the customizations that you can make:

- Add or remove fields that CRM Desktop synchronizes.
- Change the design that CRM Desktop uses to display a custom form in the client.
- Change a control that CRM Desktop deploys to IBM Notes.
- Change a security rule.

The customization package describes the following information:

- The extensions to the CRM Desktop user interface. This interface includes IBM Notes views, forms, custom elements on standard IBM Notes forms, lookup controls, and Action bar buttons.
- Translated text strings that CRM Desktop uses to create prompts and labels in IBM Notes.
- Data validation and security logic.
- Descriptions of synchronization preset filters and view modes that CRM Desktop uses during synchronization.
- Criteria that CRM Desktop uses to detect duplicate objects.
- Business logic that JavaScript provides.
- Data mapping between Siebel fields and IBM Notes fields.

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

For more information, see [Chapter 9, “Customizing Siebel CRM Desktop.”](#)

About Metadata Files

A *metadata file* is an XML, DXL, or JavaScript file that Siebel CRM Desktop uses to display Siebel CRM data and user interface behavior. CRM Desktop uses these files in the following ways:

- **XML files.** Describes the default synchronization objects, synchronization mapping, custom views and forms in IBM Notes, and so on.
- **DXL files.** Includes design elements that IBM Notes uses to display an object. For example, it uses a DXL file to display the Sharing Bar on a predefined IBM Notes calendar entry form.
- **JavaScript files.** Describes business logic that CRM Desktop uses for data validation, custom actions that it provides access to in toolbars, and other custom processing that it does in IBM Notes.

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

- A customization package includes a collection of metadata files. These files describe the entire CRM Desktop add-in that you deploy to IBM Notes.
- A customization package consists of a set of metadata files.
- CRM Desktop requires that a customization package include a minimum number of files. These files are described in [“Files in the Customization Package” on page 355](#).
- 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 32](#) illustrates this relationship where the same metadata file occurs in different packages. For example, packages 1 and 3 include metadata file 2.
- CRM Desktop creates a relationship between a customization package and a single Siebel responsibility. It creates a relationship between the user and this responsibility so that the user can access the customization package and the CRM Desktop configuration that you deploy to IBM Notes.

For more information, see [“Files in the Customization Package” on page 355](#).

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

Siebel CRM Desktop creates a relationship between each customization package and a collection of metadata files, and it can use each metadata file with more than one customization package. You cannot modify the metadata file after CRM Desktop creates a relationship between this metadata file and an active deployment package. You can export, modify, and then reimport the metadata file to create a new metadata file. Although multiple customization packages can reference the same metadata files, one or more metadata files typically use different customization packages that include different information.

For example, the IBM Notes Sales Representative responsibility is different and separate from the IBM Notes Sales Manager responsibility. Although you can create a relationship between an existing responsibility and a customization package, it is recommended that you create a new responsibility. This configuration provides you with more control in determining the users that CRM Desktop uses with a customization package.

For more information, see [“Using the Windows Registry to Control Siebel CRM Desktop” on page 103](#).

About Metadata Administration

You use metadata administration to create and manage a customization package. You can do 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.
- Create a relationship between a metadata file and a customization package.
- Download the necessary metadata files through web services for the user.
- Track the expiration of a customization package and files.
- Control user privileges.
- Control user access to a customization package.

For more information about:

- Work you do to administer metadata during installation, see [“Administering Metadata Files” on page 78](#).
- A description of metadata files that you can administer, see [“About Metadata Files” on page 34](#).
- How CRM Desktop synchronizes metadata, see [“How Siebel CRM Desktop Synchronizes Data Between the Client and the Siebel Server” on page 63](#).

4

How Siebel CRM Desktop Handles Siebel CRM Data

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

- [How Siebel CRM Desktop Handles Activities on page 37](#)
- [How Siebel CRM Desktop Handles Shared Activities on page 45](#)
- [How Siebel CRM Desktop Handles IBM Notes Calendar Data on page 48](#)
- [How Siebel CRM Desktop Handles IBM Notes To Do items on page 59](#)
- [How Siebel CRM Desktop Handles IBM Notes Email Messages on page 59](#)
- [How CRM Desktop Displays Data That Is Not Directly Visible on page 60](#)
- [How a User Can Link Siebel CRM Records to IBM Notes Records on page 61](#)
- [How Siebel CRM Desktop Handles Items If the User Removes the CRM Desktop Add-In on page 62](#)

How Siebel CRM Desktop Handles Activities

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

- [Overview of How Siebel CRM Desktop Handles Activities on page 37](#)
- [How Activities Are Created or Modified on page 39](#)
- [How Siebel CRM Desktop Processes Activities on page 39](#)
- [How Siebel CRM Desktop Resolves Participants and Email Recipients of Activities on page 41](#)
- [How Siebel CRM Desktop Displays Activities in IBM Notes on page 43](#)
- [How Siebel CRM Desktop Sets the Primary Employee of Activities on page 43](#)
- [How Siebel CRM Desktop Handles Attachments on page 45](#)

Overview of How Siebel CRM Desktop Handles Activities

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
- A calendar entry that includes a contact

Siebel CRM can display an activity in the Activities screen or in the Calendar in the client of a Siebel Business Application, such as the Mobile Web Client. The Display In field of the Activities list determines where CRM Desktop displays an activity in IBM Notes. This field includes the following values:

- 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:

- Calendar Entry
- Field Repair
- Email-Outbound
- Research

Siebel CRM Desktop uses one of the following custom Siebel CRM activity objects to support an activity in IBM Notes:

- **Calendar entry.** This entry is a meeting or **calendar entry**.
- **To Do item.** This item is a To Do. For example:
 - Book a flight
 - Review new proposal
- **Email item.** This item is a record of communication. For example:
 - Correspondence was sent

Siebel CRM Desktop does not map a Siebel CRM activity to a single native object in IBM Notes. Instead, it synchronizes an activity from the Siebel Server to the client as a custom activity record rather than as the IBM Notes To Do item or calendar entry. After synchronization, CRM Desktop does the following:

- Creates the IBM Notes calendar entry that matches the calendar entry from Siebel CRM
- Creates the IBM Notes To Do item that matches the To Do item from Siebel CRM

If the activity is a Siebel CRM activity, then CRM Desktop creates a shared calendar entry in IBM Notes.

IBM Notes does not synchronize directly between native IBM Notes items and records on the Siebel Server, so CRM Desktop uses the Siebel CRM activity as an intermediary between a native IBM Notes item that resides in the user mailbox and a Siebel CRM activity that resides on the Siebel Server. If the user creates a shared IBM Notes calendar entry, email, or To Do item, then CRM Desktop creates another item in IBM Notes that represents the Siebel CRM activity record in addition to the shared native IBM Notes item.

To support this configuration, CRM Desktop uses an activity object as a proxy to synchronize all activities, regardless of type. It does the following:

- Parses each activity when it downloads this activity into calendar, email, or IBM Notes To Do objects. For example, it parses a calendar entry into the IBM Notes Calendar.
- If the user modifies a native IBM Notes item, then CRM Desktop modifies a hidden activity object that contains the information that the object requires, such as a description, start time, relations to other objects, and so on. CRM Desktop synchronizes this hidden object. It does not synchronize the native IBM Notes item. CRM Desktop uses this same configuration for a native IBM Notes item that the user creates.

How Activities Are Created or Modified

The user can do one of the following to read, update, create, or delete records:

- Use the Add Activity button.
- Use a form for an item in IBM Notes 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 IBM Notes, 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 IBM Notes, such as a calendar entry, a To Do item, or an email.

How Siebel CRM Desktop Processes Activities

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

- A native IBM Notes item, such as a calendar entry, an email, or a To Do item
- A Siebel CRM activity record in IBM Notes that CRM Desktop synchronizes from the Siebel Server
- A Siebel CRM activity record on the Siebel Server

Figure 6 illustrates the relationships between IBM Notes items in IBM Notes, Siebel CRM records in IBM Notes, 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 on. For brevity, Figure 6 does not include these types.

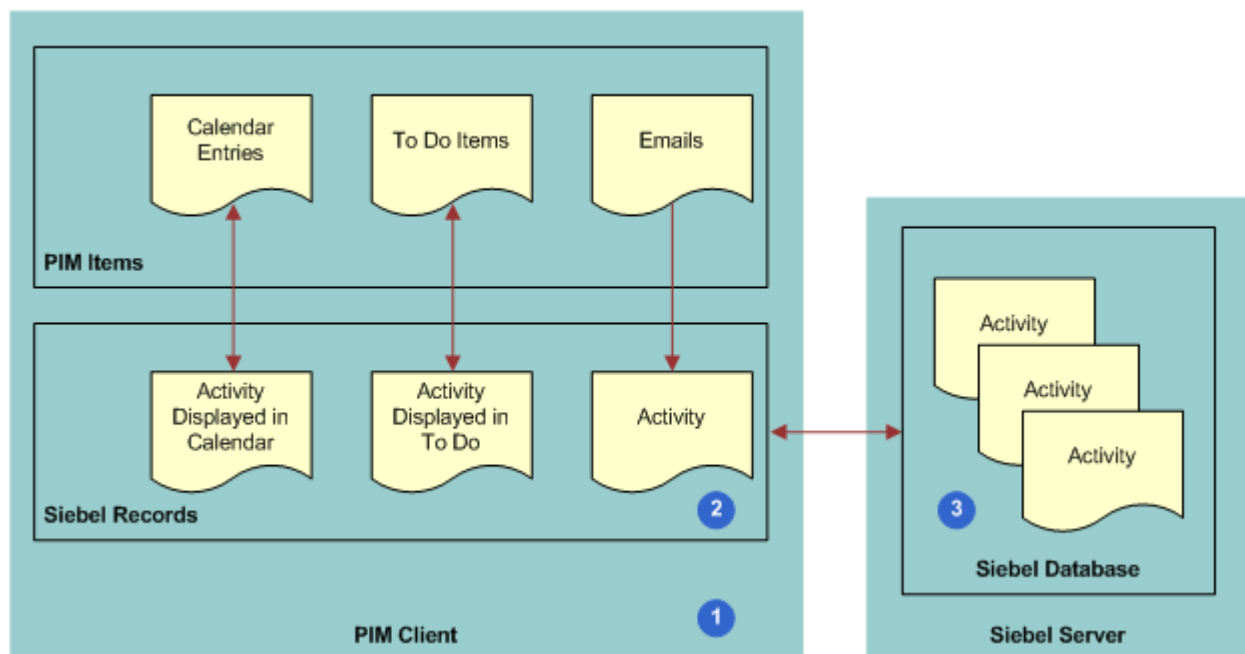


Figure 6. Relationship Between IBM Notes Items, Siebel Activities in IBM Notes, and Siebel Activities on the Siebel Server

Explanation of Callouts

Siebel CRM Desktop does the following work to process an activity:

- 1 An activity is created in IBM Notes. For more information, see ["How Activities Are Created or Modified" on page 39](#).
- 2 Siebel CRM Desktop adds a record as a Siebel CRM activity in IBM Notes.
If the user marks the IBM Notes calendar entry, email, or To Do item as shared, and then saves and closes this item, then CRM Desktop immediately creates the Siebel CRM activity in IBM Notes.
- 3 During synchronization, CRM Desktop maps the Siebel CRM activity in IBM Notes one-to-one with the corresponding activity in the Siebel database on the Siebel Server. If the user shares a new IBM Notes item or creates an activity in IBM Notes, then 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 Corresponding Native IBM Notes Items" on page 41](#).

How Siebel CRM Desktop Creates Corresponding Native IBM Notes Items

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 IBM Notes any new activities that reside on the Siebel Server.

[Table 1](#) describes how CRM Desktop creates a corresponding native IBM Notes item depending on the settings of the Display In field.

Table 1. How Siebel CRM Desktop Creates a Corresponding Native IBM Notes item

| Display In Value for the Siebel CRM Activity | Description |
|---|--|
| Calendar and Activities | CRM Desktop creates a calendar entry as a calendar entry only. If the Siebel Start date is not set for the activity, then CRM Desktop does not create a calendar entry in IBM Notes Calendar. The calendar entry is linked to this activity. |
| To Do and Activities | CRM Desktop creates a native IBM Notes To Do item that is linked with this activity. |
| Not Calendar and Activities or To Do and Activities | CRM Desktop synchronizes the activity as visible under the appropriate parent object, such as an Account Activity, Contact Activity, and so on. |
| Activities Only | If an activity is Activities Only, and if this activity is not related to another item that the user can view, such as an account or contact, then CRM Desktop synchronizes the activity but does not display it in the 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 Activities

This topic describes how Siebel CRM Desktop resolves participant lists and email recipients in the IBM Notes calendar.

How Siebel CRM Desktop Resolves Meeting Attendees

Siebel CRM Desktop does the following work:

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

- If the meeting organizer uses an MVG in the activity form to add a relationship, then it does not update the To line. This configuration allows the user to create a relationship between the Siebel CRM activity and a contact but not invite the contact to the meeting.

How Siebel CRM Desktop Resolves Owners and Assignees

Siebel CRM Desktop does the following work to resolve To Do item owners and To Do item assignees:

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

How Siebel CRM Desktop Resolves Email Recipients

Siebel CRM Desktop does the following work:

- If a user manually shares an email that another user manually sent to or received from a contact, then it does the following:
 - Sets the Owner of the activity to Generic Siebel Owner.
 - Resolves the recipients and sender of the email into contacts.
 - Suggests relationships 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 it creates a relationship between the account or opportunity and the activity that the user creates from the email.
 - If the user manually creates a relationship between the shared email and a Siebel CRM record before the user sends the email, and if a contact exists, and if the Save Correspondence feature is enabled for the email recipients of this contact, then the automail processing feature does the following work:
 - Preserves the relationships that the user makes.
 - Updates the email activity with contact relationships that CRM Desktop resolves from the email addresses of the recipients.
- If an email activity is created automatically, then it does the following:
 - Sets the Owner of the activity to Generic Siebel Owner.
 - Resolves the email recipients and sender into contacts.
 - Creates a relationship between these contacts and the email activity.
 - Of these contacts, it creates the following relationships for the first contact it encounters that contains a check mark in the Save Correspondence check box:
 - Sets this contact as the primary for the activity
 - Creates a relationship between the primary account for this contact and the activity

- If, to create a relationship for a contact, the user uses the Contact MVG dialog box for an activity that is linked to a shared email, then CRM Desktop does not add email addresses to the To field. A relationship that the user creates with a contact does not update the recipients list for the email message.

How Siebel CRM Desktop Displays Activities in IBM Notes

Siebel CRM Desktop displays data for a Siebel CRM activity in IBM Notes in the following ways:

- For a shared calendar entry, email, or To Do item, it displays details of the related activity fields in the native IBM Notes form. For example, the native IBM Notes calendar entry form displays the following information:
 - The description of the Siebel CRM activity in the Subject field of the native IBM Notes calendar entry
 - The account that is related to this Siebel CRM activity in the Account field in the extended area of the form

For example, the user can use the native IBM Notes form to review and change the account, opportunity, contacts, and employees for the Siebel CRM activity that is related to the shared item.

- 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.

CRM Desktop does not display a folder in the user mailbox for an activity, by default. You can configure the metadata to make this folder visible. For more information, see [“Type Tag of the Siebel Basic Mapping File” on page 383](#).

How Siebel CRM Desktop Sets the Primary Employee of Activities

This topic describes how Siebel CRM Desktop sets the primary employee of an activity for a calendar entry or a To Do item.

How Siebel CRM Desktop Sets the Primary Employee of a Calendar Entry

Siebel CRM Desktop sets the Primary Owned By field of a calendar entry according to the following priority:

- 1 Resolves the email address of the native IBM Notes calendar entry to a Siebel CRM employee. If CRM Desktop finds an employee record that contains this address, then it sets the meeting organizer of the IBM Notes Calendar entry as the primary.

- 2 If CRM Desktop does not find an employee that contains this address, then it compares this address with addresses from email accounts in the IBM Notes location. 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 IBM Notes record for this employee.
- 3 If CRM Desktop does not find a match among the email accounts in the IBM Notes location, then no employee is found. In this situation, it sets the primary employee to the value in the Generic Siebel Owner system preference. For more information, see [“How Siebel CRM Assigns Meeting Organizers” on page 44](#).

For more information, see [“Controlling How Siebel CRM Desktop Assigns Calendar Entry Owners” on page 108](#).

How Siebel CRM Assigns Meeting Organizers

A *Siebel user* is a user who is registered to use Siebel CRM Desktop or a Siebel Business 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 it 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 it 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 [“Controlling How Siebel CRM Desktop Assigns Calendar Entry Owners” on page 108](#).

How Siebel CRM Assigns a 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 a calendar entry in IBM Notes. In this situation, an employee record exists for this user, so Siebel CRM Desktop sets this user as the owner and then synchronizes this calendar entry 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.

For more information, see ["Resolving Synchronization Conflicts" on page 148](#).

How Siebel CRM Desktop Sets the Primary Employee for a To Do Item

Siebel CRM Desktop does following work to set the primary employee for a To Do item:

- If the user assigns the To Do item to shared contacts and employees, then it does the following work:
 - Sets the creator as the Activity Owner
 - Creates a relationship between contacts and the Contacts list
 - Creates associations with other employees
- If a user receives and shares a To Do item, then CRM Desktop creates the activity, sets the Activity Owner to Generic Siebel User, and then adds the employee who shares the To Do item to the Employees team as a nonprimary member.

How Siebel CRM Desktop Handles Attachments

Siebel CRM Desktop uses the EAI Siebel Adapter business service to do upload, download, and delete operations on attachments. It 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 Shared Activities

This topic describes how CRM Desktop handles a shared activity. It includes the following topics:

- [How the Origin of an Activity Affects Handling on page 46](#)
- [How Siebel CRM Desktop Handles a IBM Notes Meeting That Includes Multiple Attendees on page 47](#)
- [How Siebel CRM Desktop Handles a Shared IBM Notes Calendar Entry That Is Declined on page 48](#)

IBM Notes supports the concept where relationships can exist between more than one user and the same meeting or To Do item. In this situation, Siebel CRM Desktop prevents the creation of duplicate Siebel activities. It makes sure that it creates a relationship between only one Siebel CRM activity and a single IBM Notes item that more than one user synchronizes. The first user who synchronizes the IBM Notes item creates the Siebel CRM activity on the Siebel Server. Siebel CRM Desktop links IBM Notes items with the Siebel CRM activity for any subsequent user who synchronizes.

To prevent duplicate records, CRM Desktop does the following:

- 1 Uses the unique identifier for the meeting and To Do item that IBM Notes 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, it validates that no other Siebel CRM activity includes this same unique identifier.
- 4 If it finds that there is no other activity, then it creates a new activity.
- 5 If it finds that there is another activity, then it downloads the existing activity with the same unique identifier, and then links it with the IBM Notes item.

For more information, see ["How Siebel CRM Desktop Avoids Duplicate Data" 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

If an item originates in Siebel CRM, then Siebel CRM creates the activity on the Siebel Server. When Siebel CRM Desktop downloads this activity from the Siebel Server, it creates a native IBM Notes item if the current user is the owner of the To Do item, or if the current user is in the meeting participant list. CRM Desktop creates the activity as a simple calendar entry. No additional handling occurs in IBM Notes.

How the Origin of an Activity Affects Handling if the Item Originates in IBM Notes

If an item originates in IBM Notes, then Siebel CRM Desktop creates an activity in IBM Notes, uploads it to the Siebel Server during synchronization, and then downloads it to another user during an incremental synchronization. When CRM Desktop downloads this activity from the server, it does not add an item in the IBM Notes calendar. Instead, it expects IBM Notes to create the necessary item in the user mailbox. To create this item, IBM Notes runs the native process that it uses to send a meeting invitation or to assign a To Do item.

If CRM Desktop does this work before it synchronizes the Siebel CRM activity with the Siebel Server, then it links the Siebel CRM activity with the meeting or To Do item 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 IBM Notes form. If the user shares the item, then the item might not include all the details that the meeting organizer or To Do item owner specified. These details only arrive after CRM Desktop synchronizes the Siebel CRM activity from the Siebel Server.

How Siebel CRM Desktop Handles a IBM Notes Meeting That Includes Multiple Attendees

The example in this topic describes how Siebel CRM Desktop handles a IBM Notes meeting that includes multiple attendees.

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

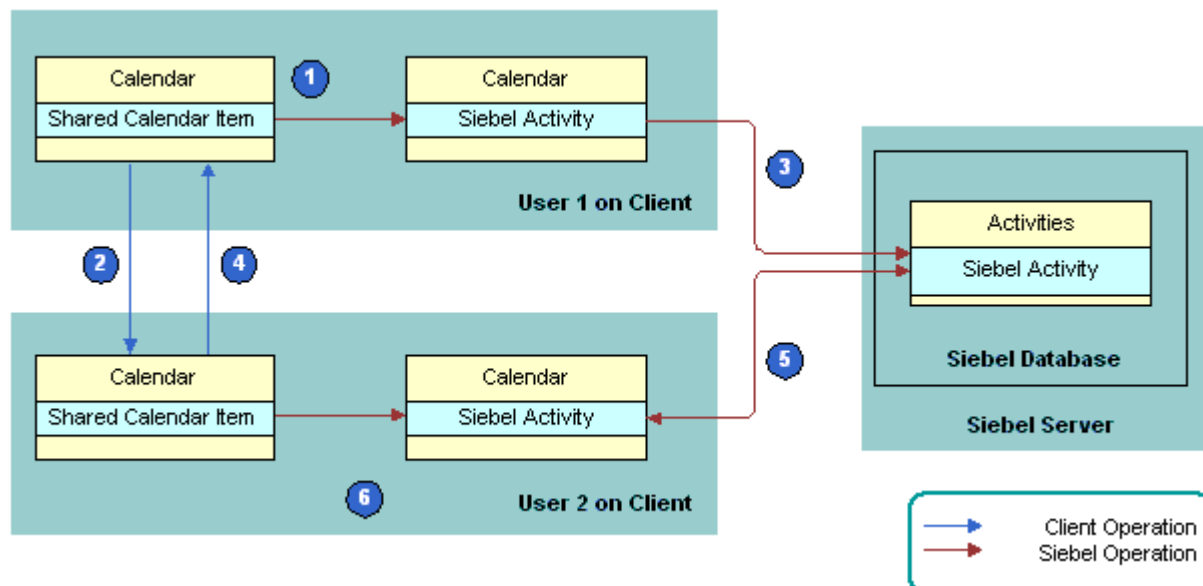


Figure 7. How Siebel CRM Desktop Handles the IBM Notes Meeting That Contains Multiple Attendees

Explanation of Callouts

The following work occurs:

- 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 UniversalId 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 IBM Notes meeting with the Siebel Server because these items use the same UniversalId and include the same meeting organizer. In this situation, Siebel CRM Desktop identifies a duplicate during synchronization.

Siebel CRM Desktop detects that the activities are equivalent and then does deduplication without displaying the collision dialog box. For more information, see [“Resolving Synchronization Conflicts” on page 148](#).

How Siebel CRM Desktop Handles a Shared IBM Notes Calendar Entry That Is Declined

The example in this topic describes how Siebel CRM Desktop handles a IBM Notes calendar entry that the meeting organizer shares and that the meeting attendee declines. In this situation, the user receives an invitation from another user through IBM Notes and then declines the invitation:

- 1 User 1, the meeting organizer, creates a calendar entry 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. CRM Desktop creates a shared meeting in the client of User 2. It shares the meeting because the preference for User 2 is to create new native IBM Notes items as shared. It 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 CRM Desktop deletes the calendar entry 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 CRM Desktop deletes the activity in IBM Notes 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, CRM Desktop updates IBM Notes, and removes user 2 from the employee team.
- 9 User 1 synchronizes. CRM Desktop synchronizes changes with the Siebel Server.
- 10 User 2 synchronizes.

How Siebel CRM Desktop Handles IBM Notes Calendar Data

This topic describes how Siebel CRM Desktop handles data in the IBM Notes calendar. It includes the following topics:

- [How Siebel CRM Desktop Handles IBM Notes Calendar Entries That Users Save, Change, or Delete on page 49](#)

- [How Siebel CRM Desktop Handles Siebel CRM Activities That Users Save, Modify, or Delete on page 49](#)
- [How Siebel CRM Desktop Handles a Calendar Entry on page 50](#)
- [How Siebel CRM Desktop Handles a Repeating Calendar Entry on page 58](#)

How Siebel CRM Desktop Handles IBM Notes Calendar Entries That Users Save, Change, or Delete

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

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

A Calendar entry that is set to a single All Day 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 IBM Notes Calendar” on page 363](#).

How Siebel CRM Desktop Handles Siebel CRM Activities That Users Save, Modify, or Delete

Siebel CRM Desktop internally creates a relationship between a Siebel CRM activity and a calendar entry or To Do item. CRM Desktop applies the following logic if a user saves, changes, or deletes a Siebel CRM activity:

- If the activity does not include a relationship with an item, then CRM Desktop attempts to find the related IBM Notes item, and then creates a relationship with the activity.
- If the activity exists in IBM Notes, then CRM Desktop links it to the corresponding IBM Notes item.
- If the activity originates as Siebel CRM data, and if CRM Desktop cannot find a correlation, then it creates a new IBM Notes item and creates a relationship between it and the activity. The type of IBM Notes item that it 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 it creates a calendar entry.
 - If the value in the Display In field is To Do and Activities, then it creates a To Do item.

The mapping that CRM Desktop creates between the IBM Notes calendar entry and the first Siebel CRM activity is the same as that described in [“How Siebel CRM Desktop Maps Fields Between Siebel Activities and IBM Notes Calendar” on page 363](#), with CRM Desktop doing the following additions:

- Sets the value in the Show Time As field of the native IBM Notes calendar entry to Busy
- Sets the calendar entry label to None

If the user modifies a Siebel CRM activity, then CRM Desktop does the following:

- If the user makes a simple modification, such as modifying the description, Start Date, and so on, then CRM Desktop synchronizes this modification to the CRM Desktop client the same way it synchronizes any other modification.
- If the user modifies a value in the Display In field, then CRM Desktop does a Delete operation and then a Create operation. For example, assume the Display In value is Calendar and Activities for a shared calendar item that the user created in Siebel. If the user modifies this value, then CRM Desktop synchronizes it to the IBM Notes calendar. If the user uses a Siebel client to modify this value to To Do and Activities, then the user must delete the IBM Notes Calendar item, and then create an IBM Notes To Do item item.

If the user deletes a Siebel CRM activity, the CRM Desktop does one of the following:

- If the record originated in Siebel, then CRM Desktop deletes it from IBM Notes.
- If the record originated in IBM Notes, then CRM Desktop unshares it.

How Siebel CRM Desktop Handles a Calendar Entry

This topic describes how Siebel CRM Desktop handles a calendar entry and a meeting. CRM Desktop can only share a calendar entry or a meeting with the Siebel Server. It cannot share or synchronize to the Siebel Server any other type of IBM Notes calendar entry.

How Siebel CRM Desktop Correlates Siebel CRM Activities with PIM Data in IBM Notes

When Siebel CRM Desktop synchronizes a Siebel CRM activity to IBM Notes, it attempts to find the PIM data that resides in IBM Notes that corresponds to the activity. PIM data is a calendar entry, To Do item, or email. If it finds this item, then it shares it and correlates it with the Siebel CRM activity. CRM Desktop does this correlation for the following items:

- The IBM Notes calendar entry for a Siebel CRM record where the Display In value of the Siebel CRM activity is Calendar and Activities.
- The IBM Notes To Do item where the Display In value of the Siebel CRM activity is To Do and Activities.
- Each IBM Notes activity where the Display In value of the Siebel CRM activity is Activities Only.

For example, to do a correlation for the IBM Notes Calendar entry, it uses the following keys:

- 1 Key 1:
 - a The CRMD Integration Id equals the UniversalId field of the Calendar entry or the \$MessageID field for an email. For more information, see [“How Siebel CRM Desktop Uses \\$MessageID” on page 53](#).
 - b The owner is the meeting organizer of the Calendar entry.
- 2 Key 2:
 - a The owner is the meeting organizer of the Calendar entry.
 - b The description is the subject of the Calendar entry.
 - c The value in the Planned field in the Siebel database equals the start time of the Calendar entry. This Planned value maps to the value that the Start field in IBM Notes contains.

If the user creates an activity in CRM Desktop from the IBM Notes Calendar entry, 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 it Synchronizes the Notes Activity as a Siebel Activity

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

- 1 User 1 does the following work:
 - a Creates a meeting in IBM Notes
 - 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. It uses the value from the UniversalId field of the meeting to populate the value in the CRMD Integration Id field in this activity record. This value is a unique value for this meeting. To identify the meeting organizer and the meeting participants, CRM Desktop uses the corresponding values in the IBM Notes meeting.

- 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 the PIM item that CRM Desktop shares with Siebel CRM. If user 2 receives the meeting invitation from user 1, then CRM Desktop does not share the meeting for this user in the calendar and it does not create a Siebel CRM activity.
- 4 When User 2 synchronizes, CRM Desktop synchronizes the activity that it added in [Step 2](#) to IBM Notes. It uses the FindPimDocument function to find the IBM Notes item that corresponds to this activity. It finds the unshared meeting because the following situations are true:
 - This meeting contains the same UniversalId 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 synchronizes 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 CRM Desktop uses the FindProxy function to find the Siebel CRM activity. If it finds this activity, then it shares the meeting with Siebel CRM.

How Siebel CRM Desktop Correlates Data if it Does Not Synchronize the Notes Activity as a Siebel Activity

Assume the following situation is true:

- To track activities, a user uses IBM Notes and a Siebel CRM application, such as Siebel Call Center.
- This user has not installed Siebel CRM Desktop.
- This user enters activities in IBM Notes and Siebel Call Center.
- The user has an activity in Siebel CRM. The user also has a calendar entry in IBM Notes that matches this activity. This activity and this calendar entry each include the same subject, start date, and activity owner.
- Assume this user installs Siebel CRM Desktop and then synchronizes.

In this situation, 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 CRM Desktop uses key 2. The following sequence describes how it uses key 2:

- 1 The CRMD Integration Id field is empty, so it skips key 1.
- 2 Correlates the activity:
 - a If the activity is a To Do item, then it uses Key 2 with the following differences:
 - The owner is the meeting organizer of the To Do item.
 - The description is the subject of the To Do item subject.For more information, see [Step 2 on page 51](#).
 - b If the activity is an email message, then it uses the following keys:
 - Key 2.2:

- CRMD Integration Id can contain part of the \$MessageID field of the email message. For more information, see [“How Siebel CRM Desktop Uses \\$MessageID” on page 53](#).
- Owner is the current user.
- Key 2.3
 - Uses the same information as Key 2.2, plus includes Planned, which is the date that CRM Desktop sends or receives the email. The value in the Planned field in the Siebel database equals the start time of the Calendar entry. This Planned value maps to the value the Start field in IBM Notes.

Note the following:

- If the Display In value of the Siebel CRM activity is To Do and Activities, then the activity is a To Do item.
- 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 \$MessageID

Siebel CRM Desktop uses the \$MessageID variable to identify the unique ID of an email message that it receives. The message syntax might vary depending on if it receives the email from the IBM Notes client or Domino Server, or from some other email client, such as Microsoft Outlook or Web mail. \$MessageID uses the following format:

`<OFhex_value_1-0Nhex_value_2-hex_value_3@domain_name>`

where:

- OF is the beginning of the hexadecimal code.
- *hex_value_1* is a hexadecimal value that includes two, eight digit values, where a period separates these values as a full stop. For example, 9264F990.72075AF4. The entire hexadecimal value typically identifies the originator of the email.
- *hex_value_2* is another hexadecimal value that uses the same format as the first hexadecimal value. \$MessageID ignores *hex_value_2*.
- *hex_value_3* is another hexadecimal value that uses the same format as the first hexadecimal value. \$MessageID ignores *hex_value_3*.
- *domain_name* identifies the domain name of the mail client of the sender. \$MessageID ignores *domain_name*.

Example \$MessageID Usage

Assume the sender mail box uses the Domino Server and the recipient mail box also uses the Domino Server. CRM Desktop uses the following \$MessageId in this IBM Notes to IBM Notes email message:

`<OF9264F990.72075AF4-0NC2257A3A.0059D96B-C2257A3A.0059E71C@Local Domain>`

where:

- 9264F990.72075AF4 is *hex_value_1*. It identifies the \$MessageId.

- C2257A3A.0059D96B is *hex_value_2*.
- C2257A3A.0059E71C is *hex_value_3*.
- LocalDomain is the *domain_name*.

\$MessageId uses this same syntax for the following communications:

- IBM Notes sender to a IBM Notes recipient
- IBM Notes sender to a POP3 recipient
- POP3 sender to a IBM Notes recipient

Example \$MessageID Usage With Multiple Recipients

If a user sends a single email message from a POP3 mail box in IBM Notes to two IBM Notes accounts, and if the first account uses the Domino Server and the second account uses POP3, then \$MessageId uses the following two, individual corresponding codes:

<0F19ABE5FC.FFEB1AEC-ONC2257A9A.004F2A95-C2257A9A.004F30C1@Local Domain>

<0F19ABE5FC.FFEB1AEC-ONC2257A9A.004F2A95-C2257A9A.004F30C1@ld8.example.com>

where:

- 19ABE5FC.FFEB1AEC is *hex_value_1*. It identifies the \$MessageId.

\$MessageID Usage With Web Mail

If a user sends an email message from a Web mail client, such as Gmail, to an IBM Notes email account that uses POP3 or the Domino Server, then CRM Desktop uses the following syntax:

<CAAZgRMT1vmOPfWQjnwNgJqC3xLNj-hgHA6tmEecha4sDN04+CA@mail.example.com>

where:

- CAAZgRMT1vmOPfWQjnwNgJqC3xLNj-hgHA6tmEecha4sDN04+CA is *hex_value_1*. It identifies the \$MessageId.
- mail.example.com is the *domain_name*.

\$MessageID Usage With Microsoft Outlook

If a user sends an email message from Microsoft Outlook to an IBM Notes email account that uses POP3 or the Domino Server, then CRM Desktop uses the following syntax:

<00fa01cdac74\$b5b0f770\$2112e650\$@Cheng@example.com>

where:

- 00fa01cdac74\$b5b0f770\$2112e650\$ is *hex_value_1*. It identifies the \$MessageId. In this example, @Cheng represents the user surname, such as Casey Cheng. \$MessageID ignores this surname.
- example.com is the *domain_name*.

How Siebel CRM Desktop Uses Natural Keys to Identify Duplicate Activities

Siebel CRM Desktop uses natural keys to detect a duplicate between IBM Notes data and Siebel CRM data. It uses the following natural keys for an activity:

- **Activity Owner and CRMD Integration Id.** These items match the UniversalId of a calendar entry.
- **Activity Owner and Description.** These items match the subject of the calendar entry and the Start Date.

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 code is an example of the natural keys that Siebel CRM Desktop might use in the Ln_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 [“Files in the Customization Package” on page 355](#).

How Siebel CRM Desktop Handles a Repeating Calendar Entry

IBM Notes uses more repeating patterns than the Siebel calendar uses. Siebel CRM Desktop establishes a correlation between IBM Notes and a Siebel CRM repeating pattern in the following way:

- If the IBM Notes pattern matches an existing Siebel CRM pattern, then CRM Desktop uses the corresponding repeating pattern to create a Siebel CRM activity.
- If the IBM Notes pattern does not match an existing Siebel CRM pattern, then 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 IBM Notes that occurs every two weeks for two months for a total of four meeting instances. If CRM Desktop attempts to synchronize this meeting with the Siebel Server, then it cannot directly support the repeating 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, CRM Desktop represents each occurrence that changed as a separate calendar entry in IBM Notes data.

CRM Desktop does the following work to handle a repeating calendar entry:

- Maps a repeating IBM Notes calendar entry to a repeating Siebel calendar entry
- Maps a repeating Siebel calendar entry to a repeating IBM Notes calendar entry

Table 2 describes the Siebel fields that CRM Desktop uses to create a repeating calendar entry.

Table 2. Siebel Fields That Siebel CRM Desktop Uses to Create a Repeating Calendar Entry

| Siebel Field | Description |
|------------------|--|
| ExceptionsList | Stores information about exceptions to instances in the repeating series. It is part of the Activity object. |
| RepeatingType | The frequency of the calendar entry. |
| RepeatingExpires | The date of occurrence of the last instance in the series. |
| Repeating | The flag that indicates a calendar entry is repeating. |

How Siebel CRM Desktop Handles a Single Instance of a Repeating Calendar Entry

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

How Siebel CRM Desktop Handles a Repeating IBM Notes Calendar Entry That Does Not Include an End Date

If a repeating IBM Notes Calendar entry that Siebel CRM Desktop shares with Siebel CRM does not include an end date, and if this repeating pattern:

- Matches a Siebel pattern, then it clears the value in the Repeat Until field of the Siebel CRM Calendar activity.
- Does not match a Siebel repeating pattern, then CRM Desktop limits this repeating pattern to a maximum duration. Table 3 describes the duration that CRM Desktop sets. It does this when it saves the Siebel CRM Calendar activity in IBM Notes.

Table 3. How Siebel CRM Desktop Handles a Repeating Calendar Entry That Does Not Include an End Date

| Repeating Pattern of the IBM Notes Calendar Entry | Maximum Duration of Occurrences That CRM Desktop Uses |
|---|---|
| Daily meetings | 1 year |
| Weekly meetings | 1 year |
| Monthly meetings | 2 years |
| Yearly meetings | 5 years |

How CRM Desktop Handles Invitee Lists for a Calendar Entry

The list of invitees in a calendar entry 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 calendar entry in IBM Notes, then the client attempts to resolve the meeting attendees and categorize the attendees as related contacts or employees when the user saves the calendar entry.

The user might not possess all the contact or employee data that CRM Desktop requires to parse all attendees, so CRM Desktop repeats this process during synchronization. If it makes an insert or update request for a calendar entry 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 related to the calendar entry.

How Siebel CRM Desktop Handles Invitee Lists 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 calendar entry in IBM Notes, 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 Invitee Lists for the Query Operation

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

- 1 It queries the calendar entry to return the invitee list in the To line of the email. 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 a calendar entry, then CRM Desktop modifies it in order to add the email addresses of the employees and contacts in the activity to the email To line. It uses a semicolon to separate each email address. It retains the employee list and the contact list.

How Siebel CRM Desktop Handles a Repeating Calendar Entry

A *repeating* calendar entry is a calendar entry that repeats at a specified interval. Siebel CRM Desktop supports synchronizing a repeating calendar entry between IBM Notes and the Siebel Server, but it handles a repeating pattern differently for the IBM Notes calendar entry than it does for a Siebel calendar activity.

Figure 8 illustrates how CRM Desktop handles a repeating calendar entry.

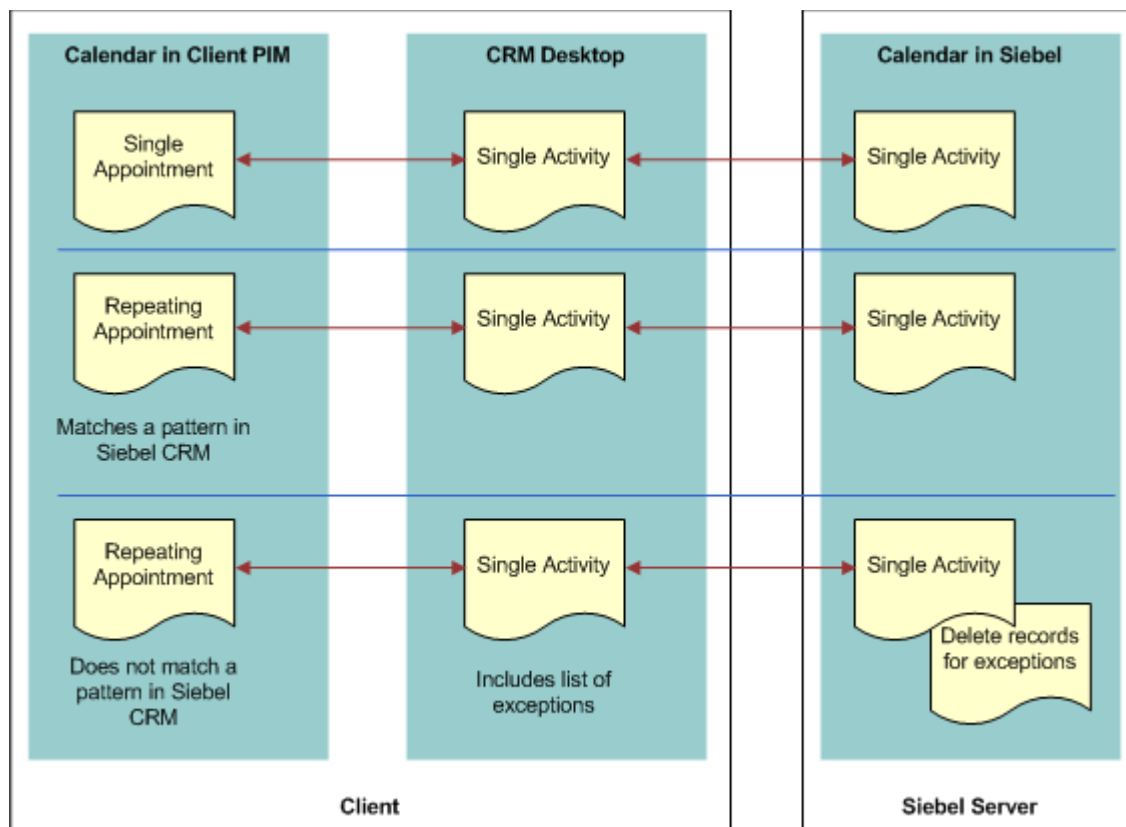


Figure 8. How Siebel CRM Desktop Handles a Repeating Calendar Entry

For more information, see ["How Siebel CRM Desktop Maps Fields Between Siebel CRM Data and IBM Notes Data"](#) on page 363.

How Siebel CRM Desktop Handles IBM Notes To Do items

This topic describes how Siebel CRM Desktop handles data for a native IBM Notes To Do item. For more information, see [“How Siebel CRM Desktop Maps Fields Between Siebel CRM Data and IBM Notes Data” on page 363](#).

If the user:

- Saves a new IBM Notes To Do item that is shared, then CRM Desktop creates a new Siebel CRM activity.
- Changes a native IBM Notes To Do item that is shared, then CRM Desktop changes the corresponding Siebel CRM activity.
- Deletes a native IBM Notes To Do item, and if this user:
 - Is the owner of the activity, then CRM Desktop deletes the corresponding Siebel CRM activity.
 - Is not the owner of the activity, then CRM Desktop removes the user from the employee team. It does not delete the corresponding Siebel CRM activity.

How Siebel CRM Desktop Handles IBM Notes Email Messages

Siebel CRM Desktop handles a IBM Notes email message in the following ways:

- Saves the email message as a Siebel CRM activity and sets the activity type depending on if the email sender:
 - Is an Employee. CRM Desktop sets the Siebel activity type to Email - Outbound. This situation occurs even if a user shares this email from the Inbox of another employee.
 - Is not an Employee. CRM Desktop sets the Siebel activity type to Email-Inbound.
- Sets the Display In value to Communications and Activities.
- Creates one Siebel CRM activity for the same IBM Notes email message that multiple recipients share with the Siebel Server.
- Allows the user to link the Siebel CRM activity to a Siebel CRM record. For more information, see [“How a User Can Link Siebel CRM Records to IBM Notes Records” on page 61](#).
- Depending on how the Saving email option is set on the Advanced tab of the Options dialog box, it does one of the following:
 - Adds only the email message or the email message attachments to the Siebel CRM activity.
 - Adds nothing to the Siebel CRM activity. For more information, see [“Controlling How Siebel CRM Desktop Handles Email Attachments” on page 109](#).

If the user deletes the source email message or moves it to a new folder, then 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 IBM Notes Data” on page 363](#).

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. Siebel CRM Desktop examines the recipients list when it receives an email message. If Siebel CRM Desktop finds an email address that matches one or more contacts with the Save correspondence option chosen, then CRM Desktop creates the corresponding Siebel CRM activity that is related with all contacts that CRM Desktop resolves from the email recipients. It resolves the primary contact first among the contacts that include a check mark in the Save Correspondence check box.

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 Siebel CRM Records to IBM Notes Records” on page 61](#).

How CRM Desktop Displays Data That Is Not Directly Visible

Siebel CRM Desktop synchronizes Siebel CRM account, activity, contact, and opportunity records to the client as complete or incomplete records. It displays incomplete records in the client as read only records. The user cannot edit the information of an incomplete record in a form that displays an incomplete record. The account, contact, and opportunity lists that CRM Desktop displays in an Explorer view include complete Siebel records and incomplete records that Online Lookup pins according to the synchronized records. For more information, see [“Controlling How Siebel CRM Desktop Handles Data That Is Not Directly Visible” on page 181](#).

Complete Records

A *complete record* is a record that matches synchronization filters. Siebel CRM Desktop synchronizes the entire record. The user can view all record information on the record form in the client and an administrator can view it on the Siebel Server. The lower corner of the Filter Records tab displays the number of complete Siebel records that CRM Desktop will synchronize. It does this when you create synchronization filters on the Filter Records tab of the Synchronization Control Panel.

Incomplete Records

An *incomplete record* is a record that does not match the synchronization filter but that Siebel CRM Desktop synchronizes anyway because it is associated with a complete record that does match the filter. An incomplete record is read-only in CRM Desktop regardless of the Siebel visibility that the user possesses for this record. The user cannot edit the record information in the client.

The user can view incomplete records in the association view or in lookup fields on record forms. For example, CRM Desktop displays incomplete:

- Contact records in the Contacts section of the Opportunity form
- Account records in the account lookup in the Account field on the Contact form

To avoid synchronizing the entire Siebel database to the client, CRM Desktop does not synchronize the associations for incomplete records to the client.

How Users Associate Complete and Incomplete Records

A user can associate complete or incomplete records in the client. Siebel CRM Desktop displays these records in SalesBook dialog boxes, such as in the Accounts SalesBook dialog box for the Account field on the Contact form. If the user enters text in a lookup field, then CRM Desktop searches complete and incomplete records and then displays the closest match.

Example of Using Complete and Incomplete Records

Assume the user sets up synchronization filters so that they synchronize the following items:

- All Siebel CRM contacts where the user is on the contact team
- All Opportunities that include a revenue of more \$10,000

Assume that Siebel CRM Desktop synchronizes Jackie Driver, a Siebel CRM contact record, to the client as a complete record because it matches the synchronization filter criteria. The user can modify all fields of this record on the contact form in the client. CRM Desktop also synchronizes all opportunities that are related to the contact, including the On-Road Assistance Package opportunity that includes \$5,000 in the Revenue field.

If the user opens the On-Road Assistance Package opportunity, then CRM Desktop displays this opportunity in a read-only form that displays the Revenue field. The user can view the revenue but not edit it. CRM Desktop synchronizes this opportunity as an incomplete record because it does not match the synchronization filters.

How a User Can Link Siebel CRM Records to IBM Notes Records

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 do the following work:

- Use an autocomplete list when the user types characters in a field.
- Use an autocomplete list 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 IBM Notes 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, 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

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

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

- **Shared calendar entry.** Removes each calendar entry that originates in Siebel CRM from the IBM Notes calendar. For the IBM Notes calendar entry, it removes any Siebel activities that are related to the IBM Notes calendar entry. The calendar entry no longer displays as shared and no contextual Siebel CRM data is related to the calendar entry.
- **Shared contact.** CRM Desktop cannot determine if a contact is IBM Notes data or Siebel CRM data, so it removes all shared contacts from the Personal Address Book (PAB) mailbox for the user in IBM Notes. It is recommended that you unshare every contact that the user must preserve before you remove the CRM Desktop add-in.
- **Shared email.** Does not remove an email message that CRM Desktop shares with Siebel CRM. It does remove Siebel activities that are related to a shared email so it no longer displays as shared in IBM Notes, and so that IBM Notes does not display any contextual data.
- **Shared To Do item.** Handles a To Do item in the same way that it handles a calendar entry. It removes each To Do item that originates in Siebel CRM from IBM Notes. It does not remove a native IBM Notes To Do item. IBM Notes does not display the To Do item as a shared To Do item and it does not display any Siebel CRM data that is related to the To Do item.

How CRM Desktop Handles Unshared Items 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 IBM Notes, 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 CRM Desktop. This situation occurs because CRM Desktop only deletes Siebel CRM data and extensions to IBM Notes that you deploy through CRM Desktop.

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 63](#)
- [How Siebel CRM Desktop Handles Synchronization Duplicates and Errors on page 73](#)

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 63](#)
- [How Siebel CRM Desktop Synchronizes Data During an Incremental Synchronization on page 64](#)
- [How Siebel CRM Desktop Synchronizes Siebel CRM Data on page 67](#)
- [How Siebel CRM Desktop Manages Synchronization Duration on page 68](#)
- [Situations Where Siebel CRM Desktop Reinstalls the Data Structure on page 68](#)
- [Factors That Determine the Data That Siebel CRM Desktop Synchronizes on page 70](#)

For more information about synchronization, see:

- [Overview of How Siebel CRM Desktop Synchronizes Data on page 24](#)
- [Chapter 8, "Controlling Synchronization"](#)

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 IBM Notes data storage with the Siebel CRM data that is available to the user. CRM Desktop downloads files in the customization package to IBM Notes the first time the user synchronizes metadata with the Siebel Server. This metadata includes the following information:

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

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 IBM Notes 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, 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 get the login ID of the user from the session, and then identifies the responsibility that the user uses, the customization package that the responsibility references, and if the package is active. For more information, see [“Relationships Between Users, Responsibilities, Customization Packages, and Metadata Files” on page 32](#).
- 4 If the customization package is published and valid, then CRM Desktop queries all metadata files of that package and creates a Siebel message. It 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 IBM Notes.

- 6 Downloads Siebel CRM data.

For more information, see [“How Siebel CRM Desktop Synchronizes Siebel CRM Data” on page 67](#).

- 7 Logs out of the synchronization session that it established in [Step 1](#).

For more information, see [“How Siebel CRM Desktop Handles Synchronization Errors” on page 74](#).

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 IBM Notes data storage to data in the Siebel database. It then does the following work:

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

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

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

CRM Desktop does the following work to complete an incremental synchronization:

- 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 IBM Notes to the Siebel Server. CRM Desktop cached these values during the prior synchronization. It does this to avoid expensive iterative operations through all responsibilities and customization packages every time it calls the Web service.
- 4 Receives a reply from the Siebel Server. This reply indicates if new metadata is available for the user.
- 5 If new metadata is available, then CRM Desktop 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 IBM Notes.
 - If the package is not changed, then it proceeds to [Step 11](#).
- 6 Determines if the currently applied package is compatible with the downloaded package and then does one of the following:
 - If the package is compatible, then CRM Desktop synchronizes the current data to the Siebel Server.
 - If the package is not compatible, then CRM Desktop stops the synchronization and the user changes are lost. The data modified in the old package is not appropriate for the current version of CRM Desktop.

For more information, see [“How Siebel CRM Desktop Determines Compatibility” on page 75](#).

- 7 Determines if the currently applied package is compatible with the current version of CRM Desktop. If the package is not compatible, then CRM Desktop does not apply the downloaded package, it displays a product incompatibility error message, and then exits this process.

For more information, see [“How Siebel CRM Desktop Determines Compatibility” on page 75](#).

- 8 If the package is compatible, then CRM Desktop determines if the object structure in the downloaded package changed.
- 9 If the object structure did not change, then it applies the new package and proceeds to [Step 11](#).
- 10 If the object structure changed, then it displays a dialog box that asks the user to do one of the following:
 - **Reinstall the object structure.** CRM Desktop does the following:

- ❑ Removes the old custom folders.
 - ❑ Removes the old data.
 - ❑ Installs the new package.
 - ❑ Displays the second part of the First Run Assistant. This part allows the user to set synchronization filters and make other settings that set up CRM Desktop to use the new customization package.
 - ❑ Starts a synchronization session after the user specifies these filters and other settings.
- **Do not reinstall the object structure.** CRM Desktop does not install the new customization package. The next time the user attempts to synchronize, it displays the same dialog box that asks the user to reinstall the object structure.
- 11** If the customization packages are identical, then CRM Desktop does the following work:
- a Sends a reply that indicates that it is not necessary to download the customization package because the package that resides on the Siebel Server is the same as the package that resides in IBM Notes.
 - b Exits this process.
- 12** Identifies the differences that exist between the data in IBM Notes and the data on the Siebel Server. To do this, it compares the change key values for all records that are available to the user in IBM Notes to the change key values that reside on the Siebel Server. The change key includes the record Id and the last time the server updated the record in the Siebel database. The value for the record Id resides in the ROW_ID column of the data table and the value for the time resides in the DB_LAST_UPD column of the data table. Depending on the differences, CRM Desktop changes the values in a data set to make sure the data between IBM Notes and the server is synchronized. For example, if CRM Desktop detects a new record during synchronization:
- On the Siebel Server, then it creates a corresponding record in IBM Notes.
 - In IBM Notes, then it creates a corresponding record on the Siebel Server.
- If the user changes synchronization filters, then CRM Desktop removes the Siebel CRM data that falls outside of the filters from IBM Notes. It does this during synchronization. A referenced record might remain in IBM Notes. For example, assume an account references a contact and this account does not match a filter. CRM Desktop continues to synchronize this account but makes it read-only in IBM Notes. It synchronizes the account details but it does not synchronize any account relationships that exist to other records.
- 13** Downloads Siebel CRM data.
- For more information, see [“How Siebel CRM Desktop Synchronizes Siebel CRM Data”](#) on page 67.
- The user can now view the newly downloaded data.
- 14** 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 allow the user to synchronize data. Before 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 CRM Desktop removes the current customization.
- 2 The user logs in with new credentials.
- 3 CRM Desktop downloads the package from the Siebel Server and then starts the First Run Assistant.

How Siebel CRM Desktop Synchronizes Siebel CRM Data

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

- 1 Calculates the number of records for each type of record, such as opportunities or accounts.
- 2 Gets the values of the change keys for all synchronization objects that are enabled, such as opportunities or accounts. For example, the record ID and the last updated date in the Siebel database.
- 3 Compares the set of IDs and timestamps in IBM Notes to the set of IDs and timestamps on the Siebel Server to do the following:
 - Identify 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 conflicts.
- 4 For each difference, CRM Desktop does one of the following operations in IBM Notes 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 IBM Notes.
 - **Siebel update.** Query the Siebel database to get the details of the updated record and then update the appropriate item in IBM Notes. Note that a Siebel update overwrites all fields in the corresponding IBM Notes item, not just the updated fields.
 - **Siebel delete.** Delete the appropriate item in IBM Notes.
 - **IBM Notes 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.
 - **IBM Notes 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 that this handling is different than in the situation where CRM Desktop changes the same record in the Siebel database or when it compares IDs and timestamps. In this situation, CRM Desktop makes the change in the Siebel database during the actual update operation.
- IBM Notes **delete**. Delete the appropriate record in the Siebel database.
- 5 If a conflict occurs, then CRM Desktop does the following work:
- a Updates the synchronization issues and conflicts log on the client.
 - b Prompts the user to choose the changes to keep in each of the following situations:
 - Update the record in IBM Notes 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 on. To shorten this duration, you or the user can do the following:

- You can modify the application configuration. For more information, see [“Controlling Synchronization” on page 127](#).
- The user can adjust settings through the synchronization filter dialog box. For more information, see [“How Filters Reduce the Data That Siebel CRM Desktop Synchronizes” on page 71](#).

The duration of an incremental synchronization session is typically shorter than for an initial synchronization because CRM Desktop downloads all objects during an initial synchronization but during an incremental synchronization it only downloads the objects that changed since the last synchronization.

Situations Where Siebel CRM Desktop Reinstalls the Data Structure

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

- 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 IBM Notes, then CRM Desktop does the following:
 - Informs the user that a problem with the data structure exists.
 - Removes the data structure.
 - Installs a new data structure.

If 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 every shared calendar entry and To Do item that originates in Siebel CRM. Each shared calendar entry and To Do item that originates in IBM Notes remain in IBM Notes.
- 3 Removes the custom data structure that it previously deployed to IBM Notes data storage. For example, to remove all custom folders in the user mailbox.
- 4 Installs the new data structure.

To reenter the appropriate Siebel CRM data in the IBM Notes data storage, the user must manually start a new, initial synchronization session.

The Customization Package Changed

During synchronization, Siebel CRM Desktop determines if the customization package for the user who is currently logged in 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 changed, and if CRM Desktop must reinstall the data structure, then it displays a prompt that is similar to the following:

A new configuration is available. Are you ready to download and apply it? Selecting "Yes" will remove your current data, re-install the data structure, and download the data again.

The Customization Package Changed But the Data Structure Has Not Changed

If Siebel CRM Desktop determines during synchronization 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 it downloads and installs the new package and informs this user about this download. A modification to a security rule is an example of where the package changed but the data structure has not changed. In this situation, CRM Desktop does not start a new, initial synchronization.

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

Siebel CRM Desktop prevents data loss if the user deletes customization package files differently depending on if IBM Notes is open:

- **IBM Notes is open.** 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 in Windows 7, then Windows Explorer does not allow the deletion:

C: \Users\user\AppData\Roaming\Oracle\CRM Desktop for IBM Notes\Profile\Data

- **IBM Notes is not open.** The user can use Windows Explorer to delete customization package files. However, if the user subsequently starts IBM Notes, then CRM Desktop restores the customization package files from local storage. For more information about this local storage, see [“How Siebel CRM Desktop Stores Siebel CRM Data” on page 25](#).

How Connectivity Failures Affect Synchronization

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 CRM Desktop ran successfully before the interruption.

Factors That Determine the Data That Siebel CRM Desktop Synchronizes

A Siebel user can typically access only a subset of data that is available in the Siebel database. This topic describes the factors that determine the data that a user can access. How you configure CRM Desktop determines many aspects of the data that it 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 specify this configuration before you deploy CRM Desktop to your users. The user can choose presets for a predefined filter and specify personal filters in the First Run Assistant. The internal filters and server application metadata configuration restricts access to some data, and the user filters apply a second layer of filtering. CRM Desktop applies these filters during initial synchronization and incremental synchronization.

How Filters Reduce the Data That Siebel CRM Desktop Synchronizes

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

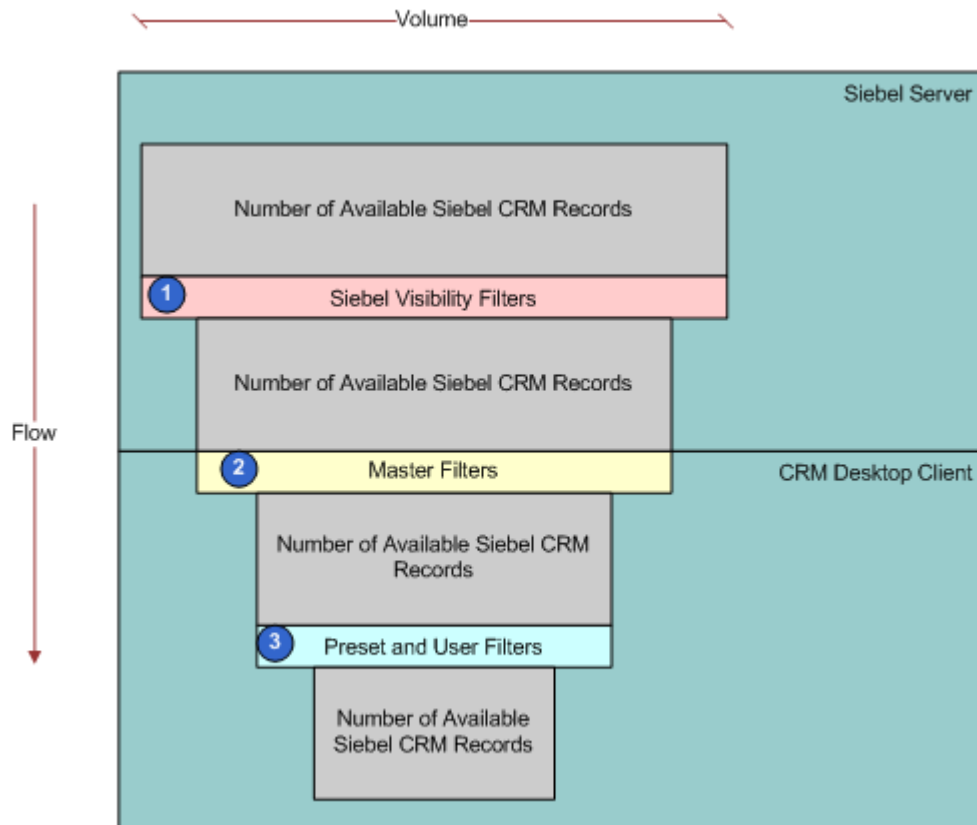


Figure 9. How Filters Reduce the Data That Siebel CRM Desktop Synchronizes

Explanation of Callouts

The following filters reduce the data that CRM Desktop synchronizes:

- 1 Siebel visibility filters.** Visibility rules that are configured in the Siebel Repository and that the Siebel Server applies affects data access. CRM Desktop integrates with the Siebel Server through the Web service interface, so security, search specifications, and other logic that is configured at the integration or business object layer limits the data that CRM Desktop synchronizes to the client. The user interface configuration does not affect the results of queries or other operations that CRM Desktop performs.
- 2 Master filters.** Internal synchronization filters that an administrator sets. They identify the Siebel CRM data that CRM Desktop synchronizes to the client. 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 the data that CRM Desktop downloads to the client.

- 3 Preset and user filters.** An administrator can create preset filters in the customization package and then specify the filter that CRM Desktop applies as the default filter. The user can use this default filter or choose another preset filter. The user can do this in the First Run Assistant during installation or later in the Filter Records Tab of the Synchronization Control Panel. To create a preset filter, the user can modify an existing preset filter. The user can apply different saved presets at different times, depending on the filter requirement. CRM Desktop uses these filters and the application configuration to identify the data to synchronize.

Depending on relationships in the data, CRM Desktop might synchronize an object that the Filter Records Tab disables for synchronization. For example, if the opportunity object is enabled but the account object is not enabled, then it still downloads any account data that the opportunity references. This download is required to make sure the data is complete. Also, 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 IBM Notes, then it uploads the account to the Siebel Server. For more information, see the following topics:

- [Customizing How the First Run Assistant Performs the Initial Synchronization on page 93](#)
- [Controlling Synchronization Filters on page 127](#)

Objects That Are Enabled for Synchronization

A set of objects that are enabled for synchronization determines the data that CRM Desktop can synchronize, depending on the configuration that CRM Desktop downloads for the user. These objects are defined in the application metadata that you deploy through the customization package that is available to the user. If the application metadata does not define an object, then CRM Desktop does not synchronize it. Application metadata also defines the field mappings that CRM Desktop uses in the synchronization. These mappings specify how CRM Desktop synchronizes objects in IBM Notes and on the Siebel Server. For more information, see [“Customizing Field Mapping” on page 154](#).

How Differences Between IBM Notes and the Siebel Server Affect Synchronization

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

- Data is created, updated, or deleted in IBM Notes.
- Data is created, updated, or deleted on the Siebel Server.

For more information, see [“How Siebel CRM Desktop Synchronizes Data During an Incremental Synchronization” on page 64](#).

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 Siebel CRM Desktop Handles Synchronization Duplicates and Errors

This topic describes how Siebel CRM Desktop handles synchronization duplicates and errors. It includes the following topics:

- [How Siebel CRM Desktop Avoids Duplicate Data on page 73](#)
- [How Siebel CRM Desktop Handles Synchronization Errors on page 74](#)

How Siebel CRM Desktop Avoids Duplicate Data

Siebel CRM Desktop includes metadata in 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 that reside in the Siebel database.
Data deduplication that you can deploy to prevent duplicate data. For more information, see [Resolving Synchronization Conflicts on page 148](#).
- The view mode that CRM Desktop uses for duplicate requests during synchronization for an object type. For more information, see ["Controlling the View Mode During Synchronization According to Object Type" on page 143](#).

CRM Desktop uses Siebel integration objects to create the data structures that are available to synchronize with IBM Notes. These objects support the user key definition that is the first additional layer of duplicate prevention. For information about how to configure user keys for integration objects and how the EAI Siebel Adapter uses them, see *Overview: Siebel Enterprise Application Integration*.

CRM Desktop also supports user key configuration in the metadata for the client. If it detects the IBM Notes insert during synchronization, then it queries the synchronization object in the Siebel database with the user key to determine if any records exist that match the record that it is inserting. If it:

- **Does not find a match.** It proceeds with the insert operation.
- **Finds a match.** It raises a synchronization issue that prevents the insert. For more information, see ["Resolving Synchronization Conflicts" on page 148](#).

How Siebel CRM Desktop Handles Synchronization Errors

If a top-level error occurs during synchronization, then the Synchronization Engine stops any further processing and displays a message to the user that describes the error. The following types of errors can occur:

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

If an operation failure occurs in the Synchronization Engine, then CRM Desktop creates a synchronization issue and then attempts to do 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 for an object changed since CRM Desktop queried this object during the current synchronization session, then it cannot do the update and the delete operations until the next synchronization cycle. In this situation, it creates an issue and then handles this issue in the subsequent synchronization. This synchronization creates a collision because the object changed since the last synchronization. A *collision* is a data integrity problem that occurs if CRM Desktop modifies the same record on the Siebel Server and in IBM Notes between synchronization sessions. For more information, see [“Resolving Synchronization Conflicts” on page 148](#).

CRM Desktop logs synchronization errors in synchronization log files and in the General Log log file. It does not enable log files, by default. For more information, see [“Log Files You Can Use with Siebel CRM Desktop” on page 116](#).

How Siebel CRM Desktop Handles Errors While Downloading the Customization Package

If an error occurs while CRM Desktop downloads the customization package, then it displays an error message near the taskbar. This error notifies the user that the customization package changed but CRM Desktop cannot download it because of errors. The problem might be due to the fact that the user does not possess the privilege that the Siebel Server requires to download the package. In this situation, the user must contact the system administrator to get the necessary privileges and then get the customization package again.

How Siebel CRM Desktop Determines Compatibility

This topic describes how Siebel CRM Desktop determines compatibility. For a description of the work CRM Desktop does depending on compatibility, see [“How Siebel CRM Desktop Synchronizes Data During an Incremental Synchronization” on page 64](#).

How Siebel CRM Desktop Determines Product and Version Compatibility

Siebel CRM Desktop uses the following element in the info.xml file to determine product and version compatibility:

```
<compatibility>
  <products>versions</products>
  <schemas>versions</schemas>
</compatibility>
```

where:

- **products.** Specifies product versions that are compatible with the package that CRM Desktop must install.
- **schemas.** Specifies package versions that are compatible with the package that CRM Desktop must install. If the current package is compatible with the new package that it must install, then CRM Desktop synchronizes the local data to the Siebel Server before it applies the new package.
- **versions** is a string that includes one or more version numbers. A dash (-) specifies a range of versions. A semi-colon (;) separates individual version numbers.

For example, the following code specifies that all product versions starting with version 3.05.15.00 through version 3.05.30.99 are compatible:

```
<compatibility preferred_product="3.05.30.00">
  <products>3.05.15.00-3.05.30.99</products>
  <schemas>3.04.00.00-3.05.30.99</schemas>
</compatibility>
```

CRM Desktop returns a preferred version in the following situations:

- The product is not compatible.
- The product is compatible but the preferred version is not the same version as the current product version.

How Siebel CRM Desktop Determines Schema Compatibility

The schema subelement of the compatibility element in the info.xml file determines schema compatibility. If Siebel CRM Desktop can save the data that it creates or modifies in the old customization package, then the schema is compatible. It saves this data to the current version of the Siebel database. An example of schema incompatibility occurs if a required field in Siebel CRM does not contain a value because the old package does not require it.

How Siebel CRM Desktop Determines Object Structure Compatibility

Siebel CRM Desktop examines the object structure to determine compatibility. For example, if you create a new object type that Siebel CRM Desktop synchronizes, then it must reinstall the current folder structure and install the new folder structure with the new package. If the object structure that the new customization package defines is not different from the object structure that the old package defines, then CRM Desktop applies changes from the new customization package.

How Siebel CRM Desktop Handles Incompatible Customization Packages

If the current version of Siebel CRM Desktop is not compatible with the downloaded customization package, then CRM Desktop does not apply the package. Instead, it displays an error message that notifies the user and then adds an entry in the CRMDesktop n .log file, where n is an incremental number that uniquely identifies the log file name. It stores the error message in the most recent log file.

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 CRM Desktop Add-In on page 81](#)
- [Process of Installing the CRM Desktop Add-In on page 84](#)
- [Options for Installing the CRM Desktop Add-In on page 88](#)
- [Troubleshooting Siebel CRM Desktop Installation on page 101](#)

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 Installing the CRM Desktop Add-In on page 84](#)

For information about ACR installation instructions, see *Siebel Maintenance Release Guide* on My Oracle Support.

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 78](#)
- 4 [Administering Server Variables on page 80](#)

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 CRM Desktop supports the environment where you implement CRM Desktop.

You must verify that the supported environments are installed before you install Siebel CRM Desktop. 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 predefined metadata files for Siebel CRM Desktop version 3.2 and later. The client uses these files to determine the data to synchronize and the validation rules to apply.

To administer metadata files

- 1 Create a .zip file that includes the following files:
 - Every file listed in [“Metadata File Types” on page 348](#).
 - Every language file that your implementation requires. If Siebel CRM Desktop must support a language other than English, then you must add the required language file. For example, the Ln_package_res.jp_JP.xml file. For more information, see [“Metadata File Types That Support Languages” on page 354](#).

Make sure every file in this .zip file is in the root directory in the zip file. This zip file must not contain a subdirectory.
- 2 With administrator privileges, log in to Siebel Sales Enterprise through a Siebel Web Client that is connected to the Siebel Server.
- 3 Navigate to the Administration - CRM Desktop screen and then the Metadata Files view.
- 4 In the Metadata Files list, click New.
- 5 In the Type list, choose IBM Notes Package.
- 6 In the File Name field, locate the file you created in [Step 1](#).
- 7 Click Menu and then click Save Record.

Creating and Publishing the Customization Package

This task is a step in [“Process of Preparing the Siebel Server” on page 77](#).

You create a relationship between a responsibility and a customization package that determines the information that is available to the user. You can publish a package when CRM Desktop finishes the updates and it is ready to download this package to the client. Publishing makes a package *read only* so that you cannot make any more modifications on the package. For more information, see [“Relationships Between Users, Responsibilities, Customization Packages, and Metadata Files” on page 32](#).

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 use a name that describes the purpose of the package configuration. For example, EMEA Sales Rep, or Field Sales Rep. |
| Responsibility | Choose the responsibility that is appropriate for the group of users that CRM Desktop uses with the package. Do not assign a user to more than one package. It is recommended that you maintain a separate set of CRM Desktop responsibilities where you can control the user assignment. This configuration helps to prevent creating relationships between a user and more than one responsibility and more than one package. If necessary, before you do this step, you can create a new responsibility and then assign specific users to this responsibility. For more information, see “Guidelines for Assigning Responsibilities to Customization Packages” on page 79. |

- 3 In the Metadata Files list, click Add, locate the .zip file that you added in [Step 1 on page 78](#), and then click OK.
- 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.

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

For more information about how CRM Desktop uses a customization package, see [“Relationships Between Users, Responsibilities, Customization Packages, and Metadata Files”](#) on page 32.

Guidelines for Assigning Responsibilities to Customization Packages

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

- You must assign only 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 create a relationship between only one of these responsibilities with an active customization package.
- Make sure the responsibility and customization package that Siebel CRM Desktop assigns to the user is unique. For example, if CRM Desktop assigns two different responsibilities and two different customization packages to the same user, then a conflict might occur and your customizations might fail.

Republishing Customization Packages

If you change a metadata file, then you must republish the customization package that references this 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 then publish the old package.

To republish a customization package

- 1 Unpublish the old customization package:
 - a With administrator privileges, log in to a Siebel CRM 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 republish.
 - d In the Packages list, click the link in the Package Name field.
 - e In the Package Details form, click Unpublish.
 - f Make sure Siebel CRM 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 78](#).

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 CRM client that is connected to the Siebel Server.
 - b Navigate to the Administration - Server Configuration screen and then the Servers view.
 - c In the Components list, query the Component field for EAI Object Manager.
 - d In the last 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 set to the following:

-1

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

For more information, see [“Controlling How Siebel CRM Desktop Assigns Calendar Entry Owners” on page 108](#).

4 Stop and then restart the Siebel Server.

Overview of Installing the CRM Desktop Add-In

This topic describes an overview of installing the CRM Desktop add-in. It includes the following topics:

■ [Changes That Siebel CRM Desktop Makes During Installation on page 82](#)

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

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

You can deploy 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 on. System Center Configuration Manager (SCCM) from Microsoft is an example of deployment software. To deploy the 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 the list of users, computers, workgroups or domains where you must distribute the software.

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

CRM Desktop displays the First Run Assistant after you complete the installation and the user starts IBM Notes. For more information, see [“Customizing the First Run Assistant” on page 89](#).

For more information about using Systems Management Server, see the documentation at the Microsoft TechNet web site.

Changes That Siebel CRM Desktop Makes During Installation

This topic describes changes that Siebel CRM Desktop makes during installation. It makes these changes to the file system, Windows Registry, and settings in IBM Notes.

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 Desktopfor IBM Notes\Profile\`

where:

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

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

`\Documents and Settings\username\Application Data\Oracle\CRM Desktopfor IBM Notes\Profile`

You can change this directory. For more information, see [“Setting the Installation Directory of the CRM Desktop Add-In” on page 100](#).

Table 4 describes where CRM Desktop stores data in the file system when the client runs on Windows XP.

Table 4. Example of Where Siebel CRM Desktop Stores Data in the File System

| Windows XP Folder on Client | Description |
|---|--|
| \Documents and Settings\username\Application Data\Oracle\CRM Desktop for IBM Notes\bin | CRM Desktop saves the following information: <ul style="list-style-type: none"> ■ Add-in dll files. ■ Resources that binary files use. ■ Microsoft Visual Studio run-time libraries. ■ Help files. |
| \Documents and Settings\username\Application Data\Oracle\CRM Desktop for IBM Notes\Profile | CRM Desktop saves the following information: <ul style="list-style-type: none"> ■ The Data folder. This folder includes package files. ■ CRM Desktop log files. ■ Various database files. |
| \Documents and Settings\username\Application Data\Oracle\CRM Desktop for IBM Notes\Profile\Data | XML, DXL files, and JavaScript files of the customization package. For more information, see “Files in the Customization Package” on page 355 . |
| \Documents and Settings\username\Application Data\Oracle\CRM Desktop for IBM Notes\Profile\Logs | For more information, see “Log Files You Can Use with Siebel CRM Desktop” on page 116 . |
| Temp\ | When you download the customization package CRM Desktop places some files in a temporary directory. |

Changes That Siebel CRM Desktop Makes in the Windows Registry

CRM Desktop adds registry entries differently depending on one of the following options that you choose when you install the CRM Desktop add-in:

- **Anyone Who Uses This Computer.** CRM Desktop adds registry entries in the following registry keys:
 - HKEY_LOCAL_MACHINE\Software\Oracle\CRM Desktop
 - HKEY_CURRENT_USER\Software\Oracle\CRM Desktop for the user who is currently logged in
- **Only For Me.** CRM Desktop adds registry entries only in the following registry key
 - HKEY_CURRENT_USER\Software\Oracle\CRM Desktop

For more information, see [“Installing the CRM Desktop Add-In” on page 87](#).

These settings include the following information:

- General settings for CRM Desktop, such as login information.
- In the Logging subkey, logging settings that CRM Desktop uses to tune logging behavior.

CRM Desktop registers COM classes in the Windows Registry when you install CRM Desktop. 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 [“Using the Windows Registry to Control Siebel CRM Desktop” on page 103](#).

Changes That Siebel CRM Desktop Makes to Settings in IBM Notes

Siebel CRM Desktop adds the following items to the IBM Notes Personal Address Book (PAB) file, which is the local names.nsf file:

- Custom folders
- Custom views
- Custom objects
- Custom forms
- Custom Script libraries
- Custom agents
- Custom database script
- Custom image resources

CRM Desktop patches some of the existing views in the Personal Address Book and native forms in the Mail Database. It does this regardless if the database resides locally or on the Domino Server.

CRM Desktop runs as a IBM Notes add-in, so it must register with IBM Notes. For more information about registering a IBM Notes add-in, see the technical documentation at the IBM Notes web site at wwwsupport.ibm.com.

For more information, see [“How Siebel CRM Desktop Stores Siebel CRM Data” on page 25](#).

Process of Installing the CRM Desktop Add-In

This process is a step in [“Roadmap for Installing Siebel CRM Desktop” on page 77](#). You do the following work to install the CRM Desktop add-in:

- 1 [Preparing Your Environment for Installation on page 85](#)
- 2 [Installing the CRM Desktop Add-In on page 87](#)

For more information, see [“Options for Installing the CRM Desktop Add-In” on page 88](#).

Preparing Your Environment for Installation

This task is a step in [“Process of Installing the CRM Desktop Add-In” on page 84](#).

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

To prepare your environment for installation

- 1 Choose your deployment software and review the conditions that apply for the installation.
For more information, see [“Overview of Installing the CRM Desktop Add-In” on page 81](#).
- 2 Configure access permissions so that you can successfully install CRM Desktop.
For more information, see [“Configuring Access Permissions So That You Can Successfully Install CRM Desktop” on page 86](#).
- 3 Make sure a direct connection to the Siebel Server is available.
CRM Desktop uses information from one of the following sources to connect to EAI (Enterprise Application Integration):
 - **Parameters during installation.** This configuration is appropriate if you install CRM Desktop in the background. For more information, see [“Setting the URL for the Siebel Server” on page 100](#).
 - **Connection settings dialog box.** This configuration is appropriate if you install CRM Desktop manually. For more information, see [“Customizing How First Run Assistant Uses the Customization Package” on page 90](#).
- 4 Make sure the EAI object manager on the Siebel Server is enabled and online.
- 5 Make sure only a single Position is defined for the user account.
The user cannot use IBM Notes to change the position. It is recommended that you use only a single Position for a given user account.
- 6 Make sure the customization package for the user position is published for only one of the user responsibilities.
- 7 Verify that the email address you use in the IBM Notes account in the location where you install CRM Desktop is the same as the email address for this employee record on the Siebel Server.
- 8 Make sure you uploaded and published the customization package on the Siebel Server.
For more information, see [“About the Customization Package” on page 33](#).
- 9 Make sure the number of records for each type of Siebel object, such as accounts, is limited to an amount that the local Personal Address Book and Mail database can accommodate.
This amount depends on the following items:
 - Size of the IBM Notes folder
 - The IBM Notes version
 - Where CRM Desktop stores the databases

- Connectivity
- Hard disk space on the client and on the computer
- Capabilities of the client computer
- And so on

It is recommended that you test these amounts in a test environment before you deploy CRM Desktop to all users. For more information, see [“Controlling the Number of Records That Synchronize” on page 137](#).

Configuring Access Permissions So That You Can Successfully Install CRM Desktop

IBM Notes uses access permissions that allow the user to interact with the following items:

- **Data.** For example, email messages and contact records.
- **Elements.** For example, a field in the Email form or the Contact form. IBM Notes uses these elements to display data. You can use IBM Domino Designer to configure them.

You can configure IBM Notes to store the Mail database in the following locations. These locations can determine the access level that the user possesses to data and elements:

- **Client.** This Mail database might use POP3 or SMTP, so the user possesses administrator privileges to it.
- **Domino Server.** You can limit access to the Mail database according to the following roles:
 - **Administrator.** User can interact with all elements. It is the highest level of user access.
 - **Designer.** User can modify elements.
 - **Contributor.** User can modify data. User can read but not modify elements. For example, the user can view and delete email data but cannot modify the Email form.
 - **Editor.** The user can edit some data entries but cannot delete them. The user can only view elements.
 - **Reader.** The user can only view data from the Mail database.

It is important that you configure your environment so the user can access the data and elements that they must interact with. For example, assume the following occurs:

- You configure CRM Desktop to store the Mail database on the Domino Server.
- You configure CRM Desktop to include a new, custom field on the Email form.
- A user who possesses the Reader, Contributor, or Editor access permission installs CRM Desktop. In this situation, the user cannot add the new, custom field to the Email form. If this user attempts to connect to the Mail database, then CRM Desktop displays an IBM Notes security error message.

Configuring Siebel CRM Desktop To Store Data and Elements

You must not configure CRM Desktop to store all elements only in the Personal Address Book or only in the Mail database. For example, CRM Desktop stores the Account form and the Opportunity form in the database that the Personal Address Book uses, and it stores the email form that displays email messages in the Mail database. It also stores email messages in the Mail database. You can configure CRM Desktop to store these items in various locations. For more information, see [“How Siebel CRM Desktop Stores Siebel CRM Data” on page 25](#).

Installing the CRM Desktop Add-In

This topic describes how to manually install the CRM Desktop add-in.

To install the CRM Desktop add-in

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

The CRMDesktopfor IBM Notes.msi installation package validates the operating system version and the IBM Notes 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 IBM Notes is installed on the client computer and configured for use.

If it is not, then an error occurs and CRM Desktop ends the installation.
- 3 Make sure you possess rights on the client computer so that you can run the executable file that CRM Desktop provides in the installation package.
- 4 Manually copy the CRMDesktopfor IBM Notes.msi file to the client computer.

To use third-party deployment software to deploy the CRMDesktopfor IBM Notes.msi file to multiple users, see [“Installing Siebel CRM Desktop in the Background” on page 97](#).
- 5 Locate the CRMDesktopfor IBM Notes.msi installation package on the client computer.

The following directory is a typical location:

C: \Documents And Settings\username\Desktop
- 6 Run the CRMDesktopfor IBM Notes.msi installer.
- 7 In the Welcome dialog box, click Next.
- 8 In the Customer Information dialog box, enter the user name and the organization.
- 9 Choose to install the add-in for one of the following, and then click Next:
 - **Anyone Who Uses This Computer.** Any user who logs on to this computer can use the CRM Desktop add-in.
 - **Only For Me.** Only the user who is logged on to the computer when CRM Desktop installs the CRM Desktop add-in can use this add-in.

- 10** In the Destination Folder dialog box, specify the folder where the installer must install CRM Desktop.

You can specify any directory. For more information, see [“Setting the Installation Directory of the CRM Desktop Add-In” on page 100](#).

- 11** In the Ready to Install the Program dialog box, click Install.

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

c:\Documents and Settings\username1\Appl i cati on Data\Oracl e

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

c:\Documents and Settings\username2\Appl i cati on Data\Oracl e

How Siebel CRM Desktop Installs the Siebel CRM Desktop Location

Siebel CRM Desktop displays a dialog box that allows the user to apply the CRM Desktop configuration to this IBM Notes location. It does this when IBM Notes runs for first time after you install the CRM Desktop add-in. The user can choose one of the following values:

- **Yes.** CRM Desktop applies the configuration and then displays the First Run Assistant.
- **No.** The CRM Desktop add-in closes. The dialog box that allows the user to apply the CRM Desktop configuration displays each time the user starts IBM Notes until the user chooses to apply this configuration.

The following choice that you make in [Step 9 on page 87](#) determines if CRM Desktop applies this behavior:

- **Anyone Who Uses This Computer.** This behavior applies to any user who logs on to this computer.
- **Only For Me.** This behavior applies only to the user who is logged on to the computer when CRM Desktop installs the CRM Desktop add-in.

Options for Installing the CRM Desktop Add-In

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

- [Customizing the First Run Assistant on page 89](#)
- [Installing Siebel CRM Desktop in the Background on page 97](#)
- [Using the Windows Command Line to Set Optional Parameters on page 98](#)

Customizing the First Run Assistant

This topic describes how to customize the First Run Assistant. It includes the following topics:

- [Customizing How First Run Assistant Uses the Customization Package on page 90](#)
- [Customizing How Siebel CRM Desktop Connects to the Internet on page 91](#)
- [Changing Behavior of the CRM Desktop-Login Dialog Box on page 92](#)
- [Customizing How the First Run Assistant Performs the Initial Synchronization on page 93](#)
- [Customizing How Siebel CRM Desktop Shares Native IBM Notes Items on page 94](#)
- [Suppressing the Dialog Boxes That First Run Assistant Displays on page 95](#)

The *First Run Assistant* is a wizard that guides the user through the first setup of the CRM Desktop add-in. CRM Desktop displays the CRM Desktop icon in the system tray and starts the First Run Assistant. It does this the first time the user starts IBM Notes after you install the CRM Desktop add-in. The user can begin using IBM Notes after the user finishes using this assistant.

The First Run Assistant displays a dialog box at each step that allows the user to specify 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 24](#).

Customizing How First Run Assistant Uses the Customization Package

[Table 5](#) describes how you can customize the First Run Assistant to register and get the customization package. It lists work items in the order that the user performs them while the user uses this assistant. The user must install the CRM Desktop add-in first and then use the assistant. For more information, see [“Installing the CRM Desktop Add-In” on page 87](#).

Table 5. How First Run Assistant Registers and Gets the Customization Package

| Step | Description | Possible Customization |
|------|--|--|
| 1 | The user opens IBM Notes the first time after the CRM Desktop add-in is installed. It is the first time that IBM Notes is open after the add-in is installed, so First Run Assistant displays the welcome screen and then the user clicks it. | Not applicable |
| 2 | <p>CRM Desktop chooses the Use Internet Explorer Settings for Proxy-Server option, by default.</p> <p>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:</p> <ul style="list-style-type: none"> ■ 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. <p>The proxy server requires a separate host name and a port number.</p> | For more information, see “Customizing How Siebel CRM Desktop Connects to the Internet” on page 91 . |

Table 5. How First Run Assistant Registers and Gets the Customization Package

| Step | Description | Possible Customization |
|------|--|---|
| 3 | <p>First Run Assistant displays the CRM Desktop-Login dialog box. The user enters the user name and password.</p> <p>This user name must include the First Name and Last Name or the User ID of the user record that resides in the Siebel database. The user can enter the First and Last name in any order.</p> <p>The USERID is the same user ID that the user uses for the Siebel Web Client. For example, Wasaka Takuda, or WTAKUDA.</p> <p>The password is the same password as the password that the user uses for the Siebel Web Client.</p> | <p>For more information, see the following topics:</p> <ul style="list-style-type: none"> ■ Changing Behavior of the CRM Desktop-Login Dialog Box on page 92 ■ Using the Windows Registry to Control Siebel CRM Desktop on page 103 |
| 4 | <p>First Run Assistant automatically enters the URL that the Siebel Business Application uses to connect to the Siebel Server. It enters this URL in the Server URL window. For example:</p> <p><code>http://server_name/eai/enu</code></p> | <p>You can specify the URL. For more information, see "Setting the URL for the Siebel Server" on page 100.</p> |

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.
For more information, see ["Files in the Customization Package" on page 355](#).
- 2 Locate the platform section.
- 3 Add the initialization_script section to the section that you located in [Step 2](#):

```
<platform>
  <initialization_script>
    <![CDATA[
      application.settings.set("ProxyUsage", value);
    ]]>
  </initialization_script>
</platform>
```

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 CRM Desktop-Login Dialog Box

You can change the behavior of the CRM Desktop-Login dialog box. For information about authentication options, see [Chapter 12, "Customizing Authentication."](#)

To change behavior of the CRM Desktop-Login dialog box

- Hide the Save Password check box that Siebel CRM Desktop displays in the CRM Desktop-Login dialog box. You set the following Windows Registry key to 1:

Siebel : HideSavePasswordOption

If the user clicks Save Password in the CRM Desktop-Login dialog box, then 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 CRM Desktop. For more information, see ["Using the Windows Registry to Control Siebel CRM Desktop" on page 103.](#)

- Prevent CRM Desktop from displaying the CRM Desktop-Login dialog box. You do the following:
 - a Set the following Windows Registry key to 1:

SuppressLoginDialog

- b Set the save_password parameter and the Login externally.

If you do not set the save_password parameter, then CRM Desktop requires the user to enter the password every time the user opens IBM Notes and then synchronizes.

For more information, see ["How Siebel CRM Desktop Suppresses the Desktop-Login Dialog Box" on page 92.](#)

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 exist in the Windows Registry, and if save_password exists in the Windows Registry and is set to 1, then CRM Desktop attempts to validate the user credentials on the Siebel Server.

- If the Siebel Server returns an error for this login, then 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 does not exist in the Windows Registry, or if it is set to 0, then the Siebel Server returns a Credentials Verification Failed error.

Customizing How the First Run Assistant Performs the Initial Synchronization

Siebel CRM Desktop applies custom design elements, as described in [Table 5 on page 90](#), and then displays the second part of the First Run Assistant. It prompts the user to set preferences and to run the first synchronization session that downloads Siebel CRM records to IBM Notes. [Table 6](#) describes the work that you can do to customize how the assistant does this initial synchronization. It lists work items in the order that the user does them while the user runs the assistant.

Table 6. How First Run Assistant Performs the Initial Synchronization

| Step | Description | Administrative Work |
|------|--|---|
| 1 | <p>First Run Assistant applies custom design elements. It then displays the following choices in the Filter Records tab of the Synchronization Control Panel dialog box:</p> <ul style="list-style-type: none"> ■ Leave the filters at their default settings. ■ Choose a filter from the predefined filter that CRM Desktop deploys with the CRM Desktop add-in. ■ Specify filter settings. <p>The user can also specify the synchronization frequency and other settings that CRM Desktop uses.</p> | <p>For more information, see the following topics:</p> <ul style="list-style-type: none"> ■ Controlling the Object Types That Siebel CRM Desktop Displays in the Filter Records Tab on page 127 ■ Controlling the Size and Type of Synchronized Records on page 135 |
| 2 | <p>The First Run Assistant displays a dialog box that allows the user to configure synchronization settings. CRM Desktop does the following, by default:</p> <ul style="list-style-type: none"> ■ 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 | <p>For more information, see “Controlling the Synchronization Intervals That Display in the Synchronization Tab” on page 132.</p> |

Table 6. How First Run Assistant Performs the Initial Synchronization

| Step | Description | Administrative Work |
|------|---|--|
| 3 | The First Run Assistant displays a dialog box that allows the user to share with CRM Desktop each new native IBM Notes calendar entry, contact, or To Do item that the user creates in IBM Notes. CRM Desktop includes a check mark in the Calendar Entry, Contacts, and To Do items check boxes, by default. | For more information, see “Customizing How Siebel CRM Desktop Shares Native IBM Notes Items” on page 94 |
| 4 | The First Run Assistant displays the Siebel CRM Desktop dialog box. For more information, see “Sharing a Calendar Entry, Contact, or To Do Item” on page 95 . | For more information, see “Controlling How Siebel CRM Desktop Assigns Calendar Entry Owners” on page 108 . |

The user finishes specifying the configuration settings, and then CRM Desktop automatically starts the synchronization and adds content to the Siebel CRM folders. This content depends on choices the user specifies in the First Run Assistant. The synchronization finishes, and then the user can find the Siebel CRM data that CRM Desktop downloaded in the corresponding Siebel CRM folders. The user can view Siebel contacts that Siebel CRM Desktop downloaded to the IBM Notes Contacts folders. CRM Desktop does not automatically share contacts that existed in IBM Notes before you installed CRM Desktop.

Customizing How Siebel CRM Desktop Shares Native IBM Notes Items

You can customize Siebel CRM Desktop to share or not share any new native IBM Notes items that the user creates in IBM Notes, such as a IBM Notes calendar entry, contact, or To Do item.

To customize how Siebel CRM Desktop shares native IBM Notes items

- 1 Use an XML editor to open the platform_configuration.xml.
For more information, see [“Files in the Customization Package” on page 355](#).
- 2 Locate the platform section.
- 3 Add the following initialization_script section to the section that you located in [Step 2](#):

```
<platform>
  <initialization_script>
    <![CDATA[
      application.settings.set("SharedByDefault:NewItems", value);
    ]]>
  </initialization_script>
</platform>
```

where:

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

| Value | Description |
|-------|------------------------------|
| 0 | Do not share IBM Notes item. |
| 1 | Share IBM Notes item. |

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

Sharing a Calendar Entry, Contact, or To Do Item

The user can determine how Siebel CRM Desktop shares records with the Siebel Server according to the following settings:

- **Set default sharing for all new records.** The user can use the Advanced tab of the CRM Desktop - Options dialog box to change how Siebel CRM Desktop creates a new IBM Notes calendar entry, contact, or To Do item as shared or not shared.
- **Set sharing for individual records.** The user can click the Sharing Bar that CRM Desktop displays at the start of a record form to share or unshare a single record.

Suppressing the Dialog Boxes That First Run Assistant Displays

This topic describes how to suppress the dialog boxes that First Run Assistant displays.

To suppress the dialog boxes that First Run Assistant displays

- 1 Use a JavaScript editor to open the application_script.js file.
- 2 Modify the following code:

```
var fra = application.fra;
function fra_handler(fra)
{
    var current_form = null;
    var on_closed = function()
    {
        current_form = null;
        fra.exit_current_step(false);
    }
    function on_fra_step(id)
    {
        if (id == "advanced")
        {
            var xml = ui.get_dialog_xml("PropSheetHost");
            xml = helpers.replace_all(
                "$prop_sheet_layout", "options_advanced_page", xml);
            current_form = ui.create_dialog_from_xml(0, xml);
            current_form.on_closed.connect(on_closed);
            current_form.visible = true;
        }
    }
}
```

```

    }
  }
  fra.on_step.connect(on_fra_step);
  fra.add_built_in_step("welcome");
  fra.add_built_in_step("sync_filters");
  fra.add_built_in_step("sync_schedule");
  fra.add_step("advanced", session.res_string("sa-advanced_settings-capti on"),
session.res_string("sa-advanced_settings-descri ption"), "sa-advanced_setti ngs-
picture", true);
  fra.add_built_in_step("convert_i tems");
  fra.add_built_in_step("fi rst_sync");
}
function create_fra_handler(fra)
{
return fra != null ? new fra_handler(fra) : null;
}
var g_fra_handler = create_fra_handler(application.fra);

```

where:

- **bold** indicates code you can modify, as described in [Table 7 on page 96](#).

[Table 7](#) describes the code that you can modify to suppress the dialog boxes that First Run Assistant displays.

Table 7. Code That Displays the Dialog Boxes That First Run Assistant Displays

| Default Code | Description |
|--|--|
| fra.add_built_in_step("welcome"); | This code displays the Welcome dialog box. It is recommended that you do not remove it. |
| fra.add_built_in_step("sync_filters"); | This code displays the default synchronization filters. You can remove this code to hide these filters. |
| fra.add_built_in_step("sync_schedule"); | This code displays the default synchronization schedule. You can remove this code to hide this schedule. |
| fra.add_step("advanced", session.res_string("sa-advanced_settings-capti on"), session.res_string("sa-advanced_settings-descri ption"), "sa-advanced_setti ngs-pi cture", true); | This code displays the default advanced settings. You can remove this code to hide the advanced settings. |
| fra.add_built_in_step("convert_i tems"); | This code displays the native contacts conversion. You can remove this code to hide the native contacts conversion. |
| fra.add_built_in_step("fi rst_sync"); | This code displays the first synchronization step, It is required. Almost no CRM Desktop functionality is available before the first synchronization. You must not remove this code. |

Installing Siebel CRM Desktop in the Background

This topic describes how to install Siebel CRM Desktop in the background.

Using Microsoft System Center Configuration Manager to Install Siebel CRM Desktop

This topic describes how to use Microsoft System Center Configuration Manager to install Siebel CRM Desktop. For more information, see the documentation about using System Center Configuration Manager at the Microsoft TechNet Web site.

To use Microsoft System Center Configuration Manager to install Siebel CRM Desktop

- 1 Log on to the computer that includes System Center Configuration Manager, and then Open Microsoft System Center Configuration Manager 2007 or Microsoft Systems Management Server 2003.

- 2 Add all custom properties to the Windows Installer transform (.mst) file.

For more information, see ["Adding Custom Properties to the Windows Installer Transform File" on page 97](#).

- 3 Run the installer. Open a Windows command line and then enter the following command:

```
msiexec /I "CRMDesktopfor IBM Notes.msi" TRANSFORMS="crmdesktop.mst" ALLUSERS=1 /qb!
```

where:

- ALLUSERS=1 is an optional parameter. To install CRM Desktop for anyone who uses the client computer, you must include the ALLUSERS parameter.

Use the following guidelines:

- Run setup with one of the following administrative rights:
 - **Administrative rights.** Installs CRM Desktop for anyone who uses the client computer.
 - **User rights.** Installs CRM Desktop only for the person who is currently logged into Windows on the client computer. Make sure this user possesses the permissions that Windows requires to run the installer.

Adding Custom Properties to the Windows Installer Transform File

You must add all custom properties to the Windows Installer transform (.mst) file. A *custom property* is any property that MSDN (Microsoft Developer Network Platforms) does not describe. For example, SIEBEL_SERVER_PROTOCOL and SIEBEL_SERVER_PORT are custom properties. For a complete list of the custom properties you must add, see ["Setting the URL for the Siebel Server" on page 100](#).

Using a Windows Group Policy to Install Siebel CRM Desktop

This topic describes how to use a Windows group policy to install Siebel CRM Desktop for each user. It includes an optional step that describes how to install it for anyone who uses the client computer. For more information, see the documentation about using group policies at the Microsoft TechNet Web site.

To use a Windows group policy to install Siebel CRM Desktop

- 1 Log on to the computer that includes your group policy manager.
- 2 Make sure the directory that stores the installer and the .mst file is available on the local network.
- 3 Open the Microsoft Group Policy Editor.
- 4 Create an installation package in the GPO snap-in in the following branch:

Computer Configuration - > Software Settings - > Software installation

- 5 Set the Deployment type to Assigned.
- 6 Create a Windows Installer transform .mst file.
- 7 Add the path to the Windows Installer .mst transform file.
- 8 Add all custom properties to the transform file.

You cannot use the command line with a group policy object (GPO). You must specify all properties in the .mst file. For more information, see ["Adding Custom Properties to the Windows Installer Transform File" on page 97](#).

- 9 (Optional) To install CRM Desktop for anyone who uses the client computer, do the following:
 - a Set the ALLUSERS property to 1 in the Property table. You set this property in the transform file that you create in [Step 6](#).
 - b Make sure each CRM Desktop user possesses the permissions to run this msi package.

An administrator might disallow the parameter that provides these permissions. If the user does not possess these permissions, then CRM Desktop does not run the installation when it creates the IBM Notes location.

Using the Windows Command Line to Set Optional Parameters

You can use the Windows command line to set optional parameters that affect installation. You can run the CRMDesktopfor IBM Notes.msi installation package from the Windows command line interface on the client computer. Siebel CRM Desktop supports all parameters that you can set in the Windows Installer msixexec command line. For more information, see the documentation about command line options for Windows Installer at the Microsoft TechNet Web site.

To use the Windows command line to set optional parameters

- 1 On the client computer, open a Windows command line:

- a In Windows, click Start and then click Run.
 - b In the Run dialog box, enter cmd and then click OK.
- 2 Navigate to the directory that contains the CRMDesktopfor IBM Notes.msi file.

For example:

C:\Documents and Settings\username\Desktop

- 3 Enter the Windows Installer command using the following format:

```
msiexec.exe /I CRMDesktopfor IBM Notes.msi optional_parameter_1
optional_parameter_n
```

where:

- *optional_parameter* is a parameter you can enter that CRM Desktop runs. For example:

```
msiexec /i "C:\Documents and Settings\username\Desktop\CRMDesktopfor IBM
Notes.msi" SI EBEL_SERVER_HOST="si ebel server.com" SI EBEL_SERVER_PORT="80"
SI EBEL_SERVER_SUFFIX="SWEExtSource=WebService&SWEExtCmd=Execute&WSSOAP=1"
SI EBEL_SERVER_PROTOCOL="http" SI EBEL_SERVER_COMPONENT="eai /enu"
```

- (Optional) Add the following optional parameter to enable the SOAP log:

SOAP_DUMP_ENABLED=1

- (Optional) Add the following optional parameter to enable the SYNC log:

SYNC_DUMP_ENABLED=1

For more information, see [“Guidelines for Using Synchronization Log Parameters”](#) on page 99

Note the following requirements:

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

For information about how to set CRM Desktop SSO parameters, see [“Using the Windows Command Line to Set Optional Parameters for Siebel CRM SSO”](#) on page 294.

- 4 Press Enter.

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

Guidelines for Using Synchronization Log Parameters

A synchronization log includes the following parameters. These parameters measure the average percentage of CPU load time that Siebel CRM Desktop uses during synchronization:

- **kernel_cpu**. Measures the entire system.
- **process_cpu**. Measures the process that runs CRM Desktop.

It is recommended that you do not use these parameters to debug CRM Desktop. Instead, it is recommended that you use other tools to measure CPU load time, such as the Process Explorer system utilities for Windows or the Windows Task Manager. Using these parameters consumes resources and might degrade performance.

Hiding Dialog Boxes That Require User Input

You can use the optional QR parameter to hide dialog boxes that require user input.

To hide dialog boxes that require user input

- Append the QR parameter to the msi exec command.

For example:

```
msi exec. exe /I CRMDesktopfor IBM Notes.msi INSTALLDIR=c:\My_Custom_Directory\QR
```

If you add this parameter, then the CRMDesktopfor IBM Notes.msi installation package does not display dialog boxes that require user input.

Setting the Installation Directory of the CRM Desktop Add-In

You can use the optional INSTALLDIR parameter to change the default location where the CRMDesktopfor IBM Notes.msi installation package saves files during installation for a single user. CRMDesktopfor IBM Notes.msi installs to the following directory, by default:

```
c:\Documents and Settings\username\Application Data\Oracle\CRM Desktopfor IBM Notes\
```

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

- Enter the following parameter on the msi exec command line anywhere after the mandatory CRMDesktopfor IBM Notes.msi name parameter:

```
INSTALLDIR=directory_path
```

For example:

```
\Documents and Settings\username\Desktop\CCRMDesktopfor IBM Notes.msi
```

where:

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

Setting the URL for the Siebel Server

You can specify the URL that the Synchronization Engine uses to connect with the Siebel Server.

To set the URL for the Siebel Server

- Enter the following parameters on the msi exec command line anywhere after the mandatory CRMDesktopfor IBM Notes.msi name parameter:

```
SI EBEL_SERVER_PROTOCOL=protocol SI EBEL_SERVER_HOST=host_name_or_address  
SI EBEL_SERVER_PORT=server_port SI EBEL_SERVER_COMPONENT=component_name  
SI EBEL_SERVER_SUFFIX=request_suffix
```

where:

- *protocol* is http. HTTP 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. 80 is the default value.
- *component_name* is eai/enu. eai/enu is the default value.
- *request_suffix* is the following default value:

```
?SWEExtSource=WebService&SWEExtCmd=Execute&WSSOAP=1
```

For example:

```
msiexec.exe /I CRMDesktopfor IBM Notes.msi SI EBEL_SERVER_PROTOCOL=http  
SI EBEL_SERVER_HOST=sdcv440s133.siebel.com SI EBEL_SERVER_PORT=80  
SI EBEL_SERVER_COMPONENT=eai /enu SI EBEL_SERVER_SUFFIX=  
SWEExtSource=WebService&SWEExtCmd=Execute&WSSOAP=1
```

No parameters are required.

Any information that you set in these parameters sets the parameter values in the Windows Registry, so the user is not required to set them. For example, the protocol variable of the SI EBEL_SERVER_PROTOCOL parameter overrides the Siebel:Protocol entry in the Windows Registry. For more information, see [“Using the Windows Registry to Control Siebel CRM Desktop” on page 103](#)

Troubleshooting Siebel CRM Desktop Installation

Siebel CRM Desktop might display a message during installation that is similar to one of the following error messages:

The System cannot open the device or file specified

Error 2755. Server returned unexpected error 110 attempting to install package

This problem might be due to the fact that the CRMDesktopfor IBM Notes.msi file is encrypted or is located in a directory where the user does not possess run permissions.

To troubleshoot Siebel CRM Desktop installation

- 1 Open Windows Explorer.
- 2 Right-click the .msi installation file and then click Properties.
- 3 In the General tab, click Advanced.

- 4 Make sure the following option does not contain a check mark and then click OK.
 Encrypt contents to secure data
- 5 Reinstall CRM Desktop.

7

Administering Siebel CRM Desktop

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

- [Controlling the Behavior of Siebel CRM Desktop on page 103](#)
- [Controlling How Siebel CRM Desktop Handles CRM Data on page 108](#)
- [Removing Siebel CRM Desktop on page 113](#)
- [Administering Logging on page 115](#)
- [Troubleshooting Problems That Occur with Siebel CRM Desktop on page 122](#)

Controlling the Behavior of Siebel CRM Desktop

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

- [Using the Windows Registry to Control Siebel CRM Desktop on page 103](#)
- [Using the Metadata to Control Siebel CRM Desktop on page 105](#)

Using the Windows Registry to Control Siebel CRM Desktop

You can use Windows Registry keys to control Siebel CRM Desktop behavior. For example, you can specify the following items:

- Directory paths
- Passwords
- Synchronization parameters
- Connection timeouts
- Host names
- Ports
- Credentials

If you must reinstall CRM Desktop at some point in the future, then during this installation CRM Desktop deletes any changes you have made to the registry.

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 83.](#)

To use the Windows Registry to control Siebel CRM Desktop

- 1 In Microsoft Windows, choose Start and then click Run.
- 2 In the Run dialog box, enter REGEDIT and then click OK.
- 3 Add or modify Windows Registry keys, as necessary.

To automate changes to Windows Registry keys, you can use an administrative tool, such as Systems Management Server or Marimba.

For information about the keys you can change, see [“Registry Keys You Can Use with Siebel CRM Desktop” on page 325.](#)

Configuring Siebel CRM Desktop to use HTTPS

You can configure the URL protocol to use HTTPS (Hypertext Transfer Protocol Secure). For more information, see [“Setting the URL for the Siebel Server” on page 100.](#)

To configure Siebel CRM Desktop to use HTTPS

- 1 Open a Windows command line, and then type `regedit .exe`.

For more information, see [“Using the Windows Command Line to Set Optional Parameters” on page 98.](#)

- 2 Set the Siebel:Protocol registry key to https.

For more information, see [“Registry Keys That Affect Credentials” on page 329.](#)

Overriding Windows Registry Keys That Locate the Siebel Server

You can use values in the Ln_connector_configuration.xml file to override the following registry settings:

- Siebel:ComponentName
- Siebel:RequestSuffix

The CRM Desktop add-in uses these entries to locate the Siebel Server. For more information, see [“Setting the URL for the Siebel Server” on page 100](#) and [“Using the Windows Registry to Control Siebel CRM Desktop” on page 103](#).

CAUTION: You cannot use the Login dialog box of the CRM Desktop add-in to edit the name and suffix of the server component. 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 keys that locate the Siebel Server

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

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

where:

- *setting name* is the registry setting that CRM Desktop must override.
- *type* is the key type. 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 That Customizes Synchronization” on page 384](#).

Using the Metadata to Control Siebel CRM Desktop

You can use the metadata to control Siebel CRM Desktop. For example, you can set limits for the following items:

- Length of a repeating Calendar entry
- Size of a file attachment
- Visibility of an object

The files that this topic describes are part of the customization package. You can use any editor that supports editing in JavaScript or XML, such as Notepad, to modify one of these files.

To use the metadata to control Siebel CRM Desktop

- 1 Set the maximum number of days, weeks, months, or years that CRM Desktop creates for a repeating Calendar entry that does not match a Siebel CRM repeating pattern. You modify the `recurrence_processing.js` file. Use values from the following table.

| Variable with Default Value | Description |
|--|---|
| <code>var daily_max_length = 12</code> | Sets the maximum number of months that CRM Desktop uses when it creates a repeating Calendar entry. This Calendar entry occurs daily. |
| <code>var weekly_max_length = 12</code> | Sets the maximum number of months that CRM Desktop uses when it creates a repeating Calendar entry. This Calendar entry occurs weekly. |
| <code>var monthly_max_length = 24</code> | Sets the maximum number of months that CRM Desktop uses when it creates a repeating Calendar entry. This Calendar entry occurs monthly. |
| <code>var yearly_max_length = 60</code> | Sets the maximum number of months that CRM Desktop uses when it creates a repeating Calendar entry. This Calendar entry occurs yearly. |

- 2 Edit the `siebel_meta_info.xml` file, using values from the following table. For more information, see [“Customizing Meta Information” on page 157](#).

| Variable | Description |
|-------------------------------------|---|
| <code>max_commands_per_batch</code> | Sets the maximum number of commands for each batch. For more information, see the documentation in the <code>siebel_meta_info.xsd</code> file available in Article ID 1502099.1 on My Oracle Support. |
| <code>max_ids_per_command</code> | Sets the maximum number of object IDs. For more information, see the documentation in the <code>siebel_meta_info.xsd</code> file available in Article ID 1502099.1 on My Oracle Support. |

| Variable | Description |
|--------------------|---|
| open_with_url_tmpl | Sets a template for the code that CRM Desktop uses to create a URL to open the Siebel Web Client. For more information, see “Setting the URL That Siebel CRM Desktop Uses to Open the Siebel Web Client” on page 107. |
| ViewMode | <p>Sets the visibility of an object. You can use one of the following values:</p> <ul style="list-style-type: none"> ■ Sales Rep ■ Personal ■ Organization ■ All <p>For example, ViewMode = Sales Rep.</p> <p>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. For more information, see “Controlling the View Mode During Synchronization According to Object Type” on page 143.</p> |

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

You can specify 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. The user can navigate to the Siebel Web Client from the context of a record in IBM Notes to examine more details about the record. This feature is useful if the user must do work that requires data from the Siebel Web Client that is not available in IBM Notes. CRM Desktop does the following work:

- If the user clicks the Siebel icon on the CRM Desktop toolbar, or if the user clicks the Siebel icon on the Extension Bar of the form, then CRM Desktop opens a new browser window for the Siebel Web Client. Siebel CRM Desktop displays the detail form of the record that is currently chosen in IBM Notes. For example, a contact, an opportunity, or an account. The Siebel URL that you specify determines the browser window that 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 CRM Desktop displays a message that is similar to This Record is Not Found in Siebel. This situation can occur if the user creates the record in IBM Notes 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 IBM Notes.
- Does not display the button if the user chooses a native IBM Notes contact in IBM Notes.
- 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 information, see [“Customizing Meta Information” on page 157](#). For more information about templates, see the information about the `open_with_url_tmpl` variable in [“Using the Metadata to Control Siebel CRM Desktop” on page 105](#).

To set 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 [“Files in the Customization Package” on page 355](#).
- 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 on.
- 3 Modify and then insert the `open_with_url_tmpl` element in the object element of the object type that CRM Desktop must open in the Siebel Web Client.
For more information, see the documentation in the `siebel_meta_info.xsd` file available in Article ID 1502099.1 on My Oracle Support.
- 4 Repeat [Step 1](#) and [Step 3](#) for each type of object that CRM Desktop must open in the Siebel Web Client.

Controlling How Siebel CRM Desktop Handles CRM Data

This topic describes how to control how Siebel CRM Desktop handles CRM data. It includes the following topics:

- [Controlling How Siebel CRM Desktop Assigns Calendar Entry Owners on page 108](#)
- [Controlling How Siebel CRM Desktop Handles Email Attachments on page 109](#)
- [Controlling the Maximum Size of an Attachment on page 110](#)

Controlling How Siebel CRM Desktop Assigns Calendar Entry Owners

You can use Siebel Multi-Org (multiple organization) to administer a calendar entry that a non-Siebel user creates, such as an invitation from an external contact. For more information, see [“How Siebel CRM Assigns Meeting Organizers” on page 44](#).

To control how Siebel CRM Desktop assigns calendar entry owners

- 1 Log in to the Siebel CRM 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 IBM Notes meeting, then a duplication error occurs. For more information, see [“Resolving Synchronization Conflicts” on page 148](#).

- 4 In the System Preference Value field, enter a user name.

Use the following guidelines:

- Make sure relationships exist between the user you specify as the Generic Siebel Owner and all organizations. This configuration allows any other user who creates a shared meeting to choose the user. If relationships do not exist between the user that you specify as the Generic Siebel Owner and all organizations, then the selection that the user makes fails and CRM Desktop displays an error message that indicates the user cannot choose the current record.
- Do not specify a real user as the Generic Siebel Owner. If you do this, then this user receives every calendar entry that matches 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.

Controlling How Siebel CRM Desktop Handles Email Attachments

IBM Notes stores the IBM Notes email message and any attachments that this email contains in a message. The user can do one of the following to control how CRM Desktop handles an email attachment:

- Use the CRM Desktop - Options dialog box while using the First Run Assistant.
- Right-click the CRM Desktop icon in the system tray, choose Options, and then click the Advanced tab in the CRM Desktop - Options dialog box.

You can also use the Windows Registry to control how CRM Desktop handles an email attachment.

To control how Siebel CRM Desktop handles email attachments

- 1 Open the Windows Registry and then locate the following key:

HKEY_CURRENT_USER\Software\Oracle\CRM Desktopfor IBM Notes

For more information, see [“Using the Windows Registry to Control Siebel CRM Desktop” on page 103](#).

- 2 Set the MailProcessing:AttachmentsHandling key to one of the following values:

| Value | Description |
|-------|---|
| 1 | Attach the file attachments that the message contains to the Siebel CRM activity. |
| 0 | Do not attach anything to the Siebel CRM activity. |

Controlling the Maximum Size of an Attachment

You can control the maximum size of an attachment that Siebel CRM Desktop allows the user to attach to an object. The example in this topic sets the maximum size of an account attachment to 1 MB.

To control the maximum size of an attachment

- 1 (Optional) Set the maximum size of a file that CRM Desktop can attach to a specific object type that it synchronizes from the client to the Siebel Server:

- a Use an XML editor to open the connector_configuration.xml file.
- b Locate the following tag:

```
type id="Attachment"
```

Make sure you locate the tag that includes child collection container_type tags. This tag is typically the second instance of the type id="Attachment" tag in the connector_configuration.xml file.

- c Modify the FileSize attribute of one of the child tags of the tag you located in [Step b](#).

For example, to modify the maximum size for an account attachment, you modify the value type="integer" tag that the Account collection container contains. You use the following code. Bold text indicates the code you must change:

```
<type id="Attachment">
  <group link="or">
    <collection container_type="Account" foreign_key="ParentId">
      <primary_restriction>
        <group link="and">
          <binary field="FileSize" condition="le">
            <value type="integer">1048576</value>
          </binary>
        </group>
      </primary_restriction>
    </collection>
  </group>
</type>
```

where:

- 1048576 is the number of bytes in 1 MB.

- d Save your changes and then close the Ln_connector_configuration.xml file.

- 2 (Optional) Set the maximum size of an attachment that CRM Desktop can synchronize from the Siebel Server to the client:

- a Use an XML editor to open the siebel_meta_info.xml file.

- b** Set the value for the `max_out_obj_size` tag.

The `max_out_obj_size` tag sets the maximum size of an attachment for any object that CRM Desktop synchronizes from the Siebel Server to the client. If CRM Desktop encounters an attachment that is larger than the value you specify, then it displays an error message. For example, the following code sets this maximum size to 1 MB:

```
ErrMsg="#out_object_too_big_err_msg">1048576</max_out_obj_size>
```

where:

- 1048576 is the number of bytes in 1 MB.

This tag does not limit the size of a file that the user attaches in the CRM Desktop add-in.

- c** Save your changes and then close the `siebel_meta_info.xml` file.
- d** Open IBM Domino Designer, and then locate the `SBL.BusinessLogic` Script Library.
For more information, see ["Opening IBM Domino Designer" on page 111](#).
- e** Open the `GetAttachementOptions` function.
- f** Locate the following code:

```
Set Options = New AttachmentOptions("Name", "Body", CLng(max_file_size),  
ATTACHMENT_DUPLICATE_ACTION_NUMERATE, "", "")
```

where:

- `max_file_size` is an integer that specifies the maximum size of an attachment, in megabytes, that CRM Desktop allows the user to attach to an object, such as an account. CRM Desktop comes predefined with `CLng(5)`. For more information, see the topic about the `CLng` function at the Domino Designer documentation web site at <http://www.ibm.com/developerworks/lotus/documentation/dominodesigner/>.
- For example, you can use the following code:

```
Set Options = New AttachmentOptions("Name", "Body",  
CLng(1), ATTACHMENT_DUPLICATE_ACTION_NUMERATE, "", "")
```

where:

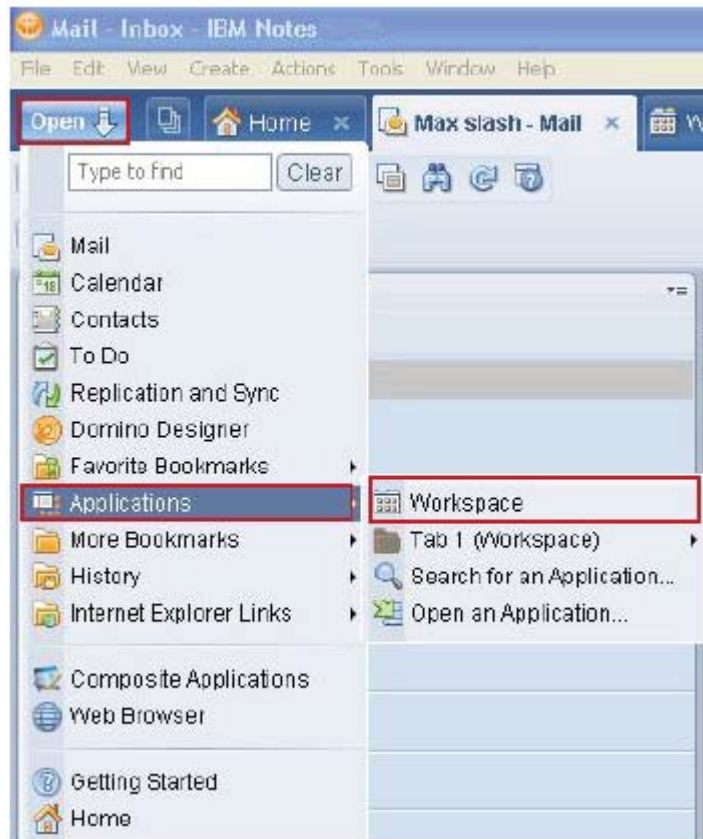
- `CLng(1)` specifies a maximum file size attachment of 1 megabyte.
- g** Save your changes and then close the `SBL.BusinessLogic` Script Library.

Opening IBM Domino Designer

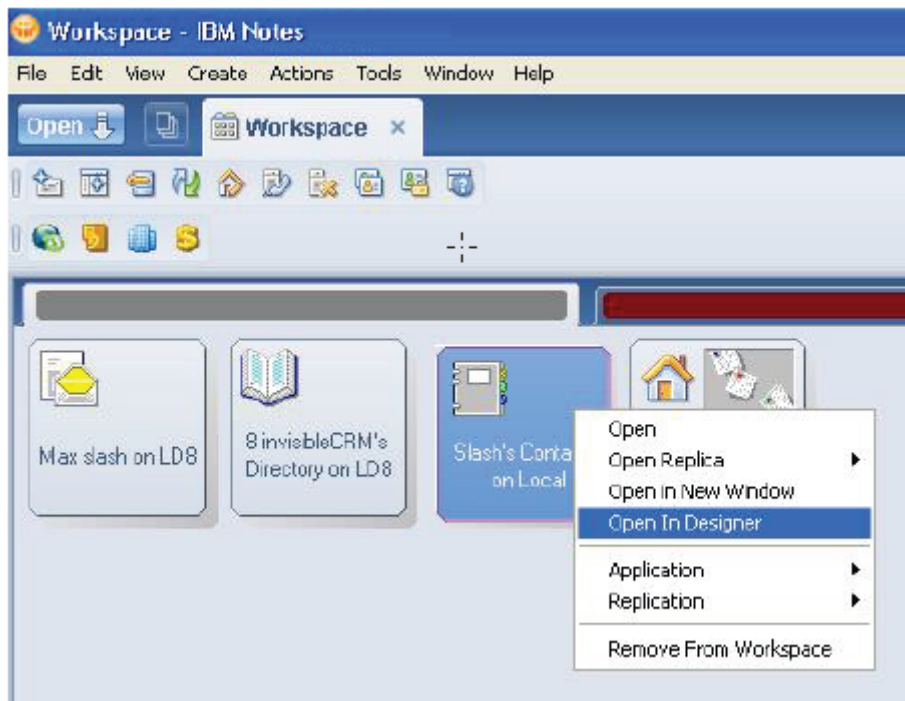
This topic describes how to open IBM Domino Designer.

To open IBM Domino Designer

- 1 In IBM Notes, click Open, click Applications, and then click Workspace:



- 2 Right-click a database that contains the object types that Siebel CRM Desktop uses, and then click Open in Designer:



Removing Siebel CRM Desktop

This topic describes how to remove the CRM Desktop add-in. It includes the following topics:

- [Removing the CRM Desktop Add-In for a Single User on page 113](#)
- [Removing the CRM Desktop Add-In for Multiple Users on page 114](#)
- 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 CRM Desktop Add-In" on page 62. Controlling the Data That Siebel CRM Desktop Removes on page 114

Removing the CRM Desktop Add-In for a Single User

This topic describes how to remove the CRM Desktop add-in if it is installed for a single user.

To remove the CRM Desktop add-in for a single user

- 1 Do the following:
 - a Do a synchronization in IBM Notes.

- b Backup personal data.

To avoid a loss of data, it is recommended that the user synchronize and back up all personal data before removing the CRM Desktop add-in. For more information, see [“How Siebel CRM Desktop Handles Items If the User Removes the CRM Desktop Add-In”](#) on page 62.

2 Remove the CRM Desktop add-in:

- a In Microsoft Windows, 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.

CRM Desktop automatically does the following work:

- Removes the data structure
- Removes Siebel CRM data
- Removes every shared calendar entry and To Do item that it created to support Siebel CRM activities
- Removes shared contacts

CRM Desktop treats native IBM Notes items in the following ways:

- Converts every unshared calendar entry, To Do item, and contact to a native IBM Notes item
- Leaves every shared calendar entry and To Do item that originated in IBM Notes as a native IBM Notes item in the corresponding IBM Notes folder
- Leaves every native IBM Notes item and IBM Notes email message in the corresponding IBM Notes folder

Removing the CRM Desktop Add-In for Multiple Users

You can use the System Center Configuration Manager (SCCM) to remove the CRM Desktop add-in in the background if it is installed for multiple users. You can use the SCCM management console to open the distribution package that you created when you installed CRM Desktop, open the Siebel CRM Desktop program, and then enter the following value in the command line:

```
msiexec /x "CRMDesktopfor IBM Notes.msi" /qr
```

You can enter the following command to remove CRM Desktop SSO for multiple users:

```
msiexec /x "Invi si bl eSSOModul e.msi" /qr
```

For more information, see [“Removing or Upgrading CRM Desktop SSO”](#) on page 296.

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 CRM Desktop Add-In”](#) on page 62.

This topic describes how to control the data that Siebel CRM Desktop removes if the user removes CRM Desktop. The CleanupHandler class in the SBL.Handlers script library allows you to specify the custom data that it removes from IBM Notes. If the user removes Siebel CRM Desktop or changes credentials, then it removes from IBM Notes all the custom data that the Ln_siebel_basic_mapping.xml file describes. You can configure Siebel CRM Desktop to not delete data for an object type. It does not remove data that the user creates in the native IBM Notes application that it shares with the Siebel Server, such as a Calendar entry, To Do item, or email message. For more information, see [“XML Code That Customizes Platform Configuration” on page 384](#).

To control the data that Siebel CRM Desktop removes

- 1 Open IBM Domino Designer, and then locate the SBL.Handlers Script Library.
- 2 Locate the CleanupHandler class.
- 3 Locate RemoveAllCustomDocuments() sub.
- 4 Locate the following code:

```
Set collection = db.Search({@IsAvailable() & ITEMN_TYPEID & {}}, Nothing, 0)
```

where:

{@IsAvailable() & ITEMN_TYPEID & {}} is a Notes @function formula that defines the selection criteria.

- 5 Modify the selection criteria.

For example:

```
{@IsAvailable() & ITEMN_TYPEID & {}} & { & ITEMN_TYPEID & { != "Action"}}
```

In this example, CRM Desktop deletes all crm-related data except Action objects.

For more information, see the Search function for the NotesDatabase class in the Domino Designer documentation web site <http://www.ibm.com/developerworks/lotus/documentation/dominodesigner/>.

- 6 Save your changes and then close the SBL.Handlers Script Library.

Administering Logging

This topic describes how to administer logging. It includes the following topics:

- [Log Files You Can Use with Siebel CRM Desktop on page 116](#)
- [Assigning Logging Profiles for Siebel CRM Desktop on page 117](#)
- [Creating Custom Logging Profile on page 119](#)
- [Creating Installation Log Files for Siebel CRM Desktop on page 120](#)
- [Administering Logging on the Siebel Server on page 121](#)
- [Using Script to Modify Logging Levels on page 121](#)

Log Files You Can Use with Siebel CRM Desktop

Table 8 describes log files that you can use with Siebel CRM Desktop. For information about how to enable these log files, see [“Assigning Logging Profiles for Siebel CRM Desktop” on page 117](#).

Table 8. Log Files That You Can Use with Siebel CRM Desktop

| Description | Where Stored |
|--|---|
| General Log. Includes information about general application events. | <p>Windows XP:</p> <p>C:\Documents and Settings\user\Application Data\Oracle\CRM Desktop for IBM Notes\Profile\Logs\General Log</p> <p>Windows Vista and Windows 7:</p> <p>C:\Users\user\AppData\Roaming\Oracle\CRMDesktop\Profile\Logs\General Log</p> <p>CRM Desktop stores the first general log file as log.0000.txt. If this first file reaches 10 MB, then it creates a new file and increments the name of this file by 1. For example, log.0001.txt. A maximum of eight files can exist. If the eighth file reaches 10 MB, then CRM Desktop deletes the oldest General Log.</p> |
| Exception Log. Includes information about CRM Desktop exceptions. These log files are for Oracle internal use only. | <p>Windows XP:</p> <p>C:\Documents and Settings\user\Application Data\Oracle\CRM Desktop for IBM Notes\Profile\Logs\ExceptionLog</p> <p>Windows Vista and Windows 7:</p> <p>C:\Users\user\AppData\Roaming\Oracle\CRM Desktop for IBM Notes\Profile\Logs\ExceptionLog</p> <p>CRM Desktop stores the first exception log file as ex_trace.0000.txt. If this first file reaches 10 MB, then it creates a file and increments the name of this file by 1. For example, ex_trace.0001.txt. A maximum of eight files can exist. If the eighth file reaches 10 MB, then CRM Desktop deletes the oldest file.</p> |

Table 8. Log Files That You Can Use with Siebel CRM Desktop

| Description | Where Stored |
|---|--|
| Crash Log. Includes information about IBM Notes and CRM Desktop add-in failures. | <p>Windows XP:</p> <p>C:\Documents and Settings\user\Application Data\Oracle\CRM Desktop for IBM Notes\Profile\Logs\CrashDump</p> <p>Windows Vista and Windows 7:</p> <p>C:\Users\user\AppData\Roaming\Oracle\CRM Desktop for IBM Notes\Profile\Logs\CrashDump</p> <p>A maximum of 48 crash log files can exist. If 48 files exist, and if CRM Desktop must create another crash log, then it deletes the oldest file.</p> |
| SOAP Log. Includes information about requests that CRM Desktop sends to the Siebel Server and replies that it receives from the Siebel Server. | <p>Windows XP:</p> <p>C:\Documents and Settings\user\Application Data\Oracle\CRM Desktop for IBM Notes\Profile\Logs\SoapDump</p> <p>Windows Vista and Windows 7:</p> <p>C:\Users\user\AppData\Roaming\Oracle\CRM Desktop for IBM Notes\Profile\Logs\SoapDump</p> <p>A maximum of 48 SOAP log files can exist. If 48 SOAP log files exist, and if CRM Desktop must create another SOAP log file, then it deletes the oldest SOAP log.</p> |
| Synchronization Log. Includes synchronization events. | <p>Windows XP:</p> <p>C:\Documents and Settings\user\Application Data\Oracle\CRM Desktop for IBM Notes\Profile\Logs\SyncDump</p> <p>Windows Vista and Windows 7:</p> <p>C:\Users\user\AppData\Roaming\Oracle\CRM Desktop for IBM Notes\Profile\Logs\ SyncDump</p> <p>A maximum of 48 synchronization log files can exist. If 48 files exist, and if CRM Desktop must create another file, then it deletes the oldest file.</p> |

Assigning Logging Profiles for Siebel CRM Desktop

A *logging profile* is a set of parameters that determine logging settings. Siebel CRM Desktop comes with three predefined logging profiles and one custom profile. You can assign a predefined logging profile. You cannot change the set of parameters that a predefined logging profile uses. You can assign only one logging profile at a time.

To assign a logging profile for Siebel CRM Desktop

- 1 On the client computer, right-click the CRM Desktop icon in the system tray.
- 2 Choose Options and then click the Advanced tab in the CRM Desktop - Options dialog box.
- 3 Click Configure Logging and Reporting.
- 4 In the Logging Configuration dialog box, choose the Logging Policy using values from the following table.

| Logging Policy | Log Enabled |
|----------------|---|
| Basic | General Log CRM Desktop sets the verbosity for the General Log to Info. |
| Detailed | CRM Desktop enables the following log files: <ul style="list-style-type: none"> ■ General Log ■ Exception Log ■ Crash Dump ■ Sync Dump ■ SOAP Dump CRM Desktop sets the verbosity for the General Log to the following: <ul style="list-style-type: none"> ■ Info for Detailed ■ Debug for Exhaustive |
| Exhaustive | |
| Custom | CRM Desktop uses a custom profile that you define. For more information, see “Creating Custom Logging Profile” on page 119 . |

- 5 Set Logging Verbosity.
For more information, see [“Setting Logging Verbosity” on page 119](#).

- 6 (Optional) Choose one or more of the following items:

- Log Application Exceptions
- Log Application Crashes
- Log Sync Dumps
- Log SOAP Dumps

At run-time, CRM Desktop creates log entries for each item you choose. For more information, see [“Log Files You Can Use with Siebel CRM Desktop” on page 116](#).

- 7 (Optional) Choose View Files for any item.

If you choose View Files, then CRM Desktop displays the folder that includes the log files for the corresponding log type.

8 Click OK.

Siebel CRM Desktop replaces the current logging settings with the setting that the profile you choose contains. It copies the parameters of this logging profile to the CRM Desktop logging settings in the Windows Registry in the following key:

HKEY_CURRENT_USER\Software\Oracle\CRM Desktop for IBM Notes\Logging

Setting Logging Verbosity

Logging verbosity is the level of detail that Siebel CRM Desktop writes to the General Log. You can set logging verbosity to one of the following values:

- **Debug.** Logs information messages, warnings, and all errors.
- **Info.** Logs only information messages.
- **Warning.** Logs only warnings.
- **Error.** Logs only errors.
- **Fatal.** Logs only fatal errors.

Creating Custom Logging Profile

The example in this topic describes how to create a logging profile that enables General Log, Synchronization Dump, and SOAP Dump log entries and sets verbosity for the General Log to Warning.

To create a custom logging profile

- 1 In Microsoft Windows, choose Start and then click Run.
- 2 In the Run dialog box, enter REGEDIT and then click OK.
- 3 Locate the following key:

HKEY_CURRENT_USER\Software\Oracle\CRM Desktop for IBM Notes\LoggingProfiles

- 4 Create a subkey in the key that you located in [Step 3](#).

Enter a name for this subkey. At run time, CRM Desktop displays this name in the logging profile in the Logging Policy drop-down list of the Logging Configuration dialog box. You cannot localize a profile name. CRM Desktop stores this name in the registry with the settings that you specify for the profile. For more information, see [“Assigning Logging Profiles for Siebel CRM Desktop” on page 117](#).

- 5 Create the following subkeys in the subkey that you create in [Step 4](#). Create one subkey for each row that the following table contains.

| Sub Key Name | Enable | Description |
|--------------|--------|--|
| GeneralLog | Yes | Set the value of the log_level parameter to the following decimal value: 3000 This value configures CRM Desktop to log only warnings. For performance reasons, it is recommended that CRM Desktop log only warnings during normal operations. For more information, see "Parameters You Can Use with the General Log" on page 336 . |
| ExceptionLog | No | For more information, see "Parameters You Can Use with the Exception Log" on page 339 . |
| CrashDump | No | For more information, see "Parameters You Can Use with the Crash Log" on page 339 . |
| SoapDump | Yes | For more information, see "Parameter You Can Use with the SOAP Log" on page 342 . |
| SyncDump | Yes | For more information, see "Parameters You Can Use with the Synchronization Log" on page 343 . |

If you remove CRM Desktop at some point in the future, then it clears all registry settings during removal.

- 6 Assign your custom profile.
For more information, see ["Assigning Logging Profiles for Siebel CRM Desktop" on page 117](#).
- 7 (Optional) You can use one of following items to distribute registry modifications across the network:
- Imported registration (.reg) files
 - regini.exe
 - Group policy
 - System policy

Creating Installation Log Files for Siebel CRM Desktop

This topic describes how to create installation log files.

To create installation log files for Siebel CRM Desktop

- 1 Run the CRM Desktop installer with the following command line parameter:


```
"msiexec /I vx! * /log_path /i CRMDesktopfor IBM Notes.msi "
```

where:

- */log_path* specifies where CRM Desktop stores the log file.

For example, run the following command:

```
"msiexec /I vx! * C:\admin\logs\log.txt /i CRMDesktopfor IBM Notes.msi "
```

This command installs CRM Desktop and saves the installation logs in the log.txt file in the following directory:

```
C:\admin\logs\
```

For more information, see ["Using the Windows Command Line to Set Optional Parameters" on page 98](#).

Administering Logging on the Siebel Server

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 (Enterprise Application Integration) stores these logs in the following file:

```
Siebel Server install directory\aei obj mgr_language_code.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 logging on the Siebel Server

- 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*.

Using Script to Modify Logging Levels

You can use the application.logger object in a script to modify logging levels.

To use script to modify logging levels

1 Use a JavaScript editor to open the application.js file.

2 Add the following code:

```
var log_settings = application.logger.settings.log_settings_1;  
log_settings_2;  
log_settings.save();
```

where:

■ *log_settings_1* is set to one of the following values:

- general_log_settings
- exception_log_settings
- crash_dump_settings
- sync_dump_settings
- soap_dump_settings

■ *log_settings_2* is set to one of the following values:

- log_settings["enabled"] = 1
- log_settings.set ("enabled", 1))

For example:

```
var log_settings = application.logger.settings.general_log_settings;  
log_settings["enabled"] = 1;  
log_settings.save();
```

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:

- [Troubleshooting Problems That Occur When Siebel CRM Desktop Connects to the Siebel Server on page 123](#)
- [Troubleshooting Problems That Occur During Synchronization on page 124](#)

For information about accessing log files, see [“Where Siebel CRM Desktop Stores Data in the File System” on page 82](#).

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 9](#). CRM Desktop uses the log.0000.txt file to log errors. It increments this log file name each time it creates another log file.

Table 9. Problems That Occur When CRM Desktop Connects to the Siebel Server

| Symptom or Error Message | Solution |
|--|---|
| <p>CRM Desktop creates an entry in the log.0000.txt file that is similar to the following error:</p> <pre>Synchroni zation canceled by error: siebel _service: failed attempting to connect to siebel</pre> <p>This error occurs if CRM Desktop cannot connect to the EAI Object manager.</p> | <p>You can make sure the following items are up and running:</p> <ul style="list-style-type: none"> ■ Siebel Server ■ Server component for the EAI Object manager |
| <p>CRM Desktop creates an entry in the log.0000.txt file that is similar to the following error:</p> <pre>Excepti on ' class siebel :: siebel _cntr_excepti on' throwi ng: siebel _service: SOAP response has unexpected structure.</pre> <p>This error typically occurs if the architecture component that CRM Desktop calls returns an HTTP error instead of an expected SOAP message and CRM Desktop cannot parse the error. An Internal Server Error is an example of an HTTP error.</p> | <p>You can look for potential problems in the following logs:</p> <ul style="list-style-type: none"> ■ SOAP log ■ Web Server log ■ Siebel Application Interface log ■ EAI log |

Table 9. Problems That Occur When CRM Desktop Connects to the Siebel Server

| Symptom or Error Message | Solution |
|--|--|
| <p>CRM Desktop creates an entry in the log.0000.txt file that is similar to the following error:</p> <pre>Exception 'struct win32_exceptions: :inet_cannot_connect' throwing: WinInet: Cannot connect to Internet! Synchronization canceled by error: WinInet: Cannot connect to Internet</pre> <p>This error occurs if CRM Desktop cannot connect to the Web server.</p> | <p>You can do the following work:</p> <ul style="list-style-type: none">■ Make sure the Web server is running.■ Make sure the computer that runs the client can connect to the Web server. |
| <p>When the user attempts to log in to the client, CRM Desktop displays a message that is similar to the following:</p> <pre>siebel_service: SOAP response has unexpected structure</pre> <p>It creates an entry in the log.0000.txt file that is similar to the following error:</p> <pre>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.</pre> <p>This error occurs if the responsibility that is associated with the user is not associated with a customization package.</p> | <p>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 78.</p> |

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 Synchronization Errors”](#) on page 74.

Resolving Exceeded Row Size Problems

The user might encounter an error that is similar to the following during the initial synchronization:

```
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 Change the value for the DSMaxFetchArraySize parameter.
For more information, see [Step 2 on page 81](#).
- 2 Notify users to synchronize data.
- 3 If the problem persists, then get help from Oracle.
For more information, see ["Getting Help From Oracle" on page 19](#)

8

Controlling Synchronization

This chapter describes how to control synchronization. It includes the following topics:

- [Controlling Synchronization Filters on page 127](#)
- [Controlling Synchronization Time, Day, and Size on page 131](#)
- [Controlling Other Configurations That Affect Synchronization on page 140](#)
- [Resolving Synchronization Conflicts on page 148](#)

For other ways to administer options that affect synchronization, see [“Using the Windows Registry to Control Siebel CRM Desktop” on page 103](#).

Controlling Synchronization Filters

This topic describes how to control synchronization filters. It includes the following topic:

- [Controlling the Object Types That Siebel CRM Desktop Displays in the Filter Records Tab on page 127](#)
- [Controlling the Synchronization Exceptions Button In the Filter Records Tab on page 128](#)
- [Controlling the Date Range in the Filter Records Tab on page 129](#)
- [Controlling the Fields That Display in a Filter on page 130](#)

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 CRM Desktop add-in. This option allows you to customize the filters and other settings that CRM Desktop displays. It displays the following objects in the Filter Records tab of the Synchronization Control Panel dialog box, by default:

- 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 Ln_connector_configuration.xml file.
For more information, see [“Files in the Customization Package” on page 355](#).
- 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 one of the following values:
 - **True.** Hides the object.
 - **False.** Displays the object.
 For more information, see [“View Tag of the Connector Configuration File” on page 386](#).
- 5 Save your changes and then republish the customization package.
For more information, see [“Republishing Customization Packages” on page 80](#).
- 6 (Optional) Add a new object to the list of objects that 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 Exceptions Button In the Filter Records Tab

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 It uses the following filters to identify the records that it must synchronize from the Siebel Server:
 - **User 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 CRM Desktop to not synchronize a contact that includes an inactive status from Siebel CRM to IBM Notes.
- 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 the Exclusions List includes a record, and if no other record references this record, then CRM Desktop removes it from IBM Notes.

To control the Synchronization Exceptions button in the Filter Records tab

- 1 Use an XML editor to open the Ln_connector_configuration.xml file.
- 2 Locate the following features tag:


```
</features>
```
- 3 Make sure the following attribute that resides in the features tag that you located in [Step 2](#) is set to true:


```
enable_sync_exclusions
```

This configuration displays the Synchronization Exceptions button on the Filter Records screen of the Synchronization Control Panel.

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.
- IBM Notes displays contact 1 in the Contacts folder of the CRM and Personal Contacts view and account 1 in the Accounts folder of the Siebel Accounts view. It does this after the first synchronization finishes.
- The user deletes account 1 and then CRM Desktop automatically moves account 1 to the Exclusions List.

Controlling the Date Range in the Filter Records Tab

The user can choose a predefined value from the Value drop-down list in the CRM Desktop Filter - Edit Criterion dialog box to modify the date range that Siebel CRM Desktop uses in a synchronization filter. If the user changes the date range, then the number of records that match the filter also change. For example, the user can set the filter to synchronize activities that start a month ago. If the user changes this date, then CRM Desktop adjusts the number of activities it synchronizes. You can customize these predefined date ranges. For more information, see [“Filters in the CRM Desktop Filter - Edit Criterion Dialog Box” on page 344](#).

Customizing the Predefined Date Ranges

You can customize the Predefined Date Ranges that CRM Desktop uses for the filters that it displays in the Value drop-down list in the CRM Desktop Filter - Edit Criterion dialog box.

To customize the predefined date ranges

- 1 Use an XML editor to open the Ln_connector_configuration.xml file.
- 2 Locate the sliding_dates_presets tag.
- 3 Change the value for a preset name.

The following code comes predefined with CRM Desktop. To change a predefined date range, you change the number that the value attribute contains:

```
<sliding_dates_presets exact_date="#sliding_dates_exact"
relative="#sliding_dates_from_today">
  <preset name="#sliding_dates_yesterday">
    <value>-1</value>
  </preset>
  <preset name="#sliding_dates_tomorrow">
    <value>1</value>
  </preset>
  <preset name="#sliding_dates_month_ago">
    <value>-30</value>
  </preset>
  <preset name="#sliding_dates_month_ahead">
    <value>30</value>
  </preset>
  <preset name="#sliding_dates_today">
    <value>0</value>
  </preset>
</sliding_dates_presets>
```

- 4 Change the corresponding resources in the Ln_package_res.xml file.

For more information, see ["Controlling the Synchronization Intervals That Display in the Synchronization Tab" on page 132](#).

Controlling the Fields That Display in a Filter

You can use the IsHidden property of the field in the siebel_meta_info.xml file to control the fields that are available in a filter. For more information, see ["Sharing a Calendar Entry, Contact, or To Do Item" on page 95](#).

To control the fields that display in a filter

- 1 Use an XML editor to open the siebel_meta_info.xml file.
For more information, see ["Files in the Customization Package" on page 355](#).
- 2 Locate the first instance of the tag that defines the field you must modify.

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

```
<field Name='Account Status' Label='Account Status' DataType='DTYPE_TEXT'
HasPicklist='yes' PicklistStatic='yes' PicklistCollectionType='ACCOUNT_STATUS'
PicklistTypeId='Picklist_Generic' IOElementName='AccountStatus' />
```

3 Do one of the following:

- Make the field not available in filter criteria. You add the following property to this tag:

`I sHi dden=' yes'`

- Make the field available in filter criteria. You add the following property to this tag:

`I sHi dden=' 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 Synchronization Time, Day, and Size

This topic describes how to control synchronization time, day, and size. It includes the following topics:

- [Overview of Controlling Synchronization Frequency on page 131](#)
- [Controlling the Synchronization Intervals That Display in the Synchronization Tab on page 132](#)
- [Controlling the Time and Day When Synchronizations Occur on page 133](#)
- [Controlling the Size and Type of Synchronized Records on page 135](#)
- [Synchronizing All Changes or Only Local Changes on page 136](#)
- [Controlling the Number of Records That Synchronize on page 137](#)
- [Configuring Siebel CRM Desktop to Disregard Erroneous Data That Users Modify on page 138](#)
- [Controlling the Number and Size of Batch Requests on page 139](#)

Overview of Controlling Synchronization Frequency

Application settings that are related to synchronization frequency determine how often and the kind of data Siebel CRM Desktop synchronizes. The user can specify frequency or you can configure it:

- The user can use the Synchronization tab of the Options dialog box to specify the interval that CRM Desktop uses to automatically start a synchronization. The user can double-click the CRM Desktop icon in the system tray or choose the Synchronize Now option from the options menu. The user can choose to synchronize all changes or only local changes. For more information, see [“Synchronizing All Changes or Only Local Changes” on page 136](#).
- You can configure the application metadata to determine how often CRM Desktop synchronizes each object. You can configure 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 the description about the frequency tag in [“Files in the Customization Package” on page 355](#).

For more information about controlling synchronization frequency, see [“Controlling Synchronization Time, Day, and Size” on page 131](#).

Controlling the Synchronization Intervals That Display in the Synchronization Tab

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

To control the synchronization intervals that display in the Synchronization tab

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

For more information, see [“Files in the Customization Package” on page 355](#).

- 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:

- The following code describes an interval in minutes and a resource string that you add in [Step 4](#):

```
10 = page-sync-periods-10-minutes
```

- 3 Save and then close the platform_configuration.xml file.
- 4 Use an XML editor to open the Ln_package_res.xml file 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>
```

For more information, see [“Controlling the Predefined Synchronization Intervals” on page 133](#).

- 5 (Optional) Set the default synchronization mode.

For more information, see [“Synchronizing All Changes or Only Local Changes” on page 136](#).

- 6 Republish the customization package on the Siebel Server.

For more information, see [“Republishing Customization Packages” on page 80](#).

Controlling the Predefined Synchronization Intervals

The following values come predefined for a Synchronize All Changes session:

- 60 minutes (One time for each hour)
- 720 minutes (Two times for each day)
- 1440 minutes (One time for each)
- 10080 minutes (One time for week)

The following values come predefined for a Synchronize Local Changes session:

- 10 minutes (Every ten minutes)
- 20 minutes (Every twenty minutes)
- 30 minutes (Every thirty minutes)
- 60 minutes (Every sixty minutes)

Resource strings in the Ln_package_res.xml file determine the predefined synchronization intervals. You can customize these intervals. For example, the following resource string controls synchronization to occur every 10 minutes:

```
<str key="page-sync-periods-10-minutes">Every 10 minutes</str>
```

How Siebel CRM Desktop Automatically Synchronizes If it Is Offline

If Siebel CRM Desktop is offline during a scheduled synchronization, then it does not run synchronization automatically. It runs the next synchronization according to the predefined schedule. For example, assume the automatic full synchronization is scheduled to run one time for each day. If the next synchronization occurs at 8:00 P.M., and if CRM Desktop is offline at 8:00 P.M., then it does not run the synchronization. The next automatic full synchronization occurs at 8:00 P.M. on the next day.

Controlling the Time and Day When Synchronizations Occur

You can control the time of day and the day when Siebel CRM Desktop synchronizes to manage the load that synchronization puts on the Siebel Server. For example, to avoid an overloaded server, you can delay synchronizations that might normally occur at 9:00 A.M. on a Monday morning after a weekend sales conference to a later time.

CRM Desktop delays synchronization for the number of milliseconds that you specify. It adds this delay before it sends each server request to the Siebel Server. The delays are summative. For example, if 500 requests exist, and if the delay is 1200 milliseconds, then it delays the synchronization for 10 minutes.

To control the time and day when synchronizations occur

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

For more information, see [“Files in the Customization Package” on page 355](#) and [“Customizing Meta Information” on page 157](#).

- 2 Add a tag named delays_schedule.

- 3 In the delays_schedule tag, set the following attribute to meet your scheduling requirements:

request_delay DaySpec

For more information, see [“Coding the Delays Schedule Tag” on page 134](#).

- 4 Repeat [Step 3](#), as necessary.

Coding the Delays Schedule Tag

You must use the following format if you code the delays_schedule tag:

- To specify a day of the week, use MON, TUE, WED, THU, FRI, SAT, or SUN.
- To specify a day, use one of the following formats:

- `yyyy/mm/dd`

- `mm/dd`

- `dd`

where:

- `yyyy/mm/dd` is the year, month, and day.

- You can use the DaySpec attribute to specify a delay in milliseconds.

If you include the day and date, then the date takes precedence. In the following example, Siebel CRM Desktop uses 2011/12/16. It does not use MON:

```
<request_delay DaySpec="MON" "2011/12/16" StartTime="12:00:00" EndTime="13:00:00"
DelayMsecs="6000" />
```

Example Code That Controls the Time and Day When Siebel CRM Desktop Synchronizes

The following code controls the time and day when Siebel CRM Desktop synchronizes:

```
<delays_schedule>
  <request_delay DaySpec="SUN" StartTime="10:22:17" EndTime="16:34:07" DelayMsecs="6000" />
  <request_delay DaySpec="MON" StartTime="12:00:00" EndTime="13:00:00" DelayMsecs="6000" />
  <request_delay DaySpec="TUE" StartTime="12:00:00" EndTime="13:00:00" DelayMsecs="6000" />
  <request_delay DaySpec="WED" StartTime="12:00:00" EndTime="13:00:00" DelayMsecs="6000" />
  <request_delay DaySpec="THU" StartTime="12:00:00" EndTime="17:00:00" DelayMsecs="6000" />
  <request_delay DaySpec="FRI" StartTime="12:00:00" EndTime="13:00:00" DelayMsecs="6000" />
  <request_delay DaySpec="SAT" StartTime="12:00:00" EndTime="13:00:00" DelayMsecs="6000" />
  <request_delay DaySpec="2009/09/01" StartTime="12:00:00" EndTime="13:00:00" DelayMsecs="6000" />
  <request_delay DaySpec="2009/12/31" StartTime="12:00:00" EndTime="13:00:00" DelayMsecs="6000" />
  <request_delay DaySpec="05/07" StartTime="12:00:00" EndTime="13:00:00" DelayMsecs="6000" />
</delays_schedule>
```

Controlling the Size and Type of Synchronized Records

You can control the size and type of records that Siebel CRM Desktop synchronizes. CRM Desktop comes with one predefined filter named Default filter. If you use this filter, then CRM Desktop downloads the following records from the Siebel Server to IBM Notes:

- All accounts, contacts, and opportunities that the user can view
- All activities that the user owns
- 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

CRM Desktop downloads any child record that is related to a parent record that the user can view. This configuration 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 that the user can view

If the user attempts to open the detail form for a record that CRM Desktop does not allow the user to view, then it displays a read-only version of the detail form. This read-only form includes only some of the details.

To control the size and type of synchronized files

- 1 Configure the Filter Presets tag of the Ln_connector_configuration.xml file.

For more information, see [“Filter Presets Tag of the Connector Configuration File” on page 388](#), and [“Example Code That Sets the Size and Type of Field” on page 388](#).

- 2 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. The settings you configure in the Filter Presets tag of the Ln_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 this limit to 9 MB. In this situation, CRM Desktop restricts the file size to 5 MB.

Synchronizing All Changes or Only Local Changes

A user can right-click the CRM Desktop icon in the system tray and then choose one of the following menu items to manually start a synchronization:

- Synchronize All Changes
- Synchronize Local Changes

You can enable or disable Synchronize Local Changes. This menu item allows the user to synchronize only local changes to the Siebel Server. It does not synchronize all changes. The installer enables it, by default. CRM Desktop synchronizes only new and modified records from IBM Notes to the Siebel Server during a Synchronize Local Changes session. It compares each modified record to the corresponding record that resides on the Siebel Server. Collisions might occur during this synchronization. For more information, see [“Resolving Synchronization Conflicts” on page 148](#).

To synchronize all changes or only local changes

- 1 Open the Windows Registry and then locate the following key:

HKEY_CURRENT_USER\Software\Oracle\CRM Desktopfor IBM Notes

For more information, see [“Using the Windows Registry to Control Siebel CRM Desktop” on page 103](#).

- 2 Right-click the EnablePeriodicSyncUpstream entry and then click Modify.
- 3 Enter one of the following values in the Value Data window of the Edit DWORD dialog box:
 - 1. Enables Synchronize Local Changes. The default value is 1.
 - 0. Disables Synchronize Local Changes.
- 4 Restart IBM Notes.

If you disable Synchronize Local Changes in [Step 3](#), then CRM Desktop removes the Synchronize Local Changes menu item from the menu that it displays if the user right-clicks the CRM Desktop icon in the system tray. It also removes the Synchronize Recent Changes to Server Automatically check box and slider from the Synchronization tab on the CRM Desktop - Options dialog box.

Note the following:

- If a full synchronization and a local synchronization are scheduled to automatically run at the same time, then CRM Desktop runs the full synchronization and skips the local changes synchronization until the next time the local synchronization is scheduled to run. *Local synchronization* is a type of synchronization that synchronizes only local changes.
- If the full synchronization is scheduled to occur more frequently than the value that the UpstreamSyncMinThreshold parameter contains, then CRM Desktop disables local synchronization. The default value for the UpstreamSyncMinThreshold parameter is 600000 milliseconds, which is 10 minutes.

For more information, see [“Controlling the Predefined Synchronization Intervals” on page 133](#).

Controlling the Number of Records That Synchronize

To produce the most desirable synchronization performance and scalability, and to maintain acceptable IBM Notes performance when the user works with Siebel CRM data, it is recommended that Siebel CRM Desktop synchronize no more than 10,000 records for a single user. If it synchronizes more than 10,000 records, then the following undesirable results might occur:

- Longer synchronization times
- More server resources required to support user synchronization sessions
- Slower IBM Notes performance when working with Siebel CRM data

To control the number of records that synchronize

- 1 With administrator privileges, log in to a Siebel CRM 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 value for the Maximum Page Size parameter to the maximum number of records that CRM Desktop can synchronize for a single user.

If CRM Desktop attempts to synchronize more records at run-time than the value you set, then it returns an error message to IBM Notes that indicates that the number of records requested is too large. For more information, see [“Resolving Exceeded Row Size Problems” on page 124](#).

Configuring Siebel CRM Desktop to Disregard Erroneous Data That Users Modify

You can configure Siebel CRM Desktop to synchronize an object in only one direction. Using one-way synchronization can be useful to restore object types that the user modifies or removes in IBM Notes or to not allow the user to modify an object type. For example, employees or positions. One-way synchronization can also reduce the time required to synchronize. The synchronization engine can detect these modifications and refresh the object in IBM Notes without causing a collision. CRM Desktop does the following work for each object that it synchronizes in only one direction:

- Does not create a collision
- Does not display a delete confirmation
- Does not start a job on the Siebel Server

This configuration allows Siebel CRM Desktop to disregard erroneous data that the user enters or erroneous modifications that users make to data so that it does not synchronize these modifications to the Siebel Server. To do this, you configure an object to synchronize in only one direction. The example in this topic configures the Employee object to synchronize in only one direction.

A user might use some IBM Notes features that compromise the validation rules that CRM Desktop uses. For example, CRM Desktop cannot control how the user uses the native All Fields tab that IBM Notes displays on IBM Notes objects. This tab is a predefined IBM Notes feature that CRM Desktop cannot disable or intercept. A user might open a Contact record, click the All Fields tab, remove the values from the First Name and Last Name fields, and then save this record even though CRM Desktop requires these names.

For more information, see ["Resolving Synchronization Conflicts" on page 148](#).

To configure Siebel CRM Desktop to disregard erroneous data that users modify

- 1 Use an XML editor to open the Ln_connector_configuration.xml file.

For more information, see ["Files in the Customization Package" on page 355](#).

- 2 Locate the `type` tag of the object that you must configure for one-way synchronization.

For example:

```
type id="Employee"
```

- 3 Locate the `synchronizer` tag of the `type` tag you located in [Step 2](#).
- 4 Set the `read_only` attribute of the `type` tag you located in [Step 3](#) to `true`.

For example:

```
<type id="Employee">
  <view label="#obj_employee" label_plural="#obj_employee_plural"
small_icon="type_image: User: 16" normal_icon="type_image: User: 24"
large_icon="type_image: User: 48" suppress_sync_ui="true">
  </view>
  <synchronizer name_format="[: (First Name):] :[: (Last Name):]"
frequency="604800" threshold="0" read_only="true">
```

```

<links>
  <link>Primary Organization Id</link>
  <link>Primary Position Id</link>
</links>
</synchronizer>
</type>

```

To configure an object type to synchronize in only one direction, you set it to read only in IBM Notes. CRM Desktop synchronizes a read-only object in only one direction from the Siebel Server to the client.

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

For more information, see [“Republishing Customization Packages” on page 80](#).

Controlling the Number and Size of Batch Requests

Siebel CRM Desktop sends requests to the Siebel Server in batches. One Web Service call is one batch. Each batch includes multiple commands. Each command can request multiple IDs. To avoid overloading the server, you can specify the number of IDs that CRM Desktop requests for each command and the number commands that each batch contains.

For example, assume the following occurs:

- CRM Desktop must get 300 accounts
- A batch contains only one command
- One command requests the IDs for all 300 accounts

The server fails in this situation. To avoid this problem, you can reduce the number of commands that each batch contains and the number of IDs that each command requests. CRM Desktop sets these values to 50, by default. If CRM Desktop uses this default value to process the 300 accounts, then it creates six separate commands where each command requests 50 account IDs. It creates one batch that includes these six commands.

To control the number and size of batch requests

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

For more information, see [“Files in the Customization Package” on page 355](#) and [“Customizing Meta Information” on page 157](#).

- 2 Create or find the existing section under the root tag that contains the following name:

```
common_settings
```

For more information, see the documentation in the siebel_meta_info.xsd file available in Article ID 1502099.1 on My Oracle Support.

- 3 Set the following subtags:

- **max_commands_per_batch.** Defines the maximum number of commands that each batch contains. The default value is 50.

- **max_ids_per_command.** Defines the maximum number of IDs that the command requests. The default value is 50.

Controlling Other Configurations That Affect Synchronization

This topic describes how to control other configurations that affect synchronization. It includes the following topic:

- [Configuring How CRM Desktop Gets Updates That Occur During Synchronization on page 140](#)
- [Configuring CRM Desktop to Synchronize Private Activities on page 141](#)
- [Controlling the View Mode During Synchronization According to Object Type on page 143](#)
- [Controlling How Siebel CRM Desktop Deletes Records During Synchronization on page 145](#)
- [Resolving Synchronization Conflicts on page 148](#)
- [Resolving Synchronization Conflicts on page 148](#)

Configuring How CRM Desktop Gets Updates That Occur During Synchronization

You can configure how Siebel CRM Desktop gets object updates from the Siebel Server that occur during a synchronization session. You can use this configuration to update a field that originates on the Siebel Server, such as an object Id or an automatically generated number. You can also use it to update a field Id that is involved with a calculated value that EAI (Enterprise Application Integration) inserts or updates in the Siebel database during synchronization.

The example in this topic adds a new Siebel ID field to the account object in the mapping schema. Siebel CRM Desktop uses this field internally to store the Siebel object Id in a text format. It updates this field during synchronization.

To configure how CRM Desktop gets updates that occur during synchronization

- 1 Use an XML editor open the siebel_meta_info.xml file.
- 2 Locate the tag of the object that represents the field that CRM Desktop must update during synchronization.

For example, locate the following object tag:

```
Typel d="Account"
```

- 3 Add the following tag to the tag that you located in [Step 2](#):

```
<fi el d Name=' Siebel ID' Label = ' Siebel ID' DataType=' DTYPE_TEXT' BackUpd=' any'  
IsFi l terabl e=' no' IsCal cul ated=' yes' Formul a=' [: (Id): ]' />
```

where:

- BackUpd is the attribute that allows CRM Desktop to get updates that occur during synchronization. You can set it to one of the following values:
- **insert**. Updates values that occur during an insert.
- **update**. Updates values that occur during an update.
- **any**. Updates values that occur during an insert or an update.

During synchronization, CRM Desktop inserts the value that the Id field contains into the Siebel ID field.

- 4 Save and then close the siebel_meta_info.xml file.
- 5 Open the Ln_siebel_basic_mapping.xml file.
- 6 Add the following code anywhere in the file:

```
<field id="Siebel ID">
  <reader>
    <lotus_std>
      <lotus_field id="sbl Siebel ID"></lotus_field>
      <convertor>
        <string/>
      </convertor>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="sbl Siebel ID"></lotus_field>
      <convertor>
        <string/>
      </convertor>
    </lotus_std>
  </writer>
</field>
```

Make sure that the value you use for the `lotus_field id` attribute is identical to the value that you use for the `field Name` attribute in [Step 3](#) except that the `user_field id` attribute includes the `sbl` prefix.

- 7 Save your changes and then republish the customization package.
- For more information, see ["Republishing Customization Packages" on page 80](#).

Configuring CRM Desktop to Synchronize Private Activities

This topic describes how to configure Siebel CRM Desktop to synchronize private activities that the user creates in IBM Notes.

To configure CRM Desktop to synchronize private activities

- 1 Use an XML editor open the siebel_meta_info.xml file and then locate the following object:

```
<object TypeId="Action"
```

- 2 Remove the following code from the object you located in [Step 1](#):

```
<master_filter_expr>  
<![CDATA[  
[Private] IS NULL OR [Private] = 'N'  
]]>  
</master_filter_expr>
```

This filter prevents CRM Desktop from synchronizing private activities.

- 3 Save your changes and then close the siebel_meta_info.xml file.
- 4 Open IBM Domino Designer, and then open the SBL.Forms script library.
For more information, see ["Opening IBM Domino Designer" on page 111](#).
- 5 Locate the FormPIMEx class and then open the InitValidators procedure that resides in this class.
- 6 Remove the following code:

```
Set ValidationCallBack = New ValidationPIMSPri vate(m_ActivityProcessor)  
vld.ValidateCustom Fields, MSG_CANT_SYNC_PRIVATE, ValidationCallBack, ""
```

- 7 Save the changes you made in the SBL.Forms script library.
- 8 Open the SBL.ActivityHandlers script library.
- 9 Locate the ActivityProcessor class, and then open the SkipApptDocument procedure that resides in this class.
- 10 Locate the following code:

```
If Not doc Is Nothing Then  
SkipApptDocument = doc.IsPrivate Or _  
doc.TypeID = AP_EVENT_TYPEID And _  
doc.Property(AP_APPOINTMENT_TYPE_FIEL D) <> AP_APPOINTMENT_TYPE_APPT And _  
doc.Property(AP_APPOINTMENT_TYPE_FIEL D) <> AP_APPOINTMENT_TYPE_MEETING  
End If
```

- 11 Modify the SkipApptDocument line that resides in the code you located in [Step 10](#). You replace IsPrivate Or with the following bolded code:

```
If Not doc Is Nothing Then  
SkipApptDocument = doc.TypeID = AP_EVENT_TYPEID And _  
doc.Property(AP_APPOINTMENT_TYPE_FIEL D) <> AP_APPOINTMENT_TYPE_APPT And _  
doc.Property(AP_APPOINTMENT_TYPE_FIEL D) <> AP_APPOINTMENT_TYPE_MEETING  
End If
```

- 12 Save the changes you made in the SBL.ActivityHandlers script library.
- 13 Save and test your changes, and then republish the customization package.

Controlling the View Mode During Synchronization According to Object Type

This topic describes how to control the view mode that Siebel CRM Desktop uses during synchronization for an object type. The view modes that are configured in the client application metadata affect 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 CRM Desktop passes to the EAI Siebel Adapter business service during synchronization. It 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. The default configuration for CRM Desktop specifies that the opportunity synchronization object must 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 29](#).

To control the view mode during synchronization according to object type

- 1 Use an XML editor open the siebel_meta_info.xml file.
- 2 Locate the `type` tag of the object that must use a view mode during synchronization.

For example, locate the following tag:

```
object TypeId='Account.Account_Note'
```

- 3 Locate the `viewmodes` subelement of the `type` tag you located in [Step 2](#).
- 4 Set the `viewmodes` subelement using the following format:

```
viewmodes view_mode_context1="view_mode_value"
view_mode_context2="view_mode_value" view_mode_context3="view_mode_value"
```

where:

- `view_mode_context` is set to one of the following values:
 - **General.** Requests server records of synchronized object types that match the user synchronization filters with master filter restrictions applied. If you specify this value, then it overrides the value in the viewmode attribute. For more information, see [Step 2 on page 106](#).
 - **Dedup.** Detects duplicate records when CRM Desktop must add records to IBM Notes or to the server database. If you do not specify this value, then it sets the value for the Deduplication view mode to the value that you set for the General view mode. For more information, see ["Resolving Synchronization Conflicts" on page 148](#).
 - **QBID.** (Query By Id) Requests objects that CRM Desktop must synchronize to IBM Notes to maintain referential integrity because this object is related to a synchronized record. It requests this object even if the object does not match a user synchronization filter. You must specify a Query by Id for each object type that CRM Desktop queries by Id. The default value is All.
- `view_mode_value` is set to one of the following values:

- All
- Sales Rep
- Personal
- Organization

For example:

```
viewmodes General="Organization" Dedup="All"
```

Example That Controls the View Mode During Synchronization According to Object Type

The following code controls the view mode that Siebel CRM Desktop uses during synchronization for the Account.Account_Note object type:

```
<object TypeId='Account.Account_Note' Label='#obj_account_note'  
LabelPlural='#obj_account_note_plural' EnableGetIdsBatching='true'  
IntObjName='CRMDesktopAccountIO' SiebMsgXmlElementName='AccountNote'  
SiebMsgXmlElementCollectionName='ListOfAccountNote' >  
  <viewmodes General="Organization" Dedup="All" />  
  <open_with_url_tmpl >  
    <![CDATA[  
      [: (protocol)]: //[: (hostname):][: (port):]/sales/:[: (lang):]/  
?SWECmd=GotoView&SWEView=Account+Note+View&SWERF=1&SWEHO=:[: (hostname):]&SWEBU=1&SWEAp  
pletO=Account+Entry+Appl et&SWERowIdO=:[: (parent_id):]&SWEAppl et1=Account+Note+Appl et&S  
WERowId1=:[: (own_id):]  
]]>  
  </open_with_url_tmpl >  
  <extra_command_options>  
    <option Name='PrimaryKey1M' Value='Id' />  
    <option Name='ForeignKey1M' Value='Account Id' />  
    <option Name='Cardinality' Value='1M' />  
    <option Name='ServerServiceVersion' Value='2' />  
  </extra_command_options>  
  <field Name='Account Id' Label='Account Id' DataType='DTYPE_ID' IsNullLabel='no'  
IsFilterable='no' IsRefObjId='yes' RefObjTypeId='Account' RefObjIsParent='yes'  
IsPartOfUserKey='yes' IOElemName='AccountId' />  
  <field Name='Conflict Id' Label='Conflict Id' DataType='DTYPE_ID'  
IsFilterable='no' IsHidden='yes' IOElemName='ConflictId' />  
  <field Name='Created' Label='#fld_account_account_note@created'  
DataType='DTYPE_DATETIME' IsPartOfUserKey='yes' IOElemName='Created' />  
  <field Name='Created By' Label='Created By' DataType='DTYPE_ID' IsFilterable='no'  
IsRefObjId='yes' RefObjTypeId='Employee' IOElemName='CreatedBy' />  
  <field Name='Created By Name' Label='#fld_account_account_note@created_by_name'  
DataType='DTYPE_TEXT' IOElemName='CreatedByName' />  
  <field Name='Created Date' Label='Created Date' DataType='DTYPE_UTCDATETIME'  
IsHidden='yes' IOElemName='CreatedDate' />  
  <field Name='DS Updated' Label='DS Updated' DataType='DTYPE_DATETIME'  
IsFilterable='no' IsHidden='yes' IsTimestamp='yes' IOElemName='DBLastUpd' />  
  <field Name='Id' Label='Id' IsPrimaryKey='yes' DataType='DTYPE_ID' IsFilterable='no'  
IsHidden='yes' IsPartOfUserKey='yes' IOElemName='Id' />  
  <field Name='Mod Id' Label='Mod Id' DataType='DTYPE_ID' IsFilterable='no'
```



```

Ishidden=' yes' IOElementName=' ModId' />
  <field Name=' Note' Label =' #fld_account_account_note@note' DataType=' DTYPE_NOTE'
IOElementName=' Note' />
  <field Name=' Note Type' Label =' #fld_account_account_note@note_type'
DataType=' DTYPE_TEXT' HasPicklist=' yes' PicklistIsStatic=' yes'
PicklistCollectionType=' FS_NOTE_TYPE' PicklistTypeId=' List_Of_Values'
IOElementName=' NoteType' />
  <field Name=' Private' Label =' Private' DataType=' DTYPE_BOOL' Ishidden=' yes'
IOElementName=' Private' />
  <field Name=' Updated' Label =' Updated' DataType=' DTYPE_DATETIME' Ishidden=' yes'
IOElementName=' Updated' />
  <field Name=' Updated By' Label =' Updated By' DataType=' DTYPE_ID' IsFilterable=' no'
Ishidden=' yes' IOElementName=' UpdatedBy' />
</object>

```

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 IBM Notes. If you enable this feature, then 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 IBM Notes, then CRM Desktop displays the Confirm Synchronization tab during synchronization. If the user confirms, then it 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 146](#).

To control how Siebel CRM Desktop deletes records during synchronization

- 1 Use an XML editor open the Ln_connector_configuration.xml file.
- 2 Configure CRM Desktop to display the Confirm Synchronization tab on the Synchronization Control Panel dialog box:
 - a Add the following code to the root tag of the Ln_connector_configuration.xml file:


```
<features deletion_confirmation_mode="enable"/>
```

 For more information, see [“Setting the Delete Confirmation Mode Attribute” on page 146](#).
 - b Specify the objects that 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 147](#).
- 3 Configure CRM Desktop to suppress display of the Confirm Synchronization tab in the Synchronization Control Panel dialog box. You add the following code to the root tag of the Ln_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 deleted determines if 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 CRM Desktop displays the Confirm Synchronization tab.
- If the user deletes only one or two accounts, then CRM Desktop does not display the Confirm Synchronization tab.

For more information, see ["Threshold That Siebel CRM Desktop Uses to Display the Confirm Synchronization Tab" on page 346](#).

Setting the Delete Confirmation Mode Attribute

You use the `deletion_confirmation_mode` attribute of the `Ln_connector_configuration.xml` file to control the Confirm Synchronization tab on the Synchronization Control Panel dialog box.

[Table 10](#) describes the values you can use for the `deletion_confirmation_mode` attribute.

Table 10. Values for the Delete Confirmation Mode Attribute

| Value | Description |
|--------------|--|
| suppress | Disables delete confirmation. Displays the Confirm Synchronization tab in the Synchronization Control Panel. |
| enable | Enables delete confirmation. Displays the Confirm Synchronization tab in the Synchronization Control Panel. Displays the Revert Deletions button and the Accept Deletions button. |
| revert_only | Displays the Confirm Synchronization tab in the Synchronization Control Panel but enables only the Revert Deletions button. Displays but does not enable the Accept Deletions button. This is the default setting. |
| user_confirm | Displays the Confirm Synchronization tab in the Synchronization Control Panel but displays only the Accept Deletions button. 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 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 Ln_connector_configuration.xml file.
- 2 Locate the object type that CRM Desktop must display in the Delete on Siebel list in the Confirm Synchronization tab list.

For example, for opportunities, you 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 147](#).

- 4 Repeat [Step 2](#) through [Step 3](#) for each type of object that CRM Desktop must display in the Delete on Siebel list.

Setting the Synchronizer Tag

The synchronizer tag in the Ln_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. This attribute is set to 5 in the following example. It causes 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 11 describes the values for the threshold attribute of the synchronizer tag.

Table 11. Values for the Threshold Attribute of the Synchronizer Tag

| Value | Description |
|--------------------------|---|
| 0 | Do not display delete confirmation for the object type. |
| 1 | Display delete confirmation for the object type. |
| Any value greater than 1 | <p>If you specify any value that is greater than one, then do the following:</p> <ul style="list-style-type: none"> ■ If the value you specify is greater than the number of deleted objects, then do not display delete confirmation for the object. ■ If the value you specify is less than or equal to the number of deleted objects, then display delete confirmation for the object. |

Resolving Synchronization Conflicts

This topic describes how to configure Siebel CRM Desktop to resolve synchronization conflicts. It includes the following topics:

- [Overview of Synchronization Conflicts on page 148](#)
- [Configuring Siebel CRM Desktop to Resolve Synchronization Conflicts on page 150](#)
- [Examples of Auto Resolver Rules on page 151](#)

Overview of Synchronization Conflicts

Auto Resolver is a Siebel CRM Desktop feature that allows you to configure CRM Desktop to automatically resolve a synchronization conflict instead of allowing the user to manually resolve this conflict. One of the following synchronization conflicts might occur when Siebel CRM Desktop synchronizes local data with data from the Siebel Server:

- Duplicating a record
- Simultaneously updating a record on the client and on the server
- Updating a record on the client and deleting this record on the server
- Updating a record on the server and deleting this record on the client

Consider the following items:

- **Activity processing.** Assume a user receives an event that CRM Desktop automatically shares. The user then downloads the same event from the Siebel Server during a synchronization. This situation might cause a *duplication conflict*, which is a type of conflict where two separate records include exactly the same information. You can configure the Auto Resolver so that CRM Desktop uses the record from the client database or from the Siebel Server database as the primary record, and then uses the data in this primary record to update the data in the nonprimary record.

- **Data update.** Assume a user updates a date in a field on the client, and that the Siebel Server updates this same date on the server. In this situation, an *update conflict* occurs during synchronization between the client and server databases.

This topic uses the following terms:

- *remote* identifies the Siebel Server.
- *local* identifies the client.

It is recommended that you configure autoreresolve for each individual field, and only when necessary. It is recommended that you do not configure Siebel CRM Desktop to resolve all conflicts. Sometimes the client will include the correct value and sometimes the Siebel Server will include the correct value. Using autoreresolve might result in Siebel CRM Desktop maintaining an incorrect value. If the client or server might include the correct value, then it is recommended that you allow the user to choose the correct value.

How the Auto Resolver Resolves Conflicts

The Auto Resolver Manager (ar_manager) includes a function that it assigns when an on_conflict event occurs. If this event occurs, then the synchronizer indicates the conflict and then provides information about this conflict. This conflict information might include the following items:

- Local object Id
- Remote object Id
- Set of local fields
- Set of remote fields
- Conflict type as a duplication conflict or a general conflict
- And so on

The Auto Resolver does the following work to resolve a conflict:

- 1 Examines the first rule that the autoreresolver.js file contains.
- 2 If the rule that it examines in [Step 1](#):
 - **Is applicable for the conflict.** The Auto Resolver attempts to apply this rule to this conflict. If this rule:
 - **Resolves the conflict.** The Auto Resolver determines if another conflict exists. If it finds another conflict, then it repeats [Step 1](#) and [Step 2](#) until it processes all conflicts. If it does not find another conflict, then it quits this process.
 - **Does not resolve the conflict.** The Auto Resolver examines the next rule that the autoreresolver.js file contains. It continues to process these rules sequentially in the order that they occur in the autoreresolver.js file until it resolves the conflict or until it processes all rules.
 - **Is not applicable for the conflict.** The Auto Resolver examines the next rule that the autoreresolver.js file contains.

- 3 If the Auto Resolver applies all rules but the conflict remains, then this conflict remains unresolved, and the Auto Resolver displays it in the Check Issues tab of the Control Panel dialog box of the CRM Desktop add-in.

Configuring Siebel CRM Desktop to Resolve Synchronization Conflicts

This topic describes how to configure Siebel CRM Desktop to resolve synchronization conflicts.

To configure Siebel CRM Desktop to resolve synchronization conflicts

- 1 Use a JavaScript editor to open the `autoresolver.js` file.
- 2 Navigate to the following section:

```
//Opportunity
```

- 3 Add the following code:

```
ar_manager.add_rule({ type: "object_type", field: "field_ID", resolution_fn:
ar_helpers.resolution_resolve_to("source", boolean) });
```

where:

- **ar_manager**. Is the Auto Resolver manager. It is an interface to the C++ code that CRM Desktop runs to resolve the conflict.
- **add_rule**. Is the C++ method that `ar_manager` sends to the C++ code.
- **type**. Identifies the object type where CRM Desktop applies the rule.
- **field**. Identifies the field that contains the data that CRM Desktop uses to resolve the conflict.
- **resolution_fn**. Identifies the rule function that CRM Desktop passes to the C++ code. This function returns a string value that indicates if the Auto Resolver resolved the conflict.
- **resolution_resolve_to**. Defines the rule. The rule is configured in the `autoresolve_helpers.js` file. It includes the following values:
 - `source` is remote or local, where remote is the Siebel Server and local is the client. It identifies the source that CRM Desktop uses to resolve the conflict.
 - `boolean` is true or false.

For example:

```
ar_manager.add_rule({type: "Opportunity", field: "Account Id ", resolution_fn:
ar_helpers.resolution_resolve_to("local", true)});
```

In this example, if a synchronization conflict exists in the Account field of the Opportunity form, then this rule uses the value in the Account field from the client database as the permanent value only if the `resolution_fn` returns the following value:

Local

For example:

```
ar_manager.add_rule({ type: "Opportunity", field: "Account Id", resolution_fn:
ar_helpers.resolution_resolve_to("Local", false) });
```

In this example, if no values exists for this field on the client, then the Auto Resolver does not apply a resolution.

For another example, you add the following code to the //Opportunity section:

```
ar_manager.add_rule({ type: "Action", field: "Email To Line", resolution_fn:
ar_helpers.resolution_resolve_to("Local", false) });
```

Examples of Auto Resolver Rules

This topic describes some of the predefined rules that Auto Resolver uses. You can use these rules as a guide when you create your custom rules. The `autoresolver.js` file contains these predefined rules. You must not modify any predefined rule.

Rule That Resolves Association Conflicts

The following predefined rule resolves association conflicts:

```
/** ASSOCIATION AUTORESOLVER */
```

```
ar_manager.add_rule(new ar_helpers.associations_resolve_rule(ar_ctx))
```

where:

- `ar_helpers.associations_resolve_rule(ar_ctx)` identifies the code that CRM Desktop uses for the rule.
- `ar_ctx` identifies the list of the required objects. For example:

```
var ar_ctx = {
  "util": util,
  "conflicts_manager": conflicts_manager,
  "application": application,
  "data_model": business_logic.create_siebel_meta_scheme2()
```

where:

- `util` is a C++ object that contains a number of methods.
- `conflicts_manager` is the object that CRM Desktop creates in the interface to the C++ code that the Auto Resolver uses.
- `application` is a C++ object that contains a number of methods.
- `data_model` is a C++ object that contains a number of methods.
- You cannot modify a reference to any of these C++ objects.

Rule That Resolves Conflicts for Objects Types

The following predefined rule resolves conflicts for certain object types regardless of field values:

```
/** TYPE RULES */

ar_manager.add_rule({ types: ["Action", "Account", "Contact", "Opportunity"],
  resolution_fn: ar_helpers.condition_update_delete_conflict_stopper });
```

where:

- `types: ["Action", "Account", "Contact", "Opportunity"]` is a list of object types where CRM Desktop applies the rule.
- `condition_update_delete_conflict_stopper` is the rule that CRM Desktop runs. The `autoresolve_helpers.js` file contains this rule.

Rule That Resolves Conflicts for Fields

The following predefined rule resolves conflicts for certain object types, including field values:

```
/** FIELD RULES */
// Not synchronized
ar_manager.add_rule({ types: ["Contact", "Opportunity", "Account"], field: "Primary
Position Id", resolution_fn:
  ar_helpers.resolution_resolve_not_synced_field_to("remote", false) });
```

where:

- `types: ["Contact", "Opportunity", "Account"]` is a list of object types where CRM Desktop applies the rule
- `field: "Primary Position Id"` identifies the field that contains the data that CRM Desktop uses to resolve the conflict.
- `resolution_resolve_not_synced_field_to` is a rule that the Auto Resolver applies only for an object that CRM Desktop has not synchronized to the client. This situation can occur in a deduplication conflict.

The Auto Resolver applies this rule only for an object that it has not synchronized to the client. This situation can occur in a deduplication conflict.

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

- [Overview of Customizing Siebel CRM Desktop on page 153](#)
- [Customizing Field Behavior on page 164](#)
- [Customizing UI Behavior on page 179](#)
- [Validating the Data That Users Enter on page 190](#)
- [Process of Adding Custom Objects on page 196](#)
- [Removing Customizations on page 207](#)
- [Troubleshooting Problems That Occur When You Customize Siebel CRM Desktop on page 208](#)

Overview of Customizing Siebel CRM Desktop

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

- [Customizing Field Mapping on page 154](#)
- [Customizing Synchronization on page 154](#)
- [Customizing Forms on page 155](#)
- [Customizing Dialog Boxes on page 155](#)
- [Customizing Views on page 156](#)
- [Customizing the SalesBook Control on page 157](#)
- [Customizing Meta Information on page 157](#)
- [Customizations That Oracle Does Not Support on page 158](#)
- [Preparing the Development Environment on page 161](#)
- [Using Siebel Tools on page 163](#)

You can customize Siebel CRM Desktop to do 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.

CRM Desktop can synchronize this information with Siebel CRM data in IBM Notes. You can also make the data available to support offline usage if the user possesses limited or no connection to the Internet. You can create an interface where CRM Desktop stores the information the user requires to complete a business process. This configuration allows the user to work in a single application rather than navigating between multiple applications.

CAUTION: XML files, DXL files, and JavaScript files contain predefined configuration information that is critical to Siebel CRM Desktop operations. If you modify any of these files, 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.

For more information, see the following topics:

- [Files in the Customization Package on page 355](#)
- [Metadata That Siebel CRM Desktop Uses on page 32](#)

Customizing How Siebel CRM Desktop Processes Objects

You can use Siebel CRM Desktop functionality, such as deduplication or data validation, to add custom logic for object processing in IBM Notes. You can also use LotusScript to create your own custom logic. You can use LotusScript to create, delete, or modify objects. You can migrate your Siebel CRM data to IBM Notes, including logic that supports a business process.

You can use a standard CRM Desktop form to display Siebel CRM objects, or you can create a completely new and custom form. A custom form can include native IBM Notes controls, such as a text box. The user can use IBM Notes embedded folders on custom forms to display relationships between Siebel CRM objects. These views are fully configurable and support the functionality of native IBM Notes views.

For more information, see [Resolving Synchronization Conflicts on page 148](#).

Customizing Field Mapping

The Ln_siebel_basic_mapping.xml file describes objects you can add to IBM Notes. It defines mapping between a Siebel CRM field and the IBM Notes field. You can also extend a set of fields that native IBM Notes objects reference. For more information, see [“XML Code That Maps a Field” on page 381](#).

Customizing Synchronization

The Ln_connector_configuration.xml file identifies the objects that Siebel CRM Desktop synchronizes. It includes synchronization settings that affect the following types of filters:

Deduplication filters. It uses special criteria that the Ln_connector_configuration.xml file describes to determine if an item already exists in the Siebel database when Siebel CRM Desktop synchronizes an item from IBM Notes to the Siebel database. For more information, see [“How Siebel CRM Desktop Avoids Duplicate Data” on page 73](#) and [“Resolving Synchronization Conflicts” on page 148](#).

- **Preset filters.** Defines preset filters for a custom synchronization. These presets help the user to synchronize only the data that the user requires. You can configure any filter preset as the default.

For more information, see [“XML Code That Customizes Synchronization” on page 384](#).

Customizing Forms

You can use IBM Domino Designer to customize the IBM Notes forms that Siebel CRM Desktop uses. All forms modifications are exported to the corresponding .dxl files. Each .dxl file is named according to the following mask: SBL.Form.<Form Name>.dxl, where <Form Name> is the name of the form. For example, the design of an Account form is defined in the SBL.Form.Account.dxl file. For information about how to set up this capability, see [“Preparing the Development Environment” on page 161](#)

Validation Rules You Can Configure for Custom Forms

The SBL.Forms script library includes a description of the validation rules that Siebel CRM Desktop uses for the object that it displays on any form. You can configure 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 this format. Validation uses LotusScript to configure a wide variety of validation.

Business Logic That You Can Configure for Custom Forms

You can use the SBL.Forms and SBL.BusinessLogic script libraries to implement custom business logic.

Customizing Dialog Boxes

You can customize the following behaviors of the MVG dialog box:

- 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 it uses to display the Primary record. The following is the default format:

position (name)

For example, District Manager 1 (Wasaka Takuda). You can specify the fields that CRM Desktop displays and the order it uses to display them.

- Fields that it displays in the IBM Notes view that represent the associations you create. You can configure CRM Desktop to display only the record attributes. For example, if it stores the Employee name in the position record, then it can display only the Employee name for the position.

- The user permissions. The customization package uses security validation rules to describe the user permissions that work with the MVG. For example, if the user is not the primary user, then the user cannot delete users from the collection because CRM Desktop turns off the delete button for any user who is not a primary user.
- Behavior of a lookup control. A lookup control searches through the File As field of associated objects to search for a record.
- 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.

For more information, see ["Customizing Picklists" on page 209](#).

Customizing Views

You can use IBM Domino Designer to customize the IBM Notes views that CRM Desktop uses. All views modifications are exported to corresponding .dxl files. Each .dxl file is named according to the following mask: SBL.View.<View Name>.dxl, where <View Name> is the name of the view. For example, the design of an Accounts view is defined in the SBL.View.Accounts.dxl file.

Hiding Custom Views

You can use IBM Domino Designer to hide a custom top-level view from IBM Notes that Siebel CRM Desktop uses. The following procedure includes an example of hiding an Opportunities view.

Hiding custom views

- 1 Import the SBL.View.Opportunities.dxl file into IBM Notes using CRM Dev Utils.
- 2 Open the PAB database that contains the imported design elements in IBM Domino Designer.
- 3 Navigate to the Views list and locate the Opportunities view.
- 4 Open the Found view and then open the Found view properties.
- 5 Place the Name field value into parentheses. For example, (Opportunities).
- 6 Save and close the modified view.
- 7 Navigate to the tools folder.
- 8 In a text editor, open the export_import_config.xml file.
- 9 Locate the instructions for exporting or importing the Opportunities view. For the Opportunities view:

```
<file name="SBL.View.Opportunities.dxl">
<notes type="view_folder">
<note>Opportunities</note>
</notes>
</file>
```

- 10 Modify the name of note design element in IBM Domino Designer:

```
<file name="SBL.View.Opportunities.dxl">
<notes type="view_folder">
<note>(Opportunities)</note>
</notes></file>
```

- 11 Save and close the export_import_config.xml file.
- 12 Export the design elements to the .dxl file using CRM Dev Utils.
- 13 Publish the modified package on the Siebel server.
- 14 Obtain and apply the package and verify your changes.

To remove a custom top-level view from IBM Notes

- 1 Navigate to the package folder.
- 2 Open the dxl_config.xml file.
- 3 Locate the instructions for importing the view you want to remove. For Opportunities view:

```
<dxl path="SBL.View.Opportunities.dxl"/>
```
- 4 Comment out the instructions for importing the view.
- 5 Publish the modified package on the Siebel server.
- 6 Obtain and apply the package and verify your changes.

Customizing the SalesBook Control

The SalesBook control is a picklist that is native in IBM Notes. It defines references between objects that CRM Desktop uses in lookup controls. You can set filters for these objects. For more information, see ["Process of Creating Dynamic Picklists That Use a SalesBook Control" on page 247](#).

Customizing Meta Information

The siebel_meta_info.xml file includes 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 CRM Desktop uses to build or parse a Siebel message. It uses information about the relations between objects from the file for the Siebel message.

- The definition of each object that CRM Desktop supports. This definition includes a unique name, an XML element, and an XML collection element that CRM Desktop uses in a Siebel message.

Every object field includes a name, a Siebel data type, and an XML element. CRM Desktop uses this information in a Siebel message to display values and filters for that field.

For more information, see the documentation in the `siebel_meta_info.xsd` file available in Article ID 1502099.1 on My Oracle Support.

Customizations That Oracle Does Not Support

This topic describes customizations that Oracle does not support.

Files That You Must Not Modify

Table 12 describes the files that implement predefined functionality. You must not modify these files under any circumstances.

Table 12. Files You Must Not Modify

| File | Description |
|--|--|
| <code>autoresolve_helpers.js</code> | Includes helpers that the <code>autoresolver.js</code> file uses. |
| <code>data_model.js</code> | Includes logic for handling relations between objects and joint fields. The <code>business_logic.js</code> file initializes this data model. |
| <code>recurrence_processing.js</code> | Includes transformation functions that the activity processor uses for repeating patterns that occur between Siebel CRM and IBM Notes. |
| <code>SD3.Lib.Interfaces.dxl</code> | Script library that includes base interfaces. |
| <code>SD3.Lib.Utills.dxl</code> | Script library that includes common utility classes and functions. |
| <code>SD3.Lib.Errors.dxl</code> | Script library that includes classes and functions for error handling. |
| <code>SD3.Lib.Strings.dxl</code> | Script library that includes common string constants. |
| <code>SD3.Lib.RegistryHelpers.dxl</code> | Script library that includes a class for working with the registry. |
| <code>SD3.Lib.Constants.dxl</code> | Script library that includes common constants. |
| <code>SD3.Lib.MQ.dxl</code> | Script library that includes the interprocess message queue implementation. |
| <code>SD3.Lib.BMProvider.dxl</code> | Script library that includes classes for reading the <code>basic_mapping.xml</code> file. |
| <code>SD3.Lib.Tools.2.dxl</code> | Script library that includes core wrapper classes for the current session, database, documents, arrays, lists, and so on. This script library also contains some common helper function for logging, getting documents from context, comparing date or time values, and so on. |

Table 12. Files You Must Not Modify

| File | Description |
|------------------------------------|--|
| SD3.Lib.HandlerHelpers.dxl | Script library that includes helper classes and functions for handling common events. For example, creating, saving, and removing documents. |
| SD3.Lib.DataModel.dxl | Script library that includes base classes for data model description, base classes for working with links, and base classes for security rules descriptions. |
| SD3.Lib.Validator.dxl | Script library that includes base classes for validation functionality. |
| SD3.Lib.Translator.dxl | Script library that includes base classes for input translation functionality. |
| SD3.Lib.FormHelpers.dxl | Script library that includes base wrapper classes for the user interface form and controls. |
| SD3.Lib.Actions.dxl | Script library that includes base classes for working with form, view, database, and custom actions. |
| SD3.Lib.ProgressBar.dxl | Script library that includes a class for working with the progress bar user interface. |
| SD3.Lib.iCalendar.dxl | Script library that includes classes and functions for working with the iCalendar format. |
| SD3.Lib.NativePIM.dxl | Script library that includes wrapper classes for native IBM Notes PIM objects. |
| SD3.Lib.iCalendarUtil.dxl | Script library that includes utility classes and functions for data interchange between native PIM objects and iCalendar values. |
| SD3.Lib.Handlers.dxl | Script library that includes base classes for handling common events. For example, creating, saving, and removing documents. |
| SD3.Lib.PickListDialogs.dxl | Script library that includes a base class for handling a picklist dialog. |
| SD3.Lib.ProductAdapter.dxl | Script library that includes factory functions for providing product object implementations for the platform code. |
| SD3.View.AssociationsByLeftId.dxl | Includes a view for searching for association documents by their left ID value. |
| SD3.View.AssociationsByRightId.dxl | Includes a view for searching for association documents by their right ID value. |
| SD3.View.AssociationSearch.dxl | Includes a view for searching for association documents with complex keys. |
| SD3.View.CRMDocuments.dxl | Includes a view that contains all Siebel CRM related documents. |
| SD3.View.MirrorDirectSearch.dxl | Includes a view for searching for linked documents with complex keys. |

Table 12. Files You Must Not Modify

| File | Description |
|--|---|
| SD3.View.SettingsDefaults.dxl | Includes a view that contains a service document with default values. For example, current user name, currency, generic user name, and so on. |
| SD3.View.ActivityHandlingSrc.dxl | Includes a view that contains all unprocessed PIM documents. This file is not included in the preconfigured Siebel CRM product. |
| SD3.Subform.ForwardInvocation.dxl | Includes the subform used for the Forward Invocation functionality. This file is not included in the preconfigured Siebel CRM product. |
| SD3.Form.PickListDialog.dxl | Includes the form with the simple picklist dialog layout. This file is not included in the preconfigured Siebel CRM product. |
| SD3.Form.PickListDialogEx.dxl | Includes the form with the extended picklist dialog layout. |
| SD3.Form.PickListDialogNarrow.dxl | Includes the form with the narrow picklist dialog layout. |
| SD3.Form.StartSync.dxl | Includes the service form used to manually start synchronization based on a toolbar action. |
| SD3.Form.MVG.dxl | Includes the form with the MVG dialog layout. This file is not included in the preconfigured Siebel CRM product. |
| SD3.Agent.NoteCreatedHandler.dxl | Includes the agent for handling a document creation event. |
| SD3.Agent.NoteDeletingHandler.dxl | Includes the agent for handling a document deletion event. |
| SD3.Agent.NoteUpdatedHandler.dxl | Includes the agent for handling a document modification event. |
| SD3.Agent.PreSynchronizeProcessor.dxl | Includes the agent for handling a presynchronization stage. |
| SD3.Agent.PostSynchronizeProcessor.dxl | Includes the agent for handling a postsynchronization stage. |
| SD3.Agent.CleanUp.dxl | Includes the agent for handling a cleanup stage. |
| SD3.Agent.SyncNow.dxl | Includes the agent to manually start synchronization based on a toolbar action. |
| SD3.Resources.dxl | Includes platform image resources. |

Preparing the Development Environment

This topic describes how to prepare your development environment so that you can modify the customization package. The customization package includes *DXL (Domino XML) files*, which are files that include Domino data expressed as XML code. It is strongly recommend that you use IBM Domino Designer to edit DXL files, although you can edit them manually. Modifying the package DXL files consists of importing the DXL into the IBM Notes database and modifying the design elements with Domino Designer, and then exporting the DXL files. To automate the DXL import and export procedures, the utility notes application, CRM Dev Utils (CRMDevUtils.nsf), is provided. The CRM Dev Utils application may be located on the CRM Desktop server or in local IBM Notes data directory.

For more information, see [“About the Customization Package” on page 33](#) and the documentation about IBM Domino at the IBM developerWorks web site.

Preparing the development environment

- 1 Locate the customization package and tools folder on your computer.
- 2 Create a new customization project folder on your computer.
- 3 Create a new package subfolder within the project folder.
- 4 Copy the tools folder from Siebel CRM Desktop v.3.2 to your project folder.
- 5 Extract the package.zip from Siebel CRM Desktop v.3.2 to the package subfolder of your project folder.
- 6 Install IBM Notes version 8.5.3 or version 9 and IBM Domino Designer version 8.5.3 or version 9 on your development computer.
- 7 Open the IBM Notes client.
- 8 Create a new database on your local machine.
 - a Press Ctrl+N in the IBM Notes client.
 - b Enter the title and file name.
 - c Select Personal Address Book from the list of templates.
- 9 Create another new database on your local machine.
 - a Enter the title and file name.
 - b Select Mail from the list of templates.
- 10 Navigate to the tools folder and open the CRMDDDevUtils.nsf file in the IBM Notes client.
- 11 Open the CRM Projects view in the CRM Dev Utils application.
- 12 Click the New CRM Project button on the action toolbar to open a new Project form.
- 13 Complete the Project form fields:
 - a In the Project Name field, enter a descriptive name of the project.
 - b In the Tools Path field, enter the full path to the tools folder.
- 14 Save and close the Project form.

- 15 Select the project document in the CRM Projects view.
- 16 Click the New crmConfig button on the action toolbar to open a new configuration form.
- 17 Fill in Configuration form fields:
 - a In the Config Name field, enter a descriptive name for the configuration.
 - b In the Package Path field, enter the full path to the package folder.
 - c For the Config File Path field, select the export_import_config.xml file from tools folder by clicking on accompanying button.
 - d For the Import order field, enter the numeric value to define the order in which configuration should be imported. This parameter is important if a project contains several configurations. This field can contain a null value.
- 18 Save and close the Configuration form.
- 19 Open the CRM Connections view in the CRM Dev Utils application.
- 20 Click the New crmConnection button on the action toolbar to open a new Connection form.
- 21 Fill in Connection form fields:
 - a In the Project Name field, select your previously created project.
 - b In the DB ID field, select PAB as the target database type.
 - c In the Path to DB field, click on accompanying button and select the Personal Address Book database from the project folder.
- 22 Save and close the Connection form.
- 23 Repeat [Step 20-Step 22](#) to create the Mail database. For the DB ID field, enter mail.

Importing DXL files

Follow the procedures in this topic to import DXL files to your development environment.

Importing DXL files

- 1 Open the IBM Notes Client.
- 2 Open the CRM Dev Utils application in IBM Notes client.
- 3 Open the CRM Projects view in the CRM Dev Utils application.
- 4 Select the project document for which the .dxl files should be imported.
- 5 Click the Import button on the action toolbar.
- 6 Click Yes in the confirmation dialog window.
- 7 Open the correspondent PAB and Mail databases in the IBM Domino Designer and verify that all Siebel CRM Desktop design elements were successfully imported.

Exporting to DXL files

After you import DXL files and modify custom design element, you must export the design elements from the PAB and Mail databases to DXL files.

Exporting to DXL files

- 1 Open the IBM Notes Client.
- 2 Open the CRM Dev Utils application in the IBM Notes client.
- 3 Navigate to the CRM Projects view in the CRM Dev Utils application.
- 4 Select the project document for which design elements should be exported.
- 5 Click the Export button on the action toolbar.
- 6 Click Yes in the confirmation dialog window.
- 7 Open the project package folder and verify that the .dxl file is modified.
- 8 Upload and publish the package on the Siebel server for further verification. For more information, see [“Creating and Publishing the Customization Package” on page 78](#).

Using Siebel Tools

This topic describes some of the basic tasks you might perform in Siebel Tools if you customize Siebel CRM Desktop. For more information, see *Using Siebel Tools*.

Checking Out Projects 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 click Options.
- 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 Field Behavior

This topic describes how to customize the user interface. It includes the following topics:

- [Displaying Siebel CRM Fields on page 164](#)
- [Hiding Siebel CRM Fields on page 167](#)
- [Making Fields Read-Only on page 168](#)
- [Adding Default Values to Fields on page 169](#)
- [Adding Postdefault Values to Fields on page 172](#)
- [Updating One Field If the User Modifies Values In Another Field on page 173](#)
- [Creating Calculated Fields on page 175](#)

Displaying Siebel CRM Fields

The example in this topic displays the Mail Stop field on the Contact form in the client. You make this field available through the Siebel API and then customize CRM Desktop to synchronize and display the field. You modify the following files:

- siebel_meta_info.xml
- Ln_siebel_basic_mapping.xml
- Ln_package_res.xml
- (SBL)\CRMContactSubform.version8 subform design element

For more information, see [“Files in the Customization Package” on page 355](#).

To display a Siebel CRM field

- 1 Open Siebel Tools and then display the object type named Integration Object.

For more information, see [“Displaying Object Types in Siebel Tools” on page 163](#).

- 2 Make sure the Mail Stop field exists on the Contact business component.

If it does not, then add it now.

- 3 Add the Mail Stop field to the CRMDesktopContactIO integration object.

In order for CRM Desktop to synchronize data with the Siebel database, you use 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 |

- 4 Repeat [Step 3](#) for the CRMDesktopAccountIO integration object.

- 5 Repeat [Step 3](#) for the CRMDesktopOpportunityIO integration object.

- 6 Compile all locked projects.

After compiling finishes, the Mail Stop field is available through the API and you can configure CRM Desktop to use the field. For more information, see *Using Siebel Tools*.

- 7 Specify the objects and fields to synchronize:

- a Use an XML editor to open the siebel_meta_info.xml file.

For more information, see [“Files in the Customization Package” on page 355](#).

- b Locate the following tag:

```
object TypeId='Contact'
```

Several child field tags reside in the `object TypeId='Contact'` tag. These children identify the fields for the Contact object.

- c Add the following field tag as a child of the tag that you located in [Step b](#):

```
<field Name='Mail Stop' Label='Mail Stop' DataType='DTYPE_TEXT'
  IOElementName='Mail Stop' />
```

- 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.

- 8 Map the Mail Stop field from the Contact object in the Siebel database to a field in CRM Desktop:

- a Use an XML editor to open the Ln_siebel_basic_mapping.xml file.

For more information, see [“Files in the Customization Package” on page 355](#).

- b In the Ln_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>
    <lotus_std>
      <lotus_field id="Mail Stop" />
      <convertor>
        <string/>
      </convertor>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="Mail Stop" />
      <convertor>
        <string/>
      </convertor>
    </lotus_std>
  </writer>
</field>
```

- d Save and close the Ln_siebel_basic_mapping.xml file.

- 9 Insert a label and the Mail Stop field just following the Job Title field on the Contact form:

- a Open IBM Domino Designer, locate, and then open one of the following forms:

- Contact.version7 form for IBM Notes version 7

- Contact.version8 form for IBM Notes version 8

This example describes how to modify the Contact.version8 design element. For more information, see [“Opening IBM Domino Designer” on page 111](#).

- b** Add the Mail Stop text field.

In IBM Domino Designer you must add one cell for the label and one cell for the field. In the label cell, you must add the localization macro. You use the following localization macro for this example:

```
(!loc:lbl_mail_stop:loc!):
```

where:

- **(!loc.** Is the beginning of the macro.
- **lbl-mail_stop.** Is the unique Id for the field.
- **loc!):.** Is the end of the macro.
- **lbl_mail_stop.** Specifies the key that the Ln_package_res.xml file uses to determine the localized value for the label.

- c** Set the name of the field cell that you added in [Step b](#) to MailStop.

- 10** Add the following code to the Ln_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 CRMD -->
```

This code provides localized values to the client. It allows the Contact form to display the Mail Stop label through a key value. The Ln_package_res.xml file provides localized values and images to CRM Desktop. You added the new Mail Stop field to the Contact form, so you must provide the text for the label. When you modified the Contact.version7 or Contact.version8 form, you created text that contains lbl_mail_stop for the text value. This text value identifies the key to use in the Ln_package_res.xml file.

- 11** Republish the updated package files.

During the next synchronization, 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 CRM Desktop synchronizes the values in this field with the Siebel Server. For more information, see [“Republishing Customization Packages” on page 80](#).

Hiding Siebel CRM Fields

The example in this topic describes how to hide the Opportunity field so that Siebel CRM Desktop does not display it in the client.

To hide Siebel CRM fields

- 1 Remove the following objects from the Ln_connector_configuration.xml file:
 - Remove all child objects, such as Opportunity.Contact.Association.
 - Remove all association objects, such as Opportunity.Contact.Association.
 - Remove links to opportunities that exist in the links and natural_keys sections.
- 2 Remove the following objects from the Ln_siebel_basic_mapping.xml file:
 - Remove all child objects, such as Opportunity.Contact.Association.
 - Remove all association objects, such as Opportunity.Contact.Association.
 - Remove links to opportunities from objects that include fields that are similar to OpportunityId.
- 3 Remove the following controls for this Opportunity object from the (SBL)form:opportunity design form:
 - Remove embedded folders from forms that display opportunities and objects that are related to opportunities.
 - Remove lookup controls and autocomplete controls from other forms that reference the Opportunity object.
- 4 Remove related actions from the views action bars.
- 5 In the SBL.BusinessLogic script library, remove the following definitions where CRM Desktop uses this Opportunity object:
 - AddMVGLink
 - AddDirectLink

You can remove other code that affects opportunities, but this code must not affect other CRM Desktop functionality.

Making Fields Read-Only

To make a field read-only, you disable the corresponding control on the form that references the field that you must make read-only. The example in this topic makes the Opportunity Name field on the opportunity form read-only.

To make a field read-only

- 1 Open IBM Domino Designer, and then open the (SBL)form:opportunity form.
For more information, see ["Opening IBM Domino Designer" on page 111](#).
- 2 Choose the Name field.
- 3 Click the Objects pane and then choose Input Enabled.
- 4 In the Event Properties tab that IBM Domino Designer displays along the right side of the workspace, add the following code:


```
@False;
```

IBM Domino Designer disables the field in the client.

- 5 Open the client and then navigate to the Opportunity form.
- 6 Open an opportunity and make sure you cannot modify the Opportunity Name.

Adding Default Values to Fields

This topic describes how to configure Siebel CRM Desktop to add a default value to a field when the user creates a new record. The example in this topic configures CRM Desktop to add the following default value to the Opportunity Name field:

CRM Opportuni ty

To add a default value to a field

- 1 Open IBM Domino Designer, and then open the SBL.BusinessLogic script library.

For more information, see ["Opening IBM Domino Designer" on page 111](#).

- 2 Locate and then open the CreateMetaScheme function.

- 3 Locate the following code:

```
Set TypeDescr = MetaScheme.GetType("Opportuni ty")
```

where:

- **MetaScheme.GetType**. Identifies the name of the object type identifier of the object type that resides in the Ln_siebel_basic_mapping.xml file. This file includes definitions for object types. Each definition includes a Lotus Id attribute. This attribute is the object type identifier that you must use for the object_type. For more information, see ["Customizing Field Mapping" on page 154](#).
- 4 In the With block of code that IBM Domino Designer displays immediately before the End With line, add the following code:

```
.Field("Name").Initial Value = "CRM Opportuni ty"
```

where:

- **.Field("Name")**. Identifies the name of a field that resides in the object type definition in the Ln_siebel_basic_mapping.xml file.
- **CRM Opportunity**. Defines the default value that CRM Desktop adds.

Setting Default Values

This topic describes the values that you can use to specify the source of the default value. You specify this value in the InitialValue setter property of the CreateMetaScheme function.

Setting Fixed Default Values

To set a fixed value, you set the InitialValue setter property of the CreateMetaScheme function to the following value:

```
i n i t i a l _ v a l u e
```

You typically use this format with a number that does not require a translation, such as a Boolean number or an empty string where the default value is empty. You use the following format:

```
. F i e l d (" f i e l d _ n a m e " ) . I n i t i a l V a l u e = " v a l u e "
```

Example 1

To set the Appt PIM Flag field that resides in the Action object, you use the following code:

```
. F i e l d (" P I M O r i g i n a t e d " ) . I n i t i a l V a l u e = " 1 "
```

This example sets the default value to true. It does not use a list of values. The Appt PIM Flag field is a text field.

Example 2

To set the country code for the phone number to a default value when the user creates a new account, you can use the following code:

```
. F i e l d (" M a i n P h o n e " ) . I n i t i a l V a l u e = "+ 3 1 "
```

Setting Default Values That Are Language Dependent

To get a language dependent value from the resource file, you set the InitialValue variable of the CreateMetaScheme function to the following value:

```
i n i t i a l _ v a l u e _ r e s
```

This value uses the value that Siebel CRM Desktop defines in the Ln_package_res.xml file and in each language-specific Ln_package_res_XX_yy.xml file. It allows you to use a different default value for each language. If the MLOVs (multi-value lists of values) are correctly configured on the Siebel Server, then CRM Desktop stores the correct Language-Independent Code on the server. You use the following format:

```
. F i e l d (" f i e l d _ n a m e " ) . I n i t i a l V a l u e = " i n i t i a l _ v a l u e _ r e s "
```

Example

The following example specifies to set the default value of the Display field that resides in the Activity object to the value that the lang_action_display_activities_only resource key contains:

```
. F i e l d (" D i s p l a y " ) . I n i t i a l V a l u e = "[ ! l o c : l a n g _ a c t i o n _ d i s p l a y _ a c t i v i t i e s _ o n l y : l o c ! ] "
```

The following values in the resource file determine the default value:

- The English resource file includes the following code:

```
<str key="lang_acti on_di spl ay_acti vi ti es_onl y" >Acti vi ti es Onl y</str>
```

At run time CRM Desktop sets the default value in the English client to Activities Only.

- The French resource file includes the following code:

```
str key="lang_acti on_di spl ay_acti vi ti es_onl y">Acti vi tés uni quement</str>
```

At run time CRM Desktop sets the default value in the Dutch client to Alleen activiteiten.

Setting Default Values for Functions

To set the default value for a function, you set the InitialValueCallback setter property of the CreateMetaScheme function to the following value:

```
i ni ti al _val ue_fn
```

This value uses a function that returns a value. It allows you to do more complex lookups. You use the following format:

```
Set. Fi el d("fi el d_name"). I ni ti al Val ueCal l back = i ni ti al _val ue_fn
```

Example

The following example comes predefined with Siebel CRM Desktop. It calls the following DefaultCurrencyIdCb function to determine the currency code that CRM Desktop sets for the account:

```
Ret. GetType("Account"). Fi el d("CurrencyCodeId"). I ni ti al Val ueCal l back =  
DefaultCurrencyIdCb
```

The following predefined code defines the CallbackDefaultCurrencyId class:

```
Public Class CallbackDefaultCurrencyId As CallbackObject  
    Public Function Invoke (params As DynamicArguments)  
        Invoke = GetSession(). Defaults.Property("CRMDSibel CurrencySymbol ")  
    End Function  
End Class
```

where:

- CRM Desktop defines CallbackDefaultCurrencyId class as the following:

```
Dim DefaultCurrencyIdCb As New CallbackDefaultCurrencyId
```

Setting Default Values For Links

To set the default links that Siebel CRM Desktop uses for an object type, you set the InitialLinksCallback property of the descriptor that this object type uses to an instance of the class that CRM Desktop gets from the CallbackObject class. CRM Desktop uses this property to create the initial links. The following example uses a multi-value group that returns multiple records:

```
Set. I ni ti al Li nksCal l back = New Cal l backI ni ti al Li nksAccount
```

In this example, the following code describes the CallbackInitialLinksAccount class:

```

Public Class CallbackInitialLinksAccount As CallbackInitialLinks
    Public Function Invoke (params As DynamicArguments)
        Dim PositionId As String
        Dim OrganizationId As String

        ' Prefill team
        PositionId = GetSession().GetCurrentUser().Property("CRMDPositionId")
        Me.Container(params).Push CreateLinkContainer(PositionId, "Position",
LINK_TAG_DIRECT)
        Me.Container(params).Push CreateLinkContainer(PositionId, "Position",
LINK_TAG_MVG)

    End Function
End Class

```

This example uses a multi-value group that associates multiple records. To add multiple team members instead of setting a single value, you can use a function that you specify elsewhere and use the following source:

```
InitialLinksCallback
```

Adding Postdefault Values to Fields

This topic describes how to add a value to a field if the user does not enter any data in this field when the user creates a new record. Adding a value in this way is known as adding a *postdefault* value. The example in this topic modifies the predefined opportunity form to make sure Siebel CRM Desktop sets the postdefault value for the Lead Quality field in the opportunity to 5-Poor if the user does not set a value for this field.

To add a postdefault value to a field

- 1 Open IBM Domino Designer, and then open the SBL.Forms script library.
For more information, see ["Opening IBM Domino Designer" on page 111](#).
- 2 Locate the FormOpportunityEx class.
- 3 Add the following function to the end of the FormOpportunityEx class:

```

' Comments for QuerySave
Public Function QuerySave As Boolean
    QuerySave = SBLFormEx. .QuerySave
    If Len(Me.DocumentEx.SafeProperty("LeadQuality", "")) = 0 Then
        Me.DocumentEx.Property("LeadQuality") = "5-Poor"
    End If

```

To determine if the user set the value for the Quality field, this if statement uses the SaveProperty method of the DocumentEx property, and it returns the Quality field value:

- If the field is empty, then the Len function returns a value of 0 and sets the value of the Quality field to 5-Poor.

- If the field is not empty, then the `empty_field_validator` function returns a value of false and the code exits the if statement.
- 4 (Optional) Support an environment that does not use English. You do the following:
 - a Replace the `Me.DocumentEx.Property("LeadQuality") = "5-Poor"` code that you added in [Step 3](#) with the following code:


```
Me.DocumentEx.Property("LeadQual i ty") = "[! loc: l ang_l ead_qual i ty_poor: l oc! ]"
```

 where:
 - `!loc: lang_lead_quality_poor: loc!` is a localization macro.
 - b Add the following code to the `Ln_package_res.xml` file:


```
<str key="l ang_l ead_qual i ty_poor">5-Poor</str>
```

 The code you added in [Step 3](#) works in an environment that uses English but fails in a multilingual environment because you hard-coded the field value. The localization macro allows you to retrieve a string from the resource file and to write code that is language independent.
 - 5 Test your changes and then republish the customization package.
For more information, see ["Republishing Customization Packages" on page 80](#).

Updating One Field If the User Modifies Values In Another Field

You can configure Siebel CRM Desktop to update the value in one field if the user changes the value in another field. This functionality is known as *on field update*. The example in this topic configures CRM Desktop to do the following work:

- If the name of the opportunity is Test, then set the Lead Quality to 5-Poor.
- If the name of the opportunity is not Test, then set the Lead Quality to 3-High.

To update one field if the user modifies values in another field

- 1 Open IBM Domino Designer, and then open the (SBL)form:opportunity form,
For more information, see ["Opening IBM Domino Designer" on page 111](#).
- 2 Click the Name field in the (SBL)form:opportunity form.
- 3 In the Objects tab, locate and choose the `onChange` event.
- 4 From the Run drop-down list, choose the Client option, and then choose the IBM Notes Script option.
- 5 Add the following code:

```
m_FormHandler.FormManager.GetControl("Name").OnChange.RaiseArgs()
```

where:

- `m_FormHandler` is an instance of the `FormOpportunityEx` class.

- 6 Save your changes.
- 7 Open the `SBL.Forms` script library.
- 8 To create the `CallbackOpportunityNameOnChange` class, add the following code:

```
' =====
' CallbackOpportunityNameOnChange
' Class that represents
' =====
Public Class CallbackOpportunityNameOnChange As CallbackObject
Private m_Doc As DocumentEx

' Comments for New
Sub New(doc As DocumentEx)
    Set m_Doc = doc
End Sub
' Comments for Invoke
Public Function Invoke(params As DynamicArguments)
    Dim NameValue As String
    NameValue = m_Doc.SafeProperty("Name", "")
    If NameValue = "Test" Then
        m_Doc.Property("LeadQuality") = "[!loc:lang_lead_quality_poor:loc!]"
    Else
        m_Doc.Property("LeadQuality") = "[!loc:lang_lead_quality_high:loc!]"
    End If
End Function
End Class
```

This code defines the callback that CRM Desktop sends if the user modifies the value of the Lead Quality field. CRM Desktop uses the following localization macros in this class for each value that this field can contain:

```
[!loc:lang_lead_quality_poor:loc!]
[!loc:lang_lead_quality_high:loc!]
```

If you use these macros, then you must also modify the `Ln_package_res.xml` file as described in [Step b on page 173](#). You can also hard code these values to 5-Poor and 3-High.

- 9 In the `FormOpportunityEx` class, switch to the `PostOpen` function, and then locate the following code:

```
Dim ControlStatus As New ControlEditable ("Status", Me.Form)
```

- 10 Add the following code immediately after the code you located in [Step 9](#):

```
Dim ControlName As New ControlEditable("Name", Me.Form)
Dim NameCallback As New CallbackOpportunityNameOnChange (Me.DocumentEx)
ControlName.OnChange.Connect NameCallback
```

where:

- `Dim ControlName As New ControlEditable("Name", Me.Form)` creates a virtual control.

- `Dim NameCallback As New CallbackOpportunityNameOnChange (Me.DocumentEx)` creates the callback for the class that you added in [Step 8](#).
- `ControlName.OnChange.Connect NameCallback` signs the callback that CRM Desktop runs for the control.

11 Test your changes and then republish the customization package.

For more information, see ["Republishing Customization Packages" on page 80](#).

Creating Calculated Fields

You can configure Siebel CRM Desktop to display a calculated field in the client so that it behaves in a way that is similar to how a calculated field behaves in the client of a Siebel Business Application. The example in this topic configures a calculated field in the client. If the user changes the opportunity name, then the value in this field also changes. You do the following:

- Expose a calculated field to an integration object and synchronize it with CRM Desktop. This configuration allows CRM Desktop to get a correct starting value for the calculated field. In this example, you use a calculated field named JVD Calculated. It includes the following calculated value:
`[Name] + " - Calculated"`
- Use a CRM Desktop calculated field to make a working copy of the original calculated field. You do this because you cannot configure CRM Desktop to make changes to a calculated field from Siebel CRM while the user is using CRM Desktop. Doing so might cause a synchronization error.
- Use JavaScript in the form to make sure the copy of the calculated field changes if the user changes the opportunity name. This configuration allows you to use the same behavior that occurs in the client of a Siebel Business Application for the calculated field.

For more information about each of these items, see ["Alternative Ways to Create Calculated Fields" on page 178](#).

To create a calculated field

- 1 Open Siebel Tools and then display the object type named Integration Object.
For more information, see ["Displaying Object Types in Siebel Tools" on page 163](#).
- 2 In the Object Explorer, click Integration Object.
- 3 In the Integration Objects list, query the Name property for CRMDesktopOpportunityIO.
- 4 In the Object Explorer, expand the Integration Object tree and then click Integration Component.

- 5 In the Integration Components list, add a new integration component using values from the following table.

| Property | Value |
|--------------------|----------------|
| Name | JVD Calculated |
| Data Type | DTYPE_TEXT |
| Length | 285 |
| External Sequence | 240 |
| External Name | JVD Calculated |
| External Data Type | DTYPE_TEXT |
| XML Sequence | 240 |
| XML Tag | JVDCalculated |

This integration component exposes the Siebel CRM calculated field to the integration object for the opportunity.

- 6 Deploy your changes to the Siebel Runtime Repository.
- 7 Use an XML editor open the siebel_meta_info.xml file.
- 8 Locate the following object:

```
Typel d="Opportuni ty"
```

- 9 Add the following code to the object you located in [Step 8](#):

```
<fi el d Name=' JVD Cal cul ated' Label =' JVD Cal cul ated' DataType=' DTYPE_TEXT'
IsFi l terabl e=' no' IsHi dden=' no' IOEl emName=' JVDCal cul ated' />
```

This code adds the JVD Calculated field to the metadata object for Opportunity. For more information, see ["JavaScript Files in the Customization Package" on page 357](#).

- 10 Set the calculated field in Siebel CRM Desktop. You add the following code to the Ln_siebel_basic_mapping.xml file:

```
<fi el d i d="JVD Cal cul ated">
  <reader>
    <l otus_STD>
      <l otus_fi el d i d="JVDCal cul ated"/>
      <convertor>
        <stri ng/>
      </convertor>
    </l otus_STD>
  </reader>
  <wri ter>
    <l otus_STD>
      <l otus_fi el d i d="JVDCal cul ated"/>
      <convertor>
        <stri ng/>
      </convertor>
    </l otus_STD>
  </wri ter>
</fi el d>
```



```

    </lotus_STD>
  </writer>
</field>

```

- 11** Open IBM Domino Designer, and then open the (SBL)form:opportunity form.

For more information, see ["Opening IBM Domino Designer" on page 111](#).

- 12** Use IBM Domino Designer to add and position the JVD Calculated field. Use the same Name for this field that you specified in the Ln_siebel_basic_mapping.xml file in [Step 8](#).

In this example, name it JVDCalculated. Make sure that the Text and Editable properties are chosen for this field.

- 13** Make the field read-only. In the Objects tab, choose the Input Enabled event, and then enter the following code in the pane that IBM Domino Designer displays on the right side of the workspace:

```
@False;
```

A calculated field in Siebel CRM is read-only. It is recommended that you also make the calculated field in CRM Desktop read-only. For more information, see ["Making Fields Read-Only" on page 168](#).

- 14** Save your work.

- 15** Open the SBL.Forms script library, and then add the following code to create a new class:

```

' =====
' CallbackOpportunityJVDCalculated
' Class that represents
' =====
Public Class CallbackOpportunityJVDCalculated As CallbackObject
Private m_Doc As DocumentEx

' Comments for New
Sub New(doc As DocumentEx)
    Set m_Doc = doc
End Sub

' Comments for Invoke
Public Function Invoke(params As DynamicArguments)
    Dim NameValue As String
    NameValue = m_Doc.SafeProperty("Name", "")
    m_Doc.Property("JVDCalculated") = NameValue & " - Calculated"
End Function
End Class

```

The Invoke function of this callback class gets the value of the Name field, and then writes this value to the JVDCalculated field plus the following string concatenated to the opportunity name:

– Calculated

To calculate the value, Siebel CRM uses the following code:

```
[Name] + " – Calculated"
```

- 16** In the FormOpportunityEx class, switch to the PostOpen function, and then locate the following code:

```
Dim ControlStatus As New ControlEditable ("Status", Me.Form)
```

- 17** Add the following code immediately after the code you located in [Step 16](#):

```
Dim JVDCalculateCallBack As New CallBackOpportunityJVDCalculate (Me.DocumentEx)
ControlName.OnChange.Connect JVDCalculateCallBack
```

This code makes sure CRM Desktop calls this function if the user changes the value in a field that the calculation uses. In this example, the calculated value depends only on the opportunity name, so this code only calls this function if the user changes the value in the Opportunity Name field.

- 18** Test your changes and then republish the customization package.

For more information, see [“Republishing Customization Packages” on page 80](#).

Alternative Ways to Create Calculated Fields

This topic describes alternative ways to create a calculated field.

Exposing Calculated Siebel CRM Fields

To create a calculated field, you can expose a calculated field that exists in Siebel CRM to an integration object and then synchronize it to Siebel CRM Desktop. However, if this field is not read-only in the client, then CRM Desktop attempts to synchronize the new value back to Siebel CRM, and this synchronization fails.

Specifying Fields in the Siebel Meta Info File as Calculated Fields

You can specify a field in the siebel_meta_info.xml file as a calculated field. To do this, you set the IsCalculated attribute to yes and then use the following code to specify a value for the Formula attribute:

```
: [ : ] Block, should contain field.
: ( ) Container for fieldname.
```

For example, you can use the following code to combine the First Name field and the Last Name fields:

```
: [: (First Name) : (Last Name): ]
```

This primary allows you to concatenate fields with the possibility to add some static characters to the concatenation. It does not allow you to configure CRM Desktop to do a calculation. You specify this code in the siebel_meta_info.xml file. CRM Desktop only determines the calculated value during synchronization. From this point the field is read-only.

Using LotusScript to Mimic Calculated Fields

You can write LotusScript that mimics the behavior of a calculated field. You can configure CRM Desktop to run this LotusScript code only in reply to something that happens in the form, such as the user changing the value in a field. This configuration updates a field if the data changes in the form but you cannot use it to control the value of a calculated field that Siebel CRM Desktop displays when the user opens a form.

Customizing UI Behavior

This topic describes how to customize behavior in the user interface. It includes the following topics:

- [Customizing the Product Name on page 179](#)
- [Customizing the Email Address of the Support Team on page 179](#)
- [Controlling How Siebel CRM Desktop Pins Objects on page 181](#)
- [Controlling How Siebel CRM Desktop Handles Data That Is Not Directly Visible on page 181](#)
- [Preventing Users from Deleting Records on page 184](#)
- [Preventing Users from Deleting Records According to Conditions on page 185](#)
- [Making Forms Read-Only on page 186](#)
- [Localizing Strings on page 188](#)

Customizing the Product Name

The client displays the following text in a number of locations:

- Siebel CRM
- Siebel CRM Desktop
- IBM Notes

You can change this text to a custom value.

To customize the product name

- 1 Use an XML editor open the Ln_package_res.xml file.
For more information, see ["Files in the Customization Package" on page 355](#).
- 2 Create or modify any of the following attributes, as required:
 - `<str key="app_name">CRM Desktop</str>`
 - `<str key="pi m_name">IBM Notes</str>`
 - `<str key="remote_app_name">Si ebel </str>`
 For example, in the remote_app_name attribute, change Siebel to your company name.
- 3 Save and test your changes and then republish the customization package.
For more information, see ["Republishing Customization Packages" on page 80](#).

Customizing the Email Address of the Support Team

The defines various resources for the customization package. In this file, you can specify the email address of the support team where the user sends feedback.

To customize the email address of the support team

1 Use an XML editor open the Ln_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 where Siebel CRM Desktop sends requests for support

For example:

```
<str key="support_email">support@example.com</str>
```

If the user clicks Send Feedback on the Feedback tab in the Options dialog box, then 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 you do not specify the email address, then CRM Desktop opens the email without an email address in the To line. CRM Desktop does not come predefined with a support email address. You must specify it.

The example in this topic describes how to configure security rules to make accounts read-only.

3 Open IBM Domino Designer, open the SBL.SecurityRules script library, and then choose the AccountSecurityRule class.asdf

For more information, see ["Opening IBM Domino Designer" on page 111](#).

4 Make sure the AccountSecurityRule class includes the following code. If it does not include this code, or if the AccountSecurityRule class does not exist, then add it now:

```
' =====
' AccountSecurityRule
' Class that represents
' =====
Public Class AccountSecurityRule As SiebelSecurityRule
    Sub New
        m_RequiredPrimaryType = "Position"
    End Sub

    Public Sub LinkAccess(docEx As DocumentEx, LinkCtx As ContextLink, result As LinkAccessItem)
        Me.TerritoryAndPrimaryPositionLinkAccess docEx, LinkCtx, result
    End Sub
End Class
```

where:

- Public Class AccountSecurityRule As SiebelSecurityRule configures CRM Desktop to get the logic for this rule from the SiebelSecurityRule class.

Controlling How Siebel CRM Desktop Pins Objects

CRM Desktop pins all objects that the Online Lookup feature brings into the client. CRM Desktop pins these items for one week, by default. It removes these pinned items from the client after one week, including the record and any links to the record. You can change this default duration value.

Controlling How Long Siebel CRM Desktop Pins Objects

This topic describes how to change the default value that determines how long CRM Desktop pins an object.

To control how long Siebel CRM Desktop pins an object

1 Use a JavaScript editor to open the `online_lookup_sbl.js` file.

2 Locate the following code:

```
online_lookup_sbl.superclass.constructor.apply(this, arguments);
```

3 Add the following line following the code that was located in [Step 2](#).

```
this.options.pinned_object_lifetime = pin_time;
```

where:

- *pin_time* specifies the number of seconds to pin an item. Note the following:
 - The default value is 604800, which is seven days.
 - One day is 86400.
 - You can specify -1 (negative one) to pin objects forever.

Controlling How Siebel CRM Desktop Handles Data That Is Not Directly Visible

You can use the `viewmodes` element to configure Siebel CRM Desktop to display or hide data that is not directly visible. *Data that is not directly visible* is a type of data that the client does not receive during synchronization, but instead gets through an association with another Siebel CRM object type. For example, filter settings might prevent CRM Desktop from synchronizing some accounts to the client, but the synchronized contacts that reference these accounts might contain this account data. In this situation, CRM Desktop displays this account information in the Contact form, but not in an Account view.

Starting with Siebel CRM Desktop version 3.1, you can configure a view mode according to the type of operation that CRM Desktop performs. You can use one of the following values for the `viewmodes` element:

- **General.** Specifies a value for the General viewmode. If this value exists, then it overrides a value that the `ViewMode` attribute specifies.

- **Dedup.** Specifies a value for the Deduplication view mode. For more information, see [Resolving Synchronization Conflicts on page 148](#).
- **QBID.** Specifies a value for the Query-By-Id view mode.
- **Not specified.** If you do not specify the viewmodes element, then CRM Desktop does the following:
 - Uses the default value for the deduplication viewmode that it gets from the value that you set for the General view mode
 - Uses All as the default value for the Query-By-Id view mode.

CRM Desktop uses a master filter only on query change requests when it uses the General view mode.

For more information, see [“How CRM Desktop Displays Data That Is Not Directly Visible” on page 60](#).

Using Query By Id to Hide Data That is Not Directly Visible

The example in this topic configures CRM Desktop to hide all data that is not directly visible for contacts. It prevents CRM Desktop from getting data from the Siebel Server and storing it in the client.

To hide data that is not directly visible for contacts

- 1 Use an XML editor open the siebel_meta_info.xml file.
- 2 Define the following view modes for the contact object type:


```
General="Sales Rep" Dedup="All" QBID="Sales Rep"/>
```
- 3 Define the following view modes for the account object type:


```
General="Sales Rep" Dedup="All" QBID="Sales Rep"/>
```
- 4 Define the following view modes for the opportunity object type:


```
General="Sales Rep" Dedup="All" QBID="Sales Rep"/>
```
- 5 Define the following view modes for Contact.Account and Account.Contact:


```
General="Sales Rep" Dedup="All" QBID="Sales Rep"/>
```
- 6 Define the following view modes for Contact.Opportunity and Opportunity.Contact:


```
General="Sales Rep" Dedup="All" QBID="Sales Rep"/>
```
- 7 Make sure the user does not modify the default filter preset that restricts the number of objects that CRM Desktop synchronizes from the Siebel Server.

Using Query By Id to Synchronize Only My Accounts and Activities

- 1 Use an XML editor open the siebel_meta_info.xml file.
- 2 Locate the following code:

Account

```
<object TypeId='Account' Label='#obj_account' LabelPlural='#obj_account_plural'
UpsertBusObjCacheSize='0' EnableGetIdsBatching='true'
IntObjName='CRMDesktopAccountIO' SiebMsgXmlElemName='Account'
SiebMsgXmlCollectionElemName='ListOfCRMDesktopAccountIO' >

<viewmodes General="Sales Rep" Dedup="All"/>
```

Account.Action

```
<object TypeId='Account.Action' Label='Activity' LabelPlural='Activities'
EnableGetIdsBatching='true' IntObjName='CRMDesktopAccountIO'
SiebMsgXmlElemName='Action' SiebMsgXmlCollectionElemName='ListOfAction' >

<viewmodes General="All" Dedup="All" />
```

- 3 Modify the code you located in [Step 2](#) to the following code. Bolded font indicates the modifications you must make:

Account

```
<object TypeId='Account' Label='#obj_account' LabelPlural='#obj_account_plural'
UpsertBusObjCacheSize='0' EnableGetIdsBatching='true'
IntObjName='CRMDesktopAccountIO' SiebMsgXmlElemName='Account'
SiebMsgXmlCollectionElemName='ListOfCRMDesktopAccountIO' >

<viewmodes General="Sales Rep" Dedup="All" QBIID="Sales Rep"/>
```

Account.Action

```
<object TypeId='Account.Action' Label='Activity' LabelPlural='Activities'
EnableGetIdsBatching='true' IntObjName='CRMDesktopAccountIO'
SiebMsgXmlElemName='Action' SiebMsgXmlCollectionElemName='ListOfAction' >

<viewmodes General="Personal" Dedup="All" QBIID="Personal"/>
```

Using Filters to Hide Data That Is Not Directly Visible

This topic describes how to hide data that is not directly visible for accounts from a custom IBM Notes view that Siebel CRM Desktop uses. It describes how to prevent CRM Desktop from storing data in the client. For information about how to prevent CRM Desktop from getting data from the Siebel Server and storing it in the client, see [“Using Query By Id to Hide Data That is Not Directly Visible” on page 182](#).

To use filters to hide data that is not directly visible

- 1 Open IBM Domino Designer, and then choose the _Account view.
For more information, see [“Opening IBM Domino Designer” on page 111](#).
- 2 In the Objects tab, choose View Selection.
- 3 Make sure IBM Domino Designer displays the View Selection code. For example:

```
SELECT CRMDType="Account" & (@Contains(ObjectStateString; "P") |
! (@Contains(ObjectStateString; "I ") | @Contains(ObjectStateString; "V")))
```

where:

- CRMDType="Account" configures CRM Desktop to choose only accounts for the view.
 - (@Contains(ObjectStateString; "P")) establishes the P object state string.
 - ! (@Contains(ObjectStateString; "I ")) makes sure CRM Desktop does not established the I object state string.
 - (@Contains(ObjectStateString; "V")) establishes the V object state string.
- 4 Use an XML editor open the Ln_connector_configuration.xml file, and then make sure the state_field for the Account type uses the following value:

```
<type id="Account" state_field="ObjectState">
```

If you copy this view, then CRM Desktop also copies this filter. This configuration allows you to create other views that you can use with Siebel CRM data.

Preventing Users from Deleting Records

This topic describes how to configure Siebel CRM Desktop to prevent the user from deleting records in the client. You can also configure it to allow the user to delete a record in the client and then confirm this deletion during synchronization. For more information, see [“Configuring Siebel CRM Desktop to Disregard Erroneous Data That Users Modify” on page 138](#) and [“Controlling How Siebel CRM Desktop Deletes Records During Synchronization” on page 145](#).

To prevent users 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 163](#).
- 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 client.
- 10 Delete a contact.
- 11 Perform a synchronization.
- 12 Make sure 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 (SBhL-EAI -04183)
- 13 Republish the customization package.

For more information, see ["Republishing Customization Packages" on page 80](#).

Preventing Users from Deleting Records According to Conditions

This topic describes how to configure Siebel CRM Desktop to prevent the user from deleting records in the client according to a condition. For example, not allowing the user to delete an opportunity if the Status is Pending. The delete button in the client is a native IBM Notes button that you cannot disable. Instead, you can modify the code that runs if the user clicks Delete.

The example in this topic prevents the user from deleting any Opportunity that includes a Status of Pending. You can configure Siebel CRM Desktop to handle all Siebel CRM objects in the same way. This configuration applies to contacts, accounts, opportunities, and any other custom Siebel CRM object, such as service requests or call reports.

To prevent users from deleting opportunities according to conditions

- 1 Open IBM Domino Designer, and then open the SBL.SecurityRules script library.

For more information, see ["Opening IBM Domino Designer" on page 111](#).
- 2 Locate the DeleteAccess function in the OpportunitySecurityRule class.
- 3 Add the following code to the end of the DeleteAccess function:

If docEx.Property("Status") = "Pending" Then DeleteAccess = False

Configuring CRM Desktop to Automatically Add Deleted Items to the Exclusion List

When a user deletes a synced Account, Contact or Opportunity from CRM Desktop, a confirmation dialog box appears and prompts the user to add the deleted item to the Exclusion list. Deleted items are deleted from IBM Notes but not from the Siebel CRM Desktop server.

You can configure CRM Desktop to automatically add an item that the user deletes to the Exclusion list without any confirmation dialogs.

Or you can totally disable deletion of objects from CRM Desktop.

To configure CRM Desktop to automatically add deleted items to the Exclusion list

- 1 Open IBM Domino Designer, and then open the SBL.Actions script library.

For more information, see [“Opening IBM Domino Designer” on page 111](#).

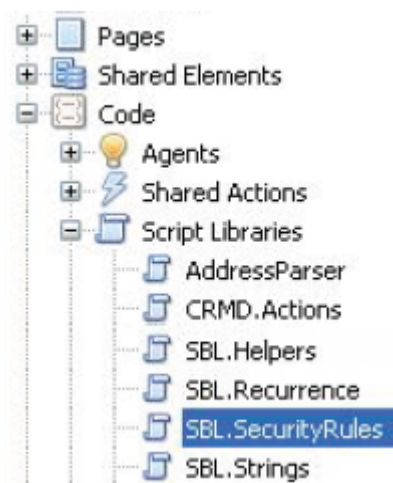
- 2 Locate the sub New in the ActionDeleteOrExclude class.
- 3 Set the m_ShowExclusionConfirmation variable to False.
- 4 Save the SBL.Actions script library.

Making Forms Read-Only

You can configure Siebel CRM Desktop to make a form in the client read-only. The user can view values in a read-only form but cannot modify these values. The example in this topic describes how to make the form that displays account information a read-only form. If you configure Siebel CRM Desktop to make a form read-only, then the user cannot use the Edit button or double-click the form to make it writable.

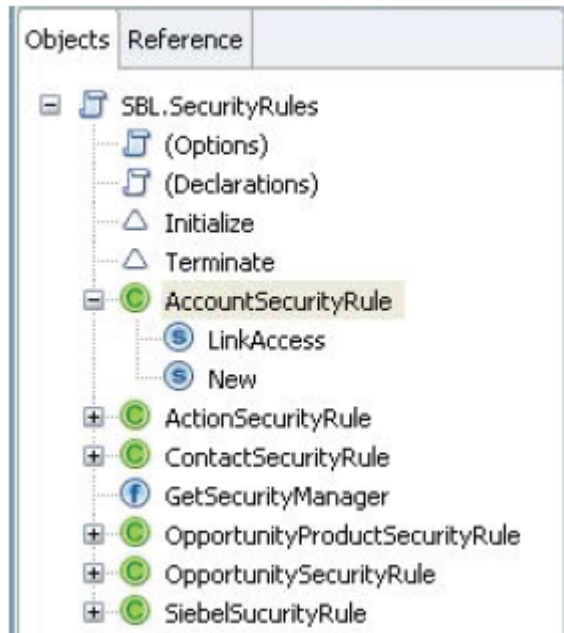
To make forms read-only

- 1 Open IBM Domino Designer, expand the Code tree, expand the Script Libraries tree, and then double-click SBL.SecurityRules:



For more information, see [“Opening IBM Domino Designer” on page 111](#).

- 2 In the SecurityRules script library that IBM Domino Designer displays, in the Object tab, expand the rule for the object type that you must make read-only. For this example, expand AccountSecurityRule:



- 3 Click the ModifyAccess function that resides in the rule that you clicked in [Step 2](#).
The ModifyAccess function does not exist in this example. If the rule that you clicked in [Step 2](#) does not include the ModifyAccess function, then you must add it using the following code:

```
Public Function ModifyAccess(docEx As DocumentEx) As Boolean
End Function
```

- 4 Modify the ModifyAccess function so that it always returns a value of false. You add the ModifyAccess = false line:

```
Public Function ModifyAccess(docEx As DocumentEx) As Boolean
    ModifyAccess = false
End Function
```

- 5 To test your work, log into the client and make sure you cannot modify the account object.

Allowing Users to Add New Records in Read-Only Forms

This topic describes how to allow the user to add a new record in a read-only form. Forms should be opened in edit mode to allow adding and saving of new child records. To make a parent form read-only and allow the addition of new child records, each form field must be read only. For more information about making fields read only, see [“Making Fields Read-Only” on page 168](#).

Embedded views with child records should be left as is, i.e., they should have correspondent actions for creating and removing child records.

To allow users to add new records in read-only forms

- 1 Open IBM Domino Designer, and then open the SBL.Forms script library.

For more information, see [“Opening IBM Domino Designer” on page 111](#).

- 2 Choose the FormAccountEx class.

- 3 Add the following method to the FormAccountEx class:

```
Public Function QueryModeChange As Boolean
    QueryModeChange = Me.Form.Editmode Or Me.Form.Isnewdoc
    If QueryModeChange Then
        QueryModeChange = SBLFormEx. . QueryModeChange()
    End If
End Function
```

- 4 Optional. To allow the user to edit information for a new account that the user saved but has not synchronized, replace the code that you added in [Step 3](#) with the following code:

```
Public Function QueryModeChange As Boolean
    QueryModeChange = Me.Form.Editmode Or Not SiebelHelper.IsSynced(Me.DocumentEx)
    If QueryModeChange Then
        QueryModeChange = SBLFormEx. . QueryModeChange()
    End If
End Function
```

- 5 Test your modifications, and then republish the customization package.

For more information, see [“Republishing Customization Packages” on page 80](#).

Localizing Strings

To localize strings, you add a resource string to a resource file and then reference that string from other XML files.

To localize strings

- 1 Add a new resource string for the custom label and attribute name or warning message that you must localize. Add this resource string in the following files:

- Use the Ln_package_res.xml file for a default resource.
- Use the package_res.xx_YY.xml file for a 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.

For more information, see ["Files in the Customization Package" on page 355](#).

- 2 Specify a localizable string. You add the following code:

```
<str key="string_id">localizable_string</str>
```

where:

- *string_id* is the 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 CRM Desktop must display the string.
- You must use different formats to specify the string in different types of files. Use values from the following table.

| File Type | Description |
|---------------|--|
| Any XML file. | <p>Use the following format:</p> <pre>#string_id</pre> <p>For example:</p> <pre><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" filters_display_mode="hidden_child"></view></pre> |

| File Type | Description |
|----------------------|---|
| Any JavaScript file. | <p>Use the following format:</p> <pre>session.res_string("string_id")</pre> <p>For example:</p> <pre>ui.message_box(0, session.res_string("msg_general_error"), session.res_string("msg_general_error_capti on"), 0x40);</pre> |
| Any DXL file. | <p>Use the following format:</p> <pre>[! I oc: string_i d: I oc!]</pre> <p>For example:</p> <pre><formul a>@I f(@I sNewDoc; "[! I oc: FRM_ACCOUNT_WI NDOW_TI TLE_NEW: I o c!]"; "[! I oc: FRM_ACCOUNT_WI NDOW_TI TLE_EXI STI NG: I oc!]": " + Name)</formul a></pre> <p>where:</p> <ul style="list-style-type: none"> ■ FRM_ACCOUNT_WI NDOW_TI TLE_NEW is an example of resource Id. <p>In another example, assume the user must use a string value instead of resource Id, and that this string contains a double quote ("). For example:</p> <pre>aaa"bbb</pre> <p>In this situation, you must add a backslash (\) immediately before the double quote. For example:</p> <pre><formul a>@I f(@I sNewDoc; "aaa\"bbb"; "[! I oc: FRM_ACCOUNT_WI NDOW_T I TLE_EXI STI NG: I oc!]": " + Name)</formul a></pre> |

4 Add the XML files to the customization package.

5 Republish the customization package.

For more information, see [“Republishing Customization Packages” on page 80](#).

Validating the Data That Users Enter

This topic describes how to validate the data that the user enters in Siebel CRM Desktop. It includes the following topics:

- [Preparing to Use Validation on page 191](#)
- [Making Sure Users Enter Information in a Field on page 191](#)
- [Making Sure Users Enter Unique Values on page 192](#)
- [Making Sure Users Do Not Exceed the Maximum Number of Characters on page 193](#)
- [Creating Custom Validations on page 194](#)

A *validator* is a type of form handler that you can specify to validate the information that a user enters. You typically specify all validators in the `InitValidators` procedure of correspondent form handler class in the `SBL.Forms` script library. There are many predefined validators in the `SD3.Validator` platform script library. Custom validator classes may also be added in `SBL.Forms` library.

For more information, see [“Validation Rules You Can Configure for Custom Forms” on page 155](#).

Preparing to Use Validation

To use validation, you must make sure the `InitValidators` procedure is defined in the class that the form references.

To prepare to use validation

- 1 Open IBM Domino Designer, and then open the `SBL.Forms` script library.
For more information, see [“Opening IBM Domino Designer” on page 111](#).
- 2 Add the following code to the class that the form references. For example, use the `FormOpportunityEx` class for the Opportunity form:

```
Public Sub InitValidators(vld As Validator)
    'Add validators here
End Sub
```

Making Sure Users Enter Information in a Field

You can use validation to make sure the user enters information in a field.

To make sure the user enters information in a field

- 1 Make sure the `form_validator` object is defined.
For more information, see [“Preparing to Use Validation” on page 191](#).
- 2 Open IBM Domino Designer, and then open the `SBL.Forms` script library and then add the following code to the class that the form references:

```
Dim Fields As New ArrayEx("")
Dim Controls As New ArrayEx("")
Fields.Push "field_name"
Controls.Push "control_id"
vld.ValidateEmptyField Fields, Controls, "validation_message", ""
```

where:

- *field_name* is the name of the field you must examine.
- *controlId* is the Lotus Id of the control in the form that CRM Desktop uses in the `InitValidators` procedure.

- *validation_message* is a variable of the validation message that *CRM Desktop* displays in the client if the validation fails.

For example, the following code makes sure the user enters information in the Opportunity Name field:

```
Dim Fields As New ArrayEx("")
Fields.Push "Name"
vld.ValidateEmptyField Fields, Fields, MSG_OPPORTUNITY_NAME_REQUIRED, ""
```

For more information, see ["Opening IBM Domino Designer" on page 111](#).

- 3 Test your changes and then republish the customization package.

For more information, see ["Republishing Customization Packages" on page 80](#).

Making Sure Users Enter Unique Values

You can use validation to make sure the value that the user enters in a field is unique.

To make sure the user enters unique values

- 1 Make sure the *InitValidators* procedure is defined in the class that the form references.

For more information, see ["Preparing to Use Validation" on page 191](#).

- 2 Open IBM Domino Designer, open the SBL.Forms script library, and then add the following code to the class that the form references:

```
vld.ValidateUniqueFields fields_array_ex,
ids_array_ex, boolean_skip_empty_fields, validation_message, ""
```

where:

- *fields_array_ex* is an *ArrayEx* of fields. This array must be unique.
- *ids_array_ex* is an *ArrayEx* of control identifiers that Siebel CRM Desktop highlights in the client if the validation fails.
- *skip_empty_fields* is a parameter that determines if CRM Desktop ignores empty fields when it compares records. You can set it to one of the following values:
 - **true**. Ignore empty fields.
 - **false**. do not ignore empty fields. *validation_message* is a string constant that contains the message that CRM Desktop displays in the client if the validation fails.

For example, the following code makes sure the user enters an opportunity name that is unique for a given account:

```
vld.ValidateUniqueFields NewSmartArray().Add("Name").Add("CRMDAccountID"),
NewSmartArray().Add("Name").Add("CRMDAccountID"), False,
MSG_OPPORTUNITY_NAME_ACCOUNT_NOT_UNIQUE, ""
```

where:

- `NewSmartArray().Add("Name").Add("CRMDAccountId")` identifies the fields that, when combined, must be unique. In this example, the Name field plus the Account Id field constitutes this unique combination.
- `NewSmartArray().Add("Name").Add("CRMDAccountId")` identifies the form controls that CRM Desktop highlights in the client if the validation fails. In this example, it highlights the control for the opportunity name and the control for the account name.
- `False` instructs CRM Desktop to not ignore validation for an empty field.
- `MSG_OPPORTUNITY_NAME_ACCOUNT_NOT_UNIQUE` identifies the string constant that contains the text for the message that CRM Desktop displays in the client if the validation fails.

For more information, see ["Opening IBM Domino Designer" on page 111](#).

- 3 Test your changes and then republish the customization package.

For more information, see ["Republishing Customization Packages" on page 80](#).

Making Sure Users Do Not Exceed the Maximum Number of Characters

You can use validation to make sure the user does not enter more than a maximum number of characters in a field.

To make sure users do not exceed the maximum number of characters

- 1 Make sure the `InitValidators` procedure is defined in the class that the form references.

For more information, see ["Preparing to Use Validation" on page 191](#).

- 2 Open IBM Domino Designer, open the `SBL.Forms` script library, and then add the following code to the class that the form references:

```

vld.ValidateFieldLength fields_array_ex, ids_array_ex, maximum_length,
StrReplace(validation_message, "[comment_max_length]", maximum_length,
boolean_skip_empty_fields), ""

```

where:

- *fields_array_ex* identifies an `ArrayEx` of fields. This array must be unique.
- *ids_array_ex* identifies an `ArrayEx` of control identifiers that CRM Desktop highlights in the client if the validation fails.
- *maximum_length* specifies the maximum length for the field in characters.
- `StrReplace` substitutes the string message in the `validation_message` that contains the `comment_max_length` string with the `maximum_length` value.

- *validation_message* is a string constant that contains the message that CRM Desktop displays in the client if the validation fails.

For example, the following code makes sure the user does not enter more than 1500 characters in the Comment field of an Activity:

```
vl d. Val i dateFi el dLength NewSmartArray(). Add("Comments"),
NewSmartArray(). Add("Comments"), 1500, StrRepl ace(MSG_ACTI VI TY_COMMENT_LENGTH,
"[comment_max_l ength]", 1500, Fal se), ""
```

where:

- *Fal se* instructs CRM Desktop to not ignore validation for an empty field.

For more information, see ["Opening IBM Domino Designer" on page 111](#).

- 3 Test your changes and then republish the customization package.

For more information, see ["Republishing Customization Packages" on page 80](#).

Creating Custom Validations

You can create a custom validation.

To create custom validations

- 1 Make sure the *InitValidators* procedure is defined in the class that the form references.

For more information, see ["Preparing to Use Validation" on page 191](#).

- 2 Open IBM Domino Designer, open the SBL.Forms script library, and then add the following code to the class that the form references:

```
vl d. Val i dateCustom i ds_array_ex, val i dat i on_message, val i dat i on_cal l back, ""
```

where:

- *ids_array_ex* identifies an *ArrayEx* of control identifiers that CRM Desktop highlights in the client if the validation fails.
- *validation_message* is a string constant that contains the message that CRM Desktop displays in the client if the validation fails.
- *validation_callback* identifies the instance of the class that this code inherits from the *CallbackValidation* class. This class overrides the *Validate* function of the base class. The function gets the *ValidationContext* object as input and allows it to access data and to access the *user interface document*, which is a form in IBM Domino Designer. If the validation is successful, then it returns a value of *true*.

For more information, see ["Opening IBM Domino Designer" on page 111](#).

- 3 Test your changes and then republish the customization package.

For more information, see ["Republishing Customization Packages" on page 80](#).

Example of Creating a Custom Validation

If the user enters a new opportunity, then the following code makes sure the close date that the user enters occurs later than the current date:

```
Dim ValidationCallbackCloseDate As New
ValidationOpportunityCloseDate(NewSmartArray().Add("CloseDateOnly"))
vld.ValidateCustom NewSmartArray().Add("CloseDateOnly"),
MSG_OPPORTUNITY_CLOSE_DATE_PASSED_OUT, ValidationCallbackCloseDate, ""
```

To do the validation, this code calls the following ValidationOpportunityCloseDate class:

```
Private Class ValidationOpportunityCloseDate As CallbackValidation
    Sub New(Fields As ArrayEx)
        Set m_Fields = Fields
    End Sub

    ' Validate - validation implementation
    Function Validate(validationCtx As ValidationContext) As Boolean
        Dim Doc As DocumentEx
        Dim SavedDoc As DocumentEx
        Dim Item As Variant
        Dim EnteredDate As NotesDateTime
        Dim OriginalDate As NotesDateTime
        Dim CurrentDate As NotesDateTime
        Dim Field As String
        Dim i As Integer
        Dim DateWasNotChanged As Boolean

        Set Doc = validationCtx.DocEx
        Set SavedDoc = validationCtx.DocBackendEx
        Set CurrentDate = New NotesDateTime("")
        CurrentDate.Setnow

        Validate = True
        For i = 0 To m_Fields.Length - 1
            Field = m_Fields.Item(i)

            Set Item = Doc.FirstItem(Field)
            Set EnteredDate = New NotesDateTime(Item.Text)
            Set Item = SavedDoc.FirstItem(Field)
            DateWasNotChanged = False
            If Not Item Is Nothing Then
                Set OriginalDate = New NotesDateTime(Item.Text)
                DateWasNotChanged = (EnteredDate.TimeDifference(OriginalDate) = 0)
            End If

            If EnteredDate.IsValidDate Then
                Validate = DateWasNotChanged Or (EnteredDate.TimeDifference(CurrentDate) >= 0)
            End If

            If Validate = False Then
                Exit For
            End If
        Next i
    End Function
End Class
```

```

Next
End Function
End Class

```

Process of Adding Custom Objects

To add a custom object in Siebel CRM Desktop, you do the following:

- 1 [Creating the Custom Object on page 196](#)
- 2 [Defining Synchronization for Custom Objects on page 202](#)
- 3 [Adding Custom Views in IBM Notes on page 203](#)
- 4 [Defining the User Interface on page 204](#)
- 5 [Adding Custom Logic on page 205](#)
- 6 [Defining the Toolbar on page 206](#)

The example in this topic adds an Activity object to IBM Notes. In Siebel CRM, this object is named Action. To add this object to IBM Notes, you modify the Ln_siebel_basic_mapping.xml file. For more information about:

- Overview of XML files that you modify in this example, see [“Overview of Customizing Siebel CRM Desktop” on page 153](#).
- Details about tags in XML files that you modify in this example, see [Appendix C, “XML Files Reference”](#)

Creating the Custom Object

This task is a step in [“Process of Adding Custom Objects” on page 196](#).

In this topic, to add a new object to IBM Notes, you describe the structure of the object and then create mappings between fields, lists, and so on. You make these customizations in the Ln_siebel_basic_mapping.xml file.

To create the custom object

- 1 Use an XML editor to open the Ln_siebel_basic_mapping.xml file.
For more information, see [“Files in the Customization Package” on page 355](#).
- 2 Specify the name of the custom object. You add the following code to the Ln_siebel_basic_mapping.xml file:

```
<type id="Action" form="sblActivity">
</type>
```

This example code defines sblAction as the form to display for this object. You specify the form layout later. Create a set of fields for the custom object.

For more information, see [“Creating a Set of Fields for the Custom Object” on page 197](#).

- 3 Specify intersection objects for many-to-many relationships.

For more information, see ["Specifying the Many-To-Many Relationships" on page 198](#).

- 4 Specify the lists.

For more information, see ["Specifying the List" on page 200](#).

Creating a Set of Fields for the Custom Object

This topic describes how to create a set of fields for the custom object. In this example, you configure CRM Desktop to synchronize the Main Phone Number and the Currency Code fields of the Account form with the corresponding data for these fields that resides on the Siebel Server.

To create a set of fields for the custom object

- 1 Open IBM Domino Designer, and then add a new Main Phone Number field and a new Currency Code field on the Account form.

For more information, see ["Opening IBM Domino Designer" on page 111](#).

- 2 Use an XML editor to open the Ln_siebel_basic_mapping.xml file.

- 3 Locate the following code:

```
<type id="Account" form="sbl Account">
```

- 4 Add the following code immediately under the code you located in [Step 3](#).

```
<field ver="1" id="Description" lotus_field="Description" />
<field ver="1" id="Type" lotus_field="CRMDActionType" />
<field ver="1" id="Priority" lotus_field="Priority" />
<field ver="1" id="Primary Owner Id" lotus_field="CRMDEmployeeId" />
<field ver="1" id="Planned Start" lotus_field="StartDateTime" />
<field ver="1" id="Planned Completion" lotus_field="EndDateTime" />
<field ver="1" id="Comments" lotus_field="Comments" />
<field ver="1" id="Account Id" lotus_field="CRMDAccountId" />
<field ver="1" id="Opportunity Id" lotus_field="CRMDOpportunityId" />
```

- 5 Use an XML editor to open the siebel_meta_info.xml file.

- 6 Locate the following code:

```
<object TypeId='Account' Label='#obj_account' LabelPlural='#obj_account_plural'
UpsertBusObjCacheSize='0' EnableGetIdsBatching='true'
IntObjName='CRMDesktopAccountIO' SiebMsgXmlElemName='Account'
SiebMsgXmlCollectionElemName='ListOfCRMDesktopAccountIO'>
```

- 7 Add the following code immediately under the code you located in [Step 6](#).

```
<field Name='Main Phone Number' Label='#fld_account@main_phone_number'
DataType='DTYPE_PHONE' IOElemName='MainPhoneNumber' />

<field Name='Currency Code' Label='Currency Code' DataType='DTYPE_ID'
IsFilterable='no' IsRefObjId='yes' RefObjType='Currency'
IOElemName='CurrencyCode' />
```

Fields That Siebel CRM Desktop Uses for the Custom Object

Table 13 describes the fields you create for this example.

Table 13. Fields That Siebel CRM Desktop Uses for the Custom Object

| Field Label | Field Name | IBM Name Item | Field Type |
|--------------------|-------------------------|------------------|------------|
| Description | Description | Description | Text |
| Type | Type | CRMActionType | Picklist |
| Priority | Priority | Priority | Picklist |
| Owner | Primary Owner Id | CRMEmployeeId | Lookup |
| Account | Account Id | CRMAccountId | Lookup |
| Opportunity | Opportunity Id | CRMOpportunityId | Lookup |
| Contacts | No field on this object | Not applicable | MVG |
| Employee | No field on this object | Not applicable | MVG |
| Planned Start | Planned Start | StartDateTime | datetime |
| Planned Completion | Planned Completion | EndDateTime | datetime |
| Comments | Comment | Comments | Textarea |

Specifying the Many-To-Many Relationships

You do not specify many-to-many relationships in [Step on page 196](#). A many-to-many relationship exists between contacts and activities, and between employees and activities. You must specify 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 include a field that indicates if this intersection record is a primary record or not a primary record.

To specify the many-to-many relationships

- 1 Open IBM Domino Designer, open the Account form, and then create the MVG button.

For more information, see ["Opening IBM Domino Designer" on page 111](#).

- 2 Use an XML editor to open the Ln_siebel_basic_mapping.xml.

- 3 Add the following code:

```
<type id="Account.Contact.Association">
  <field id="ContactId" ver="1">
    <reader>
      <lotus_std>
        <lotus_field id="CRMDRightId"/>
      </lotus_std>
    </reader>
  </field>
  <convertor>
    <binary_hexstring/>
  </convertor>
</type>
```

```

    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="CRMDRightId"/>
      <convertor>
        <binary_hexstring/>
      </convertor>
    </lotus_std>
  </writer>
</field>
<field id="AccountId" ver="1">
  <reader>
    <lotus_std>
      <lotus_field id="CRMDLeftId"/>
      <convertor>
        <binary_hexstring/>
      </convertor>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="CRMDLeftId"/>
      <convertor>
        <binary_hexstring/>
      </convertor>
    </lotus_std>
  </writer>
</field>
<field id="LeftStatus" ver="1">
  <reader>
    <lotus_std>
      <lotus_field id="CRMDLeftStatus"/>
      <convertor>
        <string/>
      </convertor>
    </lotus_std>
  </reader>
</field>
<field id="RightStatus" ver="1">
  <reader>
    <lotus_std>
      <lotus_field id="CRMDRightStatus"/>
      <convertor>
        <string/>
      </convertor>
    </lotus_std>
  </reader>
</field>
</type>

```

Specifying the List

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. Each list uses a separate object to store the values for the list. 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 specify a list object, you must specify the following set of standard fields:

- Label
- Value (string)
- SortOrder (integer)
- IsDefault (Boolean)

In this example, you configuring the Sales Methods drop-down list on the Opportunity form.

To specify the list

- 1 Create a new field:
 - a Open IBM Domino Designer.
For more information, see ["Opening IBM Domino Designer" on page 111.](#)
 - b Create a field of the combobox type.
 - c Use IBM Domino Designer to specify the filtering parameters for the field you created in [Step b.](#)
- 2 Use an XML editor to open the Ln_siebel_basic_mapping.xml file.
- 3 Locate the following code:


```
<type id="Opportunity" form="sbl Opportunity">
```
- 4 Add the following code immediately following the code you located in [Step 3](#):

```
<field id="Sales Method" ver="1">
  <reader>
    <lotus_std>
      <lotus_field id="SalesMethod"/>
      <convertor>
        <string/>
      </convertor>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="SalesMethod"/>
      <convertor>
        <string/>
      </convertor>
    </lotus_std>
  </writer>
</field>
```



```

    </lotus_std>
  </writer>
</field>

```

- 5 Add the following code anywhere that is not part of code that defines an existing type:

```

<type id="OpportunitySales MethodPickList" form="sbl Const">
  <field id="Label" ver="1">
    <reader>
      <lotus_std>
        <lotus_field id="Name"/>
        <convertor>
          <string/>
        </convertor>
      </lotus_std>
    </reader>
    <writer>
      <lotus_std>
        <lotus_field id="Name"/>
        <convertor>
          <string/>
        </convertor>
      </lotus_std>
    </writer>
  </field>
  <field id="Value" ver="1">
    <reader>
      <lotus_std>
        <lotus_field id="Value"/>
        <convertor>
          <string/>
        </convertor>
      </lotus_std>
    </reader>
    <writer>
      <lotus_std>
        <lotus_field id="Value"/>
        <convertor>
          <string/>
        </convertor>
      </lotus_std>
    </writer>
  </field>
  <field id="SortOrder" ver="1">
    <reader>
      <lotus_std>
        <lotus_field id="SortOrder"/>
        <convertor>
          <number/>
        </convertor>
      </lotus_std>
    </reader>
    <writer>
      <lotus_std>

```

```

        <lotus_field id="SortOrder"/>
        <convertor>
            <number/>
        </convertor>
    </lotus_std>
</writer>
</field>
<field id="IsDefault" ver="1">
    <reader>
        <lotus_std>
            <lotus_field id="IsDefault"/>
            <convertor>
                <string_boolean/>
            </convertor>
        </lotus_std>
    </reader>
    <writer>
        <lotus_std>
            <lotus_field id="IsDefault"/>
            <convertor>
                <string_boolean/>
            </convertor>
        </lotus_std>
    </writer>
</field>
</type>

```

Defining Synchronization for Custom Objects

This task is a step in [“Process of Adding Custom Objects”](#) on page 196.

In this topic, you create 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 open the Ln_connector_configuration.xml file.

For more information, see [“Files in the Customization Package”](#) on page 355.

- 2 Create the Links section.

The Links section includes a set of fields that reference other objects. You must create these references to allow the Synchronization Engine to synchronize objects in the correct order and to download the related items. You add the following XML code to create the links:

```

<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 Create the deduplication keys.

The business environment for this example requires that activities are the same if their descriptions are the same, so you create the Description natural key. You add the following XML code to the synchronizer tag:

```

<natural_keys>
  <natural_key>
    <field>Description</field>
  </natural_key>
</natural_keys>

```

For more information, see ["Resolving Synchronization Conflicts" on page 148](#).

4 Add the following descriptions to the Ln_connector_configuration.xml file for the intersection records that you defined for contacts and employee in [Step 3 on page 197](#):

```

<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 IBM Notes

This task is a step in ["Process of Adding Custom Objects" on page 196](#).

This topic describes how to add a custom view in IBM Notes.

To add custom views in IBM Notes

- 1 Open IBM Domino Designer, choose the File menu, New, and then click the View menu item.
For more information, see [“Opening IBM Domino Designer” on page 111](#).
- 2 In the Create View dialog box, choose a View template, specify the View location, and then configure the View selection conditions.
- 3 Save your work.

Defining the User Interface

This task is a step in [“Process of Adding Custom Objects” on page 196](#).

In this topic, you define the form that Siebel CRM Desktop uses to display the custom object. You configure the custom object to use the custom sblAction form that resides in the Ln_siebel_basic_mapping.xml file that you modified in [“Creating the Custom Object” on page 196](#). You can use the standard functionality that IBM Domino Designer provides to create a new form, add the custom controls that your deployment requires, and to arrange the physical layout.

Figure 10 illustrates the layout of the Activity Form that you must create.

The screenshot shows a form titled 'Activity' with the following fields and controls:

- Description:** Text field containing 'Activity'.
- Type:** Dropdown menu set to 'Appointment'.
- Priority:** Dropdown menu.
- Status:** Dropdown menu set to 'Unscheduled'.
- Account:** Text field containing 'Macacaunt(site)'.
- Opportunity:** Text field containing 'Inventory Management System'.
- Contacts:** Dropdown menu set to 'Madison Dennison'.
- Employees:** Dropdown menu set to 'Robert Cook'.
- Start:** Date and time picker set to 10/23/2012 07:00 PM.
- End:** Date and time picker set to 10/25/2012 08:15 PM.
- Due:** Date and time picker.
- Attachments:** Section with 'Add Attachment...' and 'Remove Attachment' buttons, and a table with columns 'Attachment Name', 'Size (In Bytes)', and 'Modified'.
- Comments:** Large text area at the bottom labeled 'Comments here'.

Figure 10. Layout of the Activity Form

Adding Custom Logic

This task is a step in [“Process of Adding Custom Objects” on page 196](#).

To improve usability, sometimes you must add custom logic to Siebel CRM Desktop that allows the user to manipulate IBM Notes data. For example:

- Automatically include the current user Id in the Owner field.
- Cause the first letter of each word in the description of an activity to automatically capitalize.

CRM Desktop uses LotusScript classes to customize IBM Notes data. It runs this LotusScript on events, so you define the events that are involved and you specify the code that this script calls when an event occurs. To do this, you may modify correspondent event handlers in the form's objects tab: QueryOpen, PostOpen, QueryModeChange, PostModeChange, QuerySave, PostSave, and QueryClose.

You create a new class and name it. For example, FormActivityEx. CRM Desktop inherits the FormActivityEx class from the SBL.Forms generic script library. You then add the following methods to the FormActivityEx class:

```
Public Sub PostOpen(ui Document As NotesUI Document, id As String)
Public Function QuerySave As Boolean
```

To add custom logic

- 1 Open IBM Domino Designer, and then open the form you must modify.

For example, open a form that you created in [“Defining the User Interface” on page 204](#).

For more information, see [“Opening IBM Domino Designer” on page 111](#).

- 2 Specify the Globals:

- a Click the Objects tab, and then navigate to Globals.
- b Locate the Initialize section.
- c Add the following code to the Sub code section that resides in the section that you located in [Step b](#):

```
Set m_FormHandler = New FormActivityEx
```

This code creates a form handler. It connects the form with the corresponding class. For more information, see [“Example of the FormActivityEx Class” on page 206](#).

- d In the Options, add the following code:

```
Use SBL.Forms
```

This code connects the common SBL.Forms script library with the new class and the new form.

- e In the Declarations section, add the following code:

```
Dim m_FormHandler As FormActivityEx
```

This code declares a variable that uses the type that m_FormHandler uses.

- 3 Specify the events:

- a In the Objects tab, switch the events for the form.
- b Choose the PostOpen event and then add the following code to the Sub code section:


```
m_FormHandler.PostOpen Source, ""
```
- c Choose the QuerySave event and then add the following code to the Sub code section:


```
Continue = m_FormHandler.QuerySave
```
- d Choose the QueryOpen event and then add the following code to the Sub code section:


```
Continue = m_FormHandler.QueryOpen(Source)
```

Example of the FormActivityEx Class

The following code is an example of the FormActivityEx class:

```
Public Class FormActivityEx As SBLFormEx
    Public Sub PostOpen(uiDocument As NotesUI Document, id As String)
        'This code prefills the Owner field with the Current User Id value
        SBLFormEx..PostOpen Ui document, Id
        If Me.DocumentEx.SafeProperty("CRMEmployeeId", "") = "" Then
            Me.DocumentEx.Property("CRMEmployeeId") = GetSession().CurrentUserID
        End If
    End Sub
    Public Function QuerySave As Boolean
        'This code makes the first letter in the Description field Capital
        Dim Description As String
        Description = Me.DocumentEx.SafeProperty("Description", "")
        If Description <> "" Then
            Me.DocumentEx.Property("Description") = UCase(Left(Description, 1)) &
                Mid(Description, 2)
        End If
    End Function
End Class
```

Defining the Toolbar

This task is a step in ["Process of Adding Custom Objects" on page 196](#).

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

To define the toolbar

- 1 Open IBM Domino Designer.

For more information, see ["Opening IBM Domino Designer" on page 111](#).

- 2 In the Applications pane, expand Resources, right-click Images, and then choose New Image Resource from the shortcut menu that IBM Domino Designer displays.
- 3 In the Open dialog, locate the image file and then click Open.
- 4 Export and then import the updated resources to the customization package.
For more information on how to do this, ["Preparing the Development Environment" on page 161](#).
- 5 In IBM Domino Designer, open the following form:
(SBL)form: acti vi ty
- 6 Add a new action to the action bar for the form.
You can use a localization macro to name a new action instead of using a text label.
- 7 Do the configuration settings.
For example, in the Icons group, you can choose the Custom option, and then navigate to the icon image that you added in [Step 2](#). IBM Domino Designer displays the new action in the bar for the action and in the Objects tab.
- 8 In the Objects tab, locate the action you created, expand it, and then choose the Click event.
- 9 In the pane to the right, choose the Client option from the first drop-down list, and then choose the LotusScript option from the second drop-down list.
- 10 In the Sub code section, add the following code:

```
Dim uni d As Variant
uni d = m_FormHandl er. DocumentEx. Uni versal I D
Acti veX. Scri pt. OPEN_I N_SIEBEL uni d
```
- 11 Open the application_script.js file, and then add the OPEN_IN_SIEBEL function. Make sure you use the correct upper case and lower case for the function name.

For example:

```
functi on OPEN_I N_SIEBEL(local_i d_hex)
{
  var local_i d = appl i cati on. sessi on. hexstri ng_to_i d(local_i d_hex);
  var remote_i d = appl i cati on. synchroni zer. i d_to_remote(local_i d);
  i f (remote_i d != null )

    appl i cati on. connector. get_async_obj ect_page(remote_i d,
on_open_context_page);
  el se
    ui .message_box(0, sessi on. res_stri ng("MSG_REMOTE_NOT_FOUND"),
sessi on. res_stri ng("MSG_REMOTE_NOT_FOUND_CAPTION"), 0x40);
}
```

Removing Customizations

You can remove the currently installed customization for a user and install a customization that is currently published for this user on the Siebel Server. Removing a customization allows you to fix an error that might occur as a result of this customization. It also allows IBM Notes to synchronize with the Siebel Server.

To remove customizations

- 1 (Optional) Manually synchronize your data with the Siebel Server.

Removing customizations uninstalls the customization and deletes all data that is not currently synchronized with the Siebel Server. To save unsynchronized data, you must manually synchronize your data with the Siebel Server before you remove the customization.

- 2 In the CRM Desktop client, open the CRM Desktop - Options dialog box.

- 3 Click the General tab.

- 4 In the Customization section, click Remove.

CRM Desktop displays a dialog box that includes a message that is similar to the following:

All data that was not synchronized to Siebel will be lost. Are you sure you want to continue?

If you click Yes, then CRM Desktop does the following:

- Uninstalls the current customization.
- Deletes all data that is not currently synchronized with the Siebel Server.
- Displays the First Run Assistant and then guides you to download the customization package from the Siebel Server.

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 14](#).

Table 14. Problems That Occur When You Customize Siebel CRM Desktop

| Symptom or Error Message | Solution |
|---|--|
| <p>After you add a new field to a CRM Desktop form, CRM Desktop displays an error that is similar to the following:</p> <p>Updating error of <i>object name</i> object on storage {CEDC3D49-DDBB-93A2-4992-E5B2CAE62932}: object_locked (Conversion of input Message to Property Set failed with the error : Cannot convert XML Hi erarchy to Integrati on Object Hi erarchy. (SBL-EAI -04111)</p> <p>This error occurs if the definition of an integration object in the Siebel Runtime Repository does not match the definition of the custom CRM Desktop form.</p> | <p>Deploy your changes to the Siebel Runtime Repository.</p> |

10 Customizing Picklists

This chapter describes how to customize picklists. It includes the following topics:

- [Overview of Customizing Picklists on page 209](#)
- [Modifying the Values That Predefined Static Picklists Display on page 212](#)
- [Modifying the Values That Predefined Lists of Values Display on page 215](#)
- [Process of Creating Predefined Picklists on page 215](#)
- [Process of Creating Custom Static Picklists on page 228](#)
- [Creating Static Picklists That Use Long Values on page 234](#)
- [Process of Creating Dynamic Picklists on page 235](#)
- [Process of Creating Dynamic Picklists That Use Custom Objects on page 239](#)
- [Process of Creating Dynamic Picklists That Use a SalesBook Control on page 247](#)
- [Process of Creating Hierarchical Picklists on page 255](#)
- [Configuring Lists of Values to Support Multiple Languages on page 264](#)

Overview of Customizing Picklists

You can customize the following types of picklists in Siebel CRM Desktop:

- **Static.** Values in this picklist are constant. CRM Desktop gets these values from a simple list of values. For example, the Account Status picklist includes static values, such as Candidate, Active, or Inactive. CRM Desktop typically displays a static picklist as a dropdown list that includes static values.
- **Dynamic.** Values in this picklist vary. CRM Desktop gets these values from another business component. For example, the accounts that CRM Desktop displays vary depending on if the user picks an account for a contact or picks an account for an opportunity. In this situation, the target business component might contain different values this week compared to last week because users enter new data over time. CRM Desktop typically displays a dynamic picklist as a pick applet that includes values that change.

For more information about picklists, see *Configuring Siebel Business Applications*.

Picklist Object Structure That Siebel CRM Desktop Uses

Figure 11 illustrates a hierarchy of the objects that Siebel CRM Desktop uses to create a picklist.

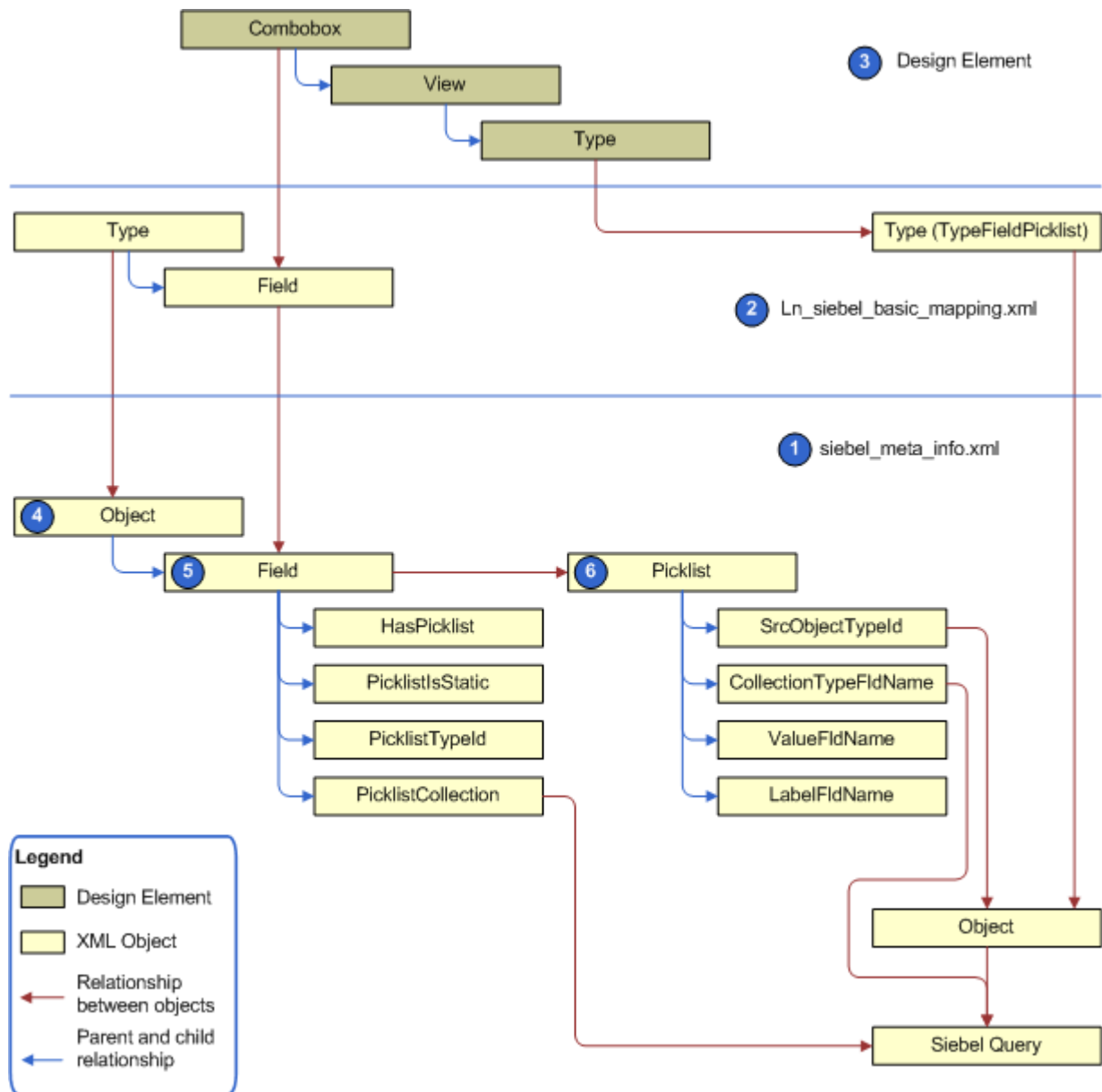


Figure 11. Picklist Object Structure That Siebel CRM Desktop Uses

Explanation of Callouts

The picklist object structure that Siebel CRM Desktop uses includes the following items:

- 1 **siebel_meta_info.xml**. Includes a representation of the field that CRM Desktop gets from the integration object in Siebel CRM. It defines the picklist object.
- 2 **Ln_siebel_basic_mapping.xml**. Maps the field that CRM Desktop gets from the integration object in Siebel CRM to the IBM Notes field. It includes the object that stores the picklist values in IBM Notes.
- 3 **Design Element. A form design element in IBM Domino Designer**. Includes the definition for the user interface. You add a control for the new field and link it to the picklist data.
- 4 **Object**. The **CRM Desktop** representation of the integration object that it receives from Siebel CRM. For example, the Opportunity Object integration object. This object includes one or more fields.
- 5 **Field**. A Siebel CRM field, such as Channel. Each field includes the following attributes for the picklist:
 - **HasPicklist**. Specifies to include a picklist for the field. The value for HasPicklist must be yes.
 - **PicklistIsStatic**. Specifies to use a static picklist.
 - **PicklistTypeId**. Specifies the name of the Picklist object in CRM Desktop.
 - **PicklistCollection**. Specifies the name of the list of values in Siebel CRM that contains the values for the picklist. For example, the value of the PicklistCollection attribute for the Channel field of the Opportunity object is OPTY_CHANNEL_TYPE.
- 6 **Picklist**. Provides an interface to the list of values. It exposes these values and associated labels and does the translation. The Picklist object includes the following attributes:
 - **SrcObjectTypeId**. References the object in the siebel_meta_info.xml file that CRM Desktop uses to get data from Siebel CRM.
 - **CollectionTypeFldName**. Defines the field in Siebel CRM that CRM Desktop queries to use the PicklistCollection property on the object level. For example, if the PicklistCollection property on the Channel field on the Opportunity object is set to OPTY_CHANNEL_TYPE, and if the CollectionTypeFldName attribute is set to List_Of_Values, Type, then CRM Desktop uses the following query:


```
[Type]='OPTY_CHANNEL_TYPE'
```
 - **ValueFldName**. Specifies the name of the field that CRM Desktop uses to get the values for the list of values from Siebel CRM.
 - **LabelFldName**. Specifies the name of the label for the field that the ValueFldName attribute identifies.

Modifying the Values That Predefined Static Picklists Display

This topic describes how to modify the values that a static picklist displays so that it only displays values that include a check mark in the Active property in Siebel Tools. The predefined Siebel CRM Desktop configuration displays values in this way only for the List_Of_Values picklist. Each of the following picklists displays a value even if the Active property of this value does not include a check mark:

- PickList_Generic
- PickList_Hierarchical
- PickList_Hierarchical_Child

The List_Of_Values picklist is the only object that automatically filters according to the Active property. To view code that illustrates this configuration, see [“About the Predefined List of Values Object” on page 231](#).

To modify the values that a predefined static picklist displays

- 1 In the client, open an activity and then click the down arrow in the Type field.
- 2 In the drop-down list, note that this list of values displays the following items:
 - Calendar Entry
 - Administration
 - Adverse Event
 - Alert
 - Analyst
 - And so on

In this example, you will modify CRM Desktop so that it does not display the Adverse Event activity type.

- 3 Open Siebel Tools and then display the Integration Object object type.
For more information, see [“Displaying Object Types in Siebel Tools” on page 163](#).
- 4 Add a field to the PickList Generic business component:
 - a In the Object Explorer, click Business Component.
 - b In the Business Components list, query the Name property for PickList Generic.
 - c In the Object Explorer, expand the Business Component tree and then click Field.

- d In the Fields list, add a new field using values from the following table.

| Property | Value |
|----------|------------|
| Name | Active |
| Column | ACTIVE_FLG |

- 5 Repeat [Step 4](#), but add the field to the PickList Hierarchical business component:
- 6 Add a field to the CRMDesktopPickListGenericIO integration object:
- a In the Object Explorer, click Integration Object.
 - b In the Integration Objects list, query the Name property for CRMDesktopPickListGenericIO.
 - c In the Object Explorer, expand the Integration Object tree and then click Integration Object Component.
 - d In the Integration Components list, query the Name property for PickList Generic.
 - e In the Object Explorer, expand the Integration Components tree and then click Integration Component Field.
 - f In the Integration Component Fields list, add a new field using values from the following table.

| Property | Value |
|--------------------|------------|
| Name | Active |
| Data Type | DTYPE_BOOL |
| External Sequence | 24 |
| External Name | Active |
| External Data Type | DTYPE_BOOL |
| XML Sequence | 24 |
| XML Tag | Active |

- 7 Repeat [Step 6](#), but add the field to the PickList Hierarchical integration component of the CRMDesktopPickListHierarchicalIO integration object.
- 8 Deploy your changes to the Siebel Runtime repository.
- 9 Use an XML editor open the siebel_meta_info.xml file.
- To make sure that CRM Desktop gets only the active list of values, you modify the PickList_Generic, PickList_Hierarchical, and PickList_Hierarchical_Child picklists. When you modify these picklists, you also modify the master filter so that it makes sure Active = Yes.
- 10 Locate the picklist TypeId='PickList_Generic' object and then modify it to the following code:

```

<picklist Typed='Picklist_Generic'
  SrcObjectType='Picklist_Generic'
  CollectionTypeFieldName='Type'
  ValueFieldName='Value'
  LabelFieldName='Value' >
  <master_filter_expr>
    <![CDATA[
      [Active] = 'Y'
    ]]>
  </master_filter_expr>
</picklist>

```

- 11** Locate the `picklist Typed='Picklist_Hierarchical'` object and then modify it to the following code:

```

<picklist Typed='Picklist_Hierarchical'
  SrcObjectType='Picklist_Hierarchical'
  CollectionTypeFieldName='Type'
  ValueFieldName='Value'
  LabelFieldName='Value'
  LangFieldName='Language' >
  <extra_src_fldname Visible='true'>Parent</extra_src_fldname>
  <master_filter_expr>
    <![CDATA[
      [Active] = 'Y'
    ]]>
  </master_filter_expr>
</picklist>

```

- 12** Locate the `picklist Typed='Picklist_Hierarchical_Child'` object and then modify it to the following code:

```

<picklist Typed='Picklist_Hierarchical_Child'
  SrcObjectType='Picklist_Hierarchical'
  CollectionTypeFieldName='Type'
  ValueFieldName='Value'
  LabelFieldName='Value'
  LangFieldName='Language' >
  <extra_src_fldname Visible='true'>Parent</extra_src_fldname>
  <master_filter_expr>
    <![CDATA[
      [Active] = 'Y' AND NOT [Parent Id] Is Null
    ]]>
  </master_filter_expr>
</picklist>

```

- 13** Save the `siebel_meta_info.xml` file, upload it to the CRM Desktop Admin screen, and then add it to the active package.
- 14** Apply the package and then synchronize.
- 15** Test your changes. Repeat [Step 2](#).

Make sure that CRM Desktop does not display the Adverse Event value in the drop-down list that the Type field uses.

Modifying the Values That Predefined Lists of Values Display

You can modify the field mapping that occurs between Siebel CRM Desktop and IBM Notes. The example in this topic modifies the mapping for the On Hold status. For more information, see [“How Siebel CRM Desktop Maps the Status Field of an Activity” on page 369](#).

To modify the values that predefined lists of values display

1 Use an XML editor open the Ln_package_res.xml file.

2 Locate the following code:

```
<str key="lang_action_status_on_hold" comment="This should be taken from Siebel SEED DATA for [Action]/[Status]">On Hold</str>
```

3 Change the code you located in [Step 2](#) to the following code:

```
<str key="lang_action_status_on_hold" comment="This should be taken from Siebel SEED DATA for [Action]/[Status]">Waiting</str>
```

Process of Creating Predefined Picklists

The example in this topic adds a predefined 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 picklist, you do the following:

- 1 [Identifying Predefined Picklist Objects in Siebel CRM on page 215](#).
- 2 [Creating an Integration Object for the Contact Method Picklist on page 217](#)
- 3 [Extending an Integration Object for the Contact Method Picklist on page 218](#)
- 4 [Adding Fields to the Customization Package on page 220](#)
- 5 [Customizing the Physical Layout for the Picklist on page 227](#)
- 6 [Publishing and Testing Picklists on page 228](#)

For more information, see [“Overview of Customizing Picklists” on page 209](#).

Identifying Predefined Picklist Objects in Siebel CRM

This task is a step in [“Process of Creating Predefined Picklists” on page 215](#).

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 drop-down list for the Contact Method field. This is the drop-down list you customize in this example.
- d** Choose the Help menu and then click About View.
The About View dialog box lists the applets in the order that Siebel Call Center displays them.
- e** Note the applet name in that Siebel Call Center displays in the Contact Method drop-down 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](#).
This applet 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 picklist.

2 Identify the picklist:

- 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 picklist 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 picklist 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 where 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 Creating Predefined Picklists” on page 215](#).

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 163](#).
- 2** Choose the File Menu and then click New Object.
- 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 | CRMDesktopPickListHierarchicalSubArea 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 CRMDesktopPickListHierarchicalSubArea, and then note the following property of the integration object you created in [Step 6](#).

| Property | Value |
|----------|--|
| XML Tag | ListOfCrmdesktoppicklisthierarchicalseubarea |

Extending an Integration Object for the Contact Method Picklist

This task is a step in [“Process of Creating Predefined Picklists”](#) on page 215.

In this topic, you extend the Contact integration component. 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 163](#).
- 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 216 . |
| 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 220 . |
| 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 216 . |

- 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 Fields to the Customization Package

This task is a step in ["Process of Creating Predefined Picklists" on page 215](#).

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 Windows command line and then navigate to the directory that contains the current files of the customization package.

For more information, see ["Files in the Customization Package" on page 355](#) and ["Using the Windows Command Line to Set Optional Parameters" on page 98](#).
 - 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** Create a working set of customization package files. You 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.dxl to v02_forms.dxl. 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:
 - a** Use an XML editor open the siebel_meta_info.xml file.

For more information, see ["Files in the Customization Package" on page 355](#).
 - b** Locate the PickList_PREFERRED_Communications object. You search for the following code:


```
<object TypeId="PickList_PREFERRED_Communications"
```

- c 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 Create the picklist. You add the following element to the siebel_meta_info.xml file:

```
<picklist Typed='PickList_Preferred_Communications' CollectionTypeFid
Name='Type' SrcObjectType='PickList_Preferred_Communications'
ValueFidName='Value' LabelFidName='Value' LangFidName='Language' >
  <master_filter_expr>
    <![CDATA[
      [Parent] = LookupValue ('OFFER_MEDIA', 'Package')
    ]]>
  </master_filter_expr>
</picklist>
```

For more information, see [“Specifying Attributes of the Pick List Element” on page 222](#).

- 4 Add the Preferred Communications field to the Contact object:

- a Locate the Contact object. You search for the following code:

```
object Typed='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" PicklistType="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 picklist between the Siebel Server and Siebel CRM Desktop for the parent Contact object:

- a Use an XML editor to open the Ln_siebel_basic_mapping.xml file.
- b Create a new object type for the picklist.

For more information, see [“Code That Creates a New Object Type for the Pick List” on page 223](#)

- c Locate the parent object. You search the Ln_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 That Creates a Map Between the Siebel Server and Siebel CRM Desktop” on page 225](#).

- 7** Add code that creates a map for the picklist 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 225](#).

- b** Copy this code to the clipboard.

- c** Locate the Account Contacts child object. Search the Ln_siebel_basic_mapping.xml file for the following code:

```
type id="Account.Contact.Association"
```

- d** Locate the Contact field. Search in the Account.Contact.Association object for the following text:

```
field id="ContactId"
```

- e** 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 225](#).

- g** Map the Preferred Communications field. You 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.

Specifying Attributes of the Pick List Element

If you specify 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' IOElementName='NoteType' />
```

In this example, you must set the TypeId attribute in the pick list element to AccountNoteType.

Table 15 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 15. 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 219](#), the `master_filter_expr` attribute constrains the values to the correct LOV Type.

Code That Creates 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" >
```

To create a new object type for a picklist, add the following code to the `Ln_siebel_basic_mapping.xml` file:

```
<type id="ContactPreferred CommunicationsPicklist">
  <field id="Label">
    <reader>
      <lotus_std>
        <lotus_field id="picklistLabel" />
        <convertor>
          <string />
        </convertor>
      </lotus_std>
    </reader>
    <writer>
      <lotus_std>
        <lotus_field id="picklistLabel" />
        <convertor>
          <string />
        </convertor>
      </lotus_std>
    </writer>
  </field>
</type>
```

```

</writer>
</field>
<field id="Value">
<reader>
<lotus_std>
<lotus_field id="picklistValue" />
<convertor>
<string />
</convertor>
</lotus_std>
</reader>
<writer>
<lotus_std>
<lotus_field id="picklistValue" />
<convertor>
<string />
</convertor>
</lotus_std>
</writer>
</field>
<field id="SortOrder">
<reader>
<lotus_std>
<lotus_field id="SortOrder" />
<convertor>
<integer />
</convertor>
</lotus_std>
</reader>
<writer>
<lotus_std>
<lotus_field id="SortOrder" />
<convertor>
<integer />
</convertor>
</lotus_std>
</writer>
</field>
<field id="IsDefault">
<reader>
<lotus_std>
<lotus_field id="IsDefault" />
<convertor>
<bool />
</convertor>
</lotus_std>
</reader>
<writer>
<lotus_std>
<lotus_field id="IsDefault" />
<convertor>
<bool />
</convertor>
</lotus_std>

```



```

    </writer>
  </field>
</type>

```

Code That Creates a Map Between the Siebel Server and Siebel CRM Desktop

To create a map between the Siebel Server and Siebel CRM Desktop, you add the following code to the Contact object of the Ln_siebel_basic_mapping.xml file:

```

<field id="Preferred Communications">
  <reader>
    <lotus_std>
      <lotus_field id="PreferredCommunications">
        </lotus_field>
        <convertor>
          <string />
        </convertor>
      </lotus_std>
    </reader>
    <writer>
      <lotus_std>
        <lotus_field id="PreferredCommunications">
          </lotus_field>
          <convertor>
            <string />
          </convertor>
        </lotus_std>
      </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 CRM Desktop automatically updates the child objects.

Example Code That Maps Child Objects for a Custom Picklist

Figure 12 illustrates the example code you use to map a child object for a custom picklist.

```

<type id="Account.Contact.Association"> ❶
...
<field id="ContactId" ver="1"> ❷
  <reader>
    <lotus_std>
      <lotus_field id="CRMRightId"/> ❸
      <converter>
        <binary_hexstring/>
      </converter>
    </lotus_std>
  </reader>
  <writer> ❹
    <multiwriter>
      <lotus_field id="CRMRightId"/>
      <linked_fields>
        <link src_field="Full Name">
          <lotus_std>
            <lotus_field id="ContactFullName"/>
            <converter>
              <string/>
            </converter>
          </lotus_std>
        </link>
        <link src_field="First Name">
          <lotus_std>
            <lotus_field id="ContactFirstName"/>
            <converter>
              <string/>
            </converter>
          </lotus_std>
        </link>
        <link src_field="Last Name">
          <lotus_std>
            <lotus_field id="ContactLastName"/>
            <converter>
              <string/>
            </converter>
          </lotus_std>
        </link>
        <link src_field="Preferred Communications"> ❺
          <lotus_std>
            <lotus_field id="ContactPreferredCommunications"/>
            <converter>
              <string/>
            </converter>
          </lotus_std>
        </link>
      </linked_fields>
    </multiwriter>
  </writer>
</field>
...
</type>

```

Figure 12. Example Code That Maps Child Objects for a Custom Picklist

Explanation of Callouts

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"ver="1"`
- 3 The following attribute identifies the matching ContactId:
`lotus_field id="CRMRightId"/`
- 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 create this field in ["Code That Creates a Map Between the Siebel Server and Siebel CRM Desktop" on page 225](#).

Customizing the Physical Layout for the Picklist

This task is a step in ["Process of Creating Predefined Picklists" on page 215](#).

Figure 13 illustrates the Contact Details section of the contact form you customize in this example. You can use IBM Domino Designer to create this layout.

The screenshot shows the 'New Contact - IBM Notes' window. The 'Contact Details' section is highlighted with a red rectangle. It includes the following fields and controls:

- Contact Name:** A text input field.
- Company Name:** A text input field.
- Job Title:** A text input field.
- Save Correspondence:** A checkbox.
- Account:** A dropdown menu.
- Contact Team:** A dropdown menu showing 'Marcos Perez'.
- Personal Addresses:** A dropdown menu.
- E-mail:** A section with four input fields for Business, Personal, Assistant, and Business 2.
- Phone Numbers:** A section with four input fields for Business, Home, Mobile, and Fax, and one for Pager.

Figure 13. Contact Details Section of the Contact Form

Publishing and Testing Picklists

This task is a step in [“Process of Creating Predefined Picklists” on page 215.](#)

In this topic, you publish and test your customization.

To publish and test a picklist

1 Publish your changes.

For more information, see [“Using the Windows Registry to Control Siebel CRM Desktop” on page 103.](#)

2 Test your changes:

- a** Open the 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 CRM Desktop changes the value in the Contact Method field to the value you choose.

Process of Creating Custom Static Picklists

To create a custom static picklist, you do the following:

- 1** [Modifying Siebel CRM Objects to Support Static Picklists on page 229](#)
- 2** [Adding Fields to the Metadata to Support Static Picklists on page 230](#)
- 3** [Adding Fields to the Basic Mapping to Support Static Picklists on page 231](#)
- 4** [Modifying the Basic Mapping to Store Values for Static Picklists on page 232](#)
- 5** [Modifying the Form to Support Static Picklists on page 233](#)
- 6** [Uploading and Testing Your Static Picklist on page 233](#)

The example in this topic adds a static picklist to the Opportunity form in Siebel CRM Desktop. Assume that a field named JVD Simple in the Opportunity business component already exists in Siebel CRM. It references a picklist named JVD Simple Picklist. [Table 16](#) describes the properties of the list of values in Siebel CRM that this picklist references. Although this topic uses a static picklist as an example, you can use this topic to create a static picklist or a multi-value static picklist.

Table 16. Properties of the List of Values That the JVD Simple Picklist References

| Type | Display Value | Translate | Language Independent Code | Language Name |
|------------|----------------|-----------|---------------------------|------------------|
| JVD_SIMPLE | Simple Value 1 | Checked | Simple Value 1 | English-American |
| JVD_SIMPLE | Simple Value 2 | Checked | Simple Value 2 | English-American |
| JVD_SIMPLE | Simple Value 3 | Checked | Simple Value 3 | English-American |

Modifying Siebel CRM Objects to Support Static Picklists

This task is a step in [“Process of Creating Custom Static Picklists” on page 228](#).

In this topic you modify Siebel CRM objects to support a static picklist.

To modify Siebel CRM objects to support a static picklist

- 1 Open Siebel Tools.
- 2 In the Object Explorer, click Integration Object.
- 3 In the Integration Objects list, query the name property for CRMDesktopOpportunityIO.
- 4 In the Object Explorer, expand the Integration Object tree and then click Integration Component.
- 5 In the Integration Components list, query the External Name Context property for Opportunity.
- 6 In the Object Explorer, expand the Integration Component tree and then click Integration Component Field.
- 7 In the Integration Component Fields list, add a new field using values from the following table.

| Property | Value |
|--------------------|------------|
| Name | JVD Simple |
| Data Type | DTYPE_TEXT |
| Length | 30 |
| External Sequence | 232 |
| External Name | JVD Simple |
| External Data Type | DTYPE_TEXT |

| Property | Value |
|--------------|-----------|
| XML Sequence | 232 |
| XML Tag | JVDSimple |

- 8 In the Object Explorer, expand the Integration Component Field tree and then click Integration Component Field User Prop.
- 9 In the Integration Component Field User Props list, add a new user property using values from the following table.

| Property | Value |
|----------|----------|
| Name | PICKLIST |
| Value | Y |

- 10 Deploy your changes to the Siebel Runtime Repository.

Adding Fields to the Metadata to Support Static Picklists

This task is a step in [“Process of Creating Custom Static Picklists” on page 228](#).

In this topic you add a field to the metadata to support a static picklist.

To add a field to the metadata to support a static picklist

- 1 Use an XML editor open the siebel_meta_info.xml file.
- 2 Locate the following object:
TypeId="Opportunity"
- 3 Add a new field to the Opportunity object. You add the following code immediately following the code line you located in [Step 2](#):

```
<field Name='JVDSimple' Label='JVDSimple' DataType='DTYPE_TEXT'
IsFilterable='no' HasPicklist='yes' PicklistIsStatic='yes'
PicklistCollectionType='JVD_SIMPLE' PicklistTypeId='List_Of_Values'
IsHidden='no' IObjectName='JVDSimple' />
```

where:

- **Field Name** is any name you choose. It is recommended that you use the same name that you use in Siebel CRM for this field.
- **Label** is equal to the value you provide for the Name.
- **DataType** is the data type you specify for the integration component field in [Step 7 on page 229](#).
- **IsFilterable** is set to the default of no. IsFilterable is not relevant for this example.

- **HasPicklist** must be set to yes.
- **PicklistIsStatic** must be set to yes.
- **PicklistCollectionType** is the name of the list of values.
- **PicklistTypeId** identifies the name of the picklist type for static picklists. For more information, see [“About the Predefined List of Values Object” on page 231](#).
- **IsHidden** must be set to no.
- **IOElemName** identifies the name you specified for the XML Tag property in [Step 7 on page 229](#).

About the Predefined List of Values Object

The following List_of_Values picklist object comes predefined with Siebel CRM Desktop. It determines the fields to get, the field type, where to query for the language, and it makes the list of values active. It is similar to the Picklist Generic business component in Siebel CRM:

```
<picklist Typed='List_of_Values'
  SrcObjectType='List_of_Values'
  CollectionTypeName='Type'
  ValueFieldName='Value'
  LabelFieldName='Value'
  LangFieldName='Language' >
  <master_filter_expr>
    <![CDATA[
      [Active] = 'Y'
    ]]>
  </master_filter_expr>
</picklist>
```

Adding Fields to the Basic Mapping to Support Static Picklists

This task is a step in [“Process of Creating Custom Static Picklists” on page 228](#).

In this topic you add a field to the basic mapping to support a static picklist.

To add a field to the basic mapping to support a static picklist

- 1 Use an XML editor to open the Ln_siebel_basic_mapping.xml file.
- 2 Locate the following object:


```
id="Opportunity"
```
- 3 Add the following code immediately following the code line you located in [Step 2](#):

```
<field id="JVD Simple">
  <reader>
    <lotus_std>
```

```

        <lotus_field id="JVDSimple" />
        <convertor>
            <string />
        </convertor>
    </lotus_std>
</reader>
<writer>
    <lotus_std>
        <lotus_field id="JVDSimple" />
        <convertor>
            <string />
        </convertor>
    </lotus_std>
</writer>
</field>

```

where:

- **field id** identifies the name of the integration component field you defined in [Step 7 on page 229](#).
- **user_field id** identifies the custom field in IBM Notes where Siebel CRM Desktop stores the field value. It is recommended that you use sbl as the first three characters for this field Id. For example, sbl JVD Simple.

Modifying the Basic Mapping to Store Values for Static Picklists

This task is a step in [“Process of Creating Custom Static Picklists” on page 228](#).

In this topic you modify the basic mapping to store values for a static picklist.

To modify the basic mapping to store values for a static picklist

- 1 Locate a predefined static list of values.

The predefined List_Of_Values object in the metadata makes sure that Siebel CRM Desktop can get the values that it displays in the dropdown list. It gets these values from Siebel CRM. It must store these values in IBM Notes to allow the user to work offline. To do this, it is recommended that you use one of the static list of values that comes predefined in CRM Desktop. These objects typically contain the same fields.

For example, locate the following object:

```
id='ContactStatusPicklist'
```

- 2 Copy the entire ContactStatusPicklist object.
- 3 Paste the entire ContactStatusPicklist object immediately following the object you located in [Step 1](#).
- 4 Change the following values:


```
<type id="type_Id field_Id Picklist" predefined_folder="1" ver="1">
<form message_class="IPM.Contact.SBL.type_Id field_Id"></form>
```

where:

- *type_Id* identifies the Id of the type, such as Opportunity.
- *field_Id* identifies the Id of the field you specified in [Step 3 on page 231](#), such as JVD Simple.

In the `message_class` attribute, replace any spaces that exist in the `type_Id` and the `field_Id` with an underscore (`_`).

For example:

```
<type id="OpportunityJVD SimplePicklist" predefined_folder="1" ver="1">
<form message_class="IPM.Contact.SBL.OpportunityJVD_SimplePicklist"></form>
```

If you do not correctly identify the type Id, then CRM Desktop does not display any picklist values in the dropdown list at run time.

The code in this book does not include the entire object. It includes only the items you must change for this example.

Modifying the Form to Support Static Picklists

This task is a step in [“Process of Creating Custom Static Picklists” on page 228](#).

In this topic you add fields to the form to support a static picklist.

To modify the form to support a static picklist

- Open IBM Domino Designer, and then do the following work to modify the form:
 - Add a field to the form.
 - Configure the field as a combobox.
 - Enter the list of predefined elements or apply the formula that the form requires.

For more information, see [“Opening IBM Domino Designer” on page 111](#).

Uploading and Testing Your Static Picklist

This task is a step in [“Process of Creating Custom Static Picklists” on page 228](#).

In this topic you upload and test your static picklist.

To upload and test your static picklist

- 1 Publish the following files that you modified:
 - `siebel_meta_info.xml`

■ Ln_siebel_basic_mapping.xml

For more information, see [“Using the Windows Registry to Control Siebel CRM Desktop” on page 103](#).

- 2 Open the client and then navigate to the Opportunity form.
- 3 Make sure this form includes a new field named JVD Simple.
- 4 Make sure the JVD Simple field includes a picklist and that this picklist includes the values listed in the Display Value column in [Table 16 on page 229](#).

Creating Static Picklists That Use Long Values

The Display Value property of a list of values in Siebel CRM is limited to 30 characters. If you use the Display Value property as the source of the values that Siebel CRM Desktop displays in a static picklist, then each of these value cannot exceed 30 characters in length. The Description property of a list of values in Siebel CRM can contain up to 255 characters. The example in this topic describes how to use this Description property to create a list of values that includes values that are longer than 30 characters in length.

The example in this topic assumes that a field named JVD Simple in the Opportunity business component already exists in Siebel CRM and that this field references a picklist named JVD Simple Picklist. [Table 16 on page 229](#) describes the properties of the list of values that this picklist references. In addition, [Table 17](#) describes values for the Description property.

Table 17. Properties of the List of Values That the JVD Simple Picklist References for Long Values

| Type | Display Value | Description |
|------------|----------------|--------------------------------|
| JVD_SIMPLE | Simple Value 1 | Description for Simple Value 1 |
| JVD_SIMPLE | Simple Value 2 | Description for Simple Value 2 |
| JVD_SIMPLE | Simple Value 3 | Description for Simple Value 3 |

To create a static picklist that uses long values

- 1 Do all the work described in [“Process of Creating Custom Static Picklists” on page 228](#) with the following modifications:
 - In [Step 3 on page 230](#) you set the following:


```
PicklistTypeId='ListOfValues_Description'
```
 - Do not do [“Uploading and Testing Your Static Picklist” on page 233](#) at this time.
- 2 Use an XML editor open the siebel_meta_info.xml file.
- 3 Locate and then make a copy of the predefined list of values.

For more information, see [“About the Predefined List of Values Object” on page 231](#).

- 4 Modify the copy you made in [Step 3](#) to the following:

```
<picklist Typed='List_Of_Values_Description'  
  SrcObjectType='List_Of_Values'  
  CollectionTypeFieldName='Type'  
  ValueFieldName='Description'  
  LabelFieldName='Description'  
  LangFieldName='Language' >  
  <master_filter_expr>  
    <![CDATA[  
      [Active] = 'Y'  
    ]]>  
  </master_filter_expr>  
</picklist>
```

Note the following:

- Bold text indicates the values that you must modify for this example.
 - You must change the value in the Typed attribute to a unique value, such as List_Of_Values_Description.
 - The predefined list of values includes a ValueFieldName attribute that is set to Value and a LabelFieldName attribute that is set to Value. You must change the values of these attributes so that they reference the Description property.
- 5 Do [“Uploading and Testing Your Static Picklist” on page 233](#) with the following modification:
- Make sure the JVD Simple field includes a picklist and that this picklist includes the values listed in the Description column in [Table 17 on page 234](#).

Process of Creating Dynamic Picklists

This topic describes how to create a dynamic picklist that uses predefined objects that already exist in Siebel CRM. You do the following work to create a dynamic picklist:

- 1 [Modifying Siebel CRM Objects to Support Dynamic Picklists on page 235](#)
- 2 [Modifying the Metadata, Basic Mapping, and Forms to Support Dynamic Picklists on page 237](#)

A dynamic picklist does not get values from the List Of Values table. It gets values from a business component that resides in Siebel CRM. The example in this topic creates a dynamic picklist that changes values according to the opportunity that the user chooses. For more information, see [“Overview of Customizing Picklists” on page 209](#).

Modifying Siebel CRM Objects to Support Dynamic Picklists

This task is a step in [“Process of Creating Dynamic Picklists” on page 235](#).

In this topic you modify Siebel CRM objects to support a dynamic picklist.

To modify Siebel CRM objects to support a dynamic picklist

- 1 Open Siebel Tools.
- 2 In the Object Explorer, click Business Component.
- 3 In the Business Components list, query the name property for Opportunity.
- 4 In the Object Explorer, expand the Business Component tree and then click Business Component Field.
- 5 In the Fields list, add a new field using values from the following table.

| Property | Value |
|----------|-------------------|
| Name | JVD Industry |
| Join | S_INDUST |
| Column | NAME |
| Type | DTYPE_TEXT |
| Picklist | PickList Industry |

- 6 In the Fields list, add a new field using values from the following table.

| Property | Value |
|----------|-----------------|
| Name | JVD Industry Id |
| Join | S_OPTY_X |
| Column | ATTRIB_03 |
| Type | DTYPE_TEXT |

- 7 Extend the integration object with the new industry field. Do [“Modifying Siebel CRM Objects to Support Static Picklists” on page 229](#) with the following modifications:
 - In the Integration Component Fields list, add a new field using values from the following table.

| Property | Value |
|--------------------|--------------|
| Name | JVD Industry |
| Data Type | DTYPE_TEXT |
| Length | 50 |
| External Sequence | 237 |
| External Name | JVD Industry |
| External Data Type | DTYPE_TEXT |

| Property | Value |
|--------------|-------------|
| XML Sequence | 237 |
| XML Tag | JVDIndustry |

- Do not add an integration component field user property.

Modifying the Metadata, Basic Mapping, and Forms to Support Dynamic Picklists

This task is a step in [“Process of Creating Dynamic Picklists” on page 235](#).

In this topic you add fields to the metadata, basic mapping, and forms to support a dynamic picklist.

To modify the metadata, basic mapping, and forms to support a dynamic picklist

- 1 Add the picklist field to the metadata. Do [“Adding Fields to the Metadata to Support Static Picklists” on page 230](#) with the following modifications:

- a Add the following code:

```
<field Name=' JVD Industry' Label =' JVD Industry' DataType=' DTYPE_TEXT'
HasPicklist=' yes' PicklistsStatic=' yes' PicklistType=' PickList_Industry'
IOFieldName=' JVDIndustry' />
```

Note that this code does not include the `CollectionType` attribute because a dynamic picklist does not use the list of values that a static picklist uses. It also includes the `PickList_Industry` picklist type.

- b Create the new `PickList_Industry` picklist type. You add the following code immediately after the code you added in [Step a](#):

```
<picklist Type=' PickList_Industry'
SrcObjectType=' Industry' ValueFieldName=' Name'
LabelFieldName=' Name' />
```

This picklist definition is not as complex as the picklist definition you use to create a static picklist. A dynamic picklist does not use a list of values and does not require the `CollectionTypeFieldName` attribute. This example uses the predefined `Industry` object to get the picklist values.

- 2 Add the JVD Industry field to the basic mapping. Do [“Adding Fields to the Basic Mapping to Support Static Picklists” on page 231](#), but use the following code:

```
<field id="JVD Industry">
<reader>
<lotus_std>
<lotus_field id="JVDIndustry" />
<convertor>
<string />
</convertor>
```

```

    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="JVDI ndustry" />
      <convertor>
        <string />
      </convertor>
    </lotus_std>
  </writer>
</field>

```

Bold text identifies the code that is different for a dynamic picklist.

- 3 Create the type that stores the list of values. Do [“Modifying the Basic Mapping to Store Values for Static Picklists” on page 232](#), but use the following code:

```

<type id="Opportuni tyJVD I ndustryPi ckl i st"
predefi ned_fol der="1" ver="1">
  <form message_cl ass="I PM. Contact. SBL. Opportuni tyJVD_I ndustryPi ckl i st"></form>

```

Bold text identifies the code that is different for a dynamic picklist. For more information about doing this step, see [Step 4 on page 232](#).

- 4 Open IBM Domino Designer, and then add the field to the following form:

(SBL) form: opportuni ty

For more information, see [“Opening IBM Domino Designer” on page 111](#).

- 5 Upload and publish your work. Do [“Uploading and Testing Your Static Picklist” on page 233](#).

- 6 Test your work:

- a Open the client and then navigate to the opportunity form.
- b Make sure the form displays the Industry picklist.
- c Choose the Industry picklist and then make sure it displays the appropriate values.

The picklist must display all the industries that are appropriate for the opportunity that Siebel CRM Desktop currently displays. For example:

- Banks
- Basic Materials
- Beef Cattle Feedlots
- Beef Cattle except feedlots
- Berry Crops
- And so on

Process of Creating Dynamic Picklists That Use Custom Objects

This topic describes how to create a dynamic picklist that uses custom objects that you define. You do the following work to add a dynamic picklist that uses custom objects:

- 1 [Modifying the Business Component on page 239](#)
- 2 [Creating an Integration Object on page 240](#)
- 3 [Modifying Siebel CRM Desktop to Support the New Integration Object on page 243](#)
- 4 [Modifying the Remaining Siebel CRM Desktop Objects on page 245](#)

For more information, see [“Overview of Customizing Picklists” on page 209](#).

Modifying the Business Component

This task is a step in [“Process of Creating Dynamic Picklists That Use Custom Objects” on page 239](#).

In this topic you modify the Opportunity business component to support a dynamic picklist that uses custom objects.

To modify the business component

- 1 Open Siebel Tools.
- 2 In the Object Explorer, click Business Component.
- 3 In the Business Components list, query the name property for Opportunity.
- 4 In the Object Explorer, expand the Business Component tree and then click Business Component Field.
- 5 In the Fields list, add a new field using values from the following table.

| Property | Value |
|----------|------------------------------|
| Name | JVD Fulfillment Center |
| Join | S_ORG_FUL |
| Column | NAME |
| Type | DTYPE_TEXT |
| Picklist | eAuto PickList Business Rule |

- 6 In the Fields list, add a new field using values from the following table.

| Property | Value |
|----------|---------------------------|
| Name | JVD Fulfillment Center Id |
| Join | S_OPTY_X |
| Column | ATTRIB_06 |
| Type | DTYPE_TEXT |

Creating an Integration Object

This task is a step in [“Process of Creating Dynamic Picklists That Use Custom Objects” on page 239](#).

The Fulfillment Center is not available as an object in predefined Siebel CRM Desktop. To make it available, you create an integration object.

To create an integration object

- 1 In Siebel Tools, choose the File menu and then click New Object.
- 2 In the New Object Wizards dialog box, click the EAI tab, click Integration Object, and then click OK.
- 3 In the Integration Object Builder dialog box, set values using information from the following table and then click Next.

| Property | Value |
|------------------|---|
| Project | Choose the project you use for this development effort. |
| Business Service | EAI Siebel Wizard |

- 4 In the next screen of the Integration Object Builder dialog box, set values using information from the following table and then click Next.

| Property | Value |
|-------------------------|-------------------------------|
| Source Object | Fulfillment Center |
| Source Root | Fulfillment Center |
| Integration Object Name | CRMDesktopFulfillmentCenterIO |

- 5 In the Integration Object Builder - Choose Integration Components dialog box, do the following:
 - a Expand the Fulfillment Center tree.
 - b Remove the check mark from the Fulfillment Center_Position check box.

- c Click Next and then click Finish.

Siebel Tools creates and then displays the new integration object.

- 6 (Optional) Make the XML Tag property consistent with the other integration objects that Siebel CRM Desktop uses. You change properties using values from the following table.

| Property | Value |
|----------|-------------------------------------|
| XML Tag | ListOfCRMDesktopFulfillmentCenterIO |

- 7 Make fields that Siebel CRM Desktop does not require inactive:
 - a In the Object Explorer, expand the Integration Object tree, expand the Integration Component Tree, and then click Integration Component Field.
 - b In the Integration Component Fields list, set the Inactive property to True for each of the following fields:
 - ❑ Description
 - ❑ Main Fax Number
 - ❑ Main Phone Number
 - ❑ Primary Position Id
 - ❑ UIActive
 - ❑ UISelected
 - ❑ operation
 - ❑ searchspec

The Integration Object Builder wizard creates an integration component that includes all fields that the business component includes, by default. You can remove the fields that Siebel CRM Desktop does not require to make web service calls more efficient.

- 8 In the Integration Component Fields list, add a new field using values from the following table.

| Property | Value |
|--------------------|----------------|
| Name | DS Updated |
| Data Type | DTYPE_DATETIME |
| Length | 30 |
| External Sequence | 10 |
| External Name | DS Updated |
| External Data Type | DTYPE_DATETIME |
| XML Sequence | 10 |
| XML Tag | DBLastUpd |

Each integration component that Siebel CRM Desktop uses must include the DS Updated field.

- 9 In the Object Explorer, click Integration Component Key.

Although the Integration Object Builder wizard creates a key for the integration component that it creates, Siebel CRM Desktop requires more keys to support the synchronization process.

- 10 Create the modification key:

- a In the Integration Component Keys list, add a new key using values from the following table.

| Property | Value |
|---------------------|------------------|
| Name | Modification Key |
| Key Sequence Number | 1 |
| Key Type | Modification Key |

- b In the Object Explorer, expand the Integration Component Key and then choose Integration Component Key Field.

- c In the Integration Component Key Fields list, create two new fields using values from the following table.

| Name | Field Name |
|-----------|------------|
| DBLastUpd | DS Updated |
| Mod Id | Mod Id |

- 11 Create the primary key:

- a** In the Integration Component Keys list, add a new key using values from the following table.

| Property | Value |
|---------------------|-------------|
| Name | Primary Key |
| Key Sequence Number | 1 |
| Key Type | User Key |

- b** In the Integration Component Key Fields list, add a new field using values from the following table.

| Name | Field Name |
|------|------------|
| Id | Id |

12 Create the status key:

- a** In the Integration Component Keys list, add a new key using values from the following table.

| Property | Value |
|---------------------|------------|
| Name | Status Key |
| Key Sequence Number | 1 |
| Key Type | Status Key |

- b** In the Integration Component Key Fields list, add four new fields using values from the following table.

| Name | Field Name |
|-----------|------------|
| DBLastUpd | DS Updated |
| Id | Id |
| Mod Id | Mod Id |
| Name | Name |

13 Deploy your changes to the Siebel Runtime Repository.

Modifying Siebel CRM Desktop to Support the New Integration Object

This task is a step in [“Process of Creating Dynamic Picklists That Use Custom Objects”](#) on page 239.

If you expose a new object to Siebel CRM Desktop, then you must create a new type in the Ln_connector_configuration.xml file. This file defines the objects that Siebel CRM Desktop synchronizes. For more information, see [“Files in the Customization Package” on page 356](#).

To modify Siebel CRM Desktop to support the new integration object

- 1 Use an XML editor open the Ln_connector_configuration.xml file and then add the following code:

```
<type id="Ful fillment Center">
  <view label="Ful fillment Center"
    label_plural="Ful fillment Centers" 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="[: (Name):]"
    frequency="604800" threshold="0">
    <links></links>
  </synchronizer>
</type>
```

- 2 Create a new metadata type to support the new integration object. You use an XML editor open the siebel_meta_info.xml file and then add the following code:

```
<object TypeId=' Ful fillment Center' Label=' Ful fillment Center'
Label Plural=' Ful fillment Centers' ViewMode=' All' EnableGetIDsBatching=' true'
IntObjName=' CRMDesktopFul fillmentCenterIO'
SiebMsgXml ElemName=' Ful fillmentCenter'
SiebMsgXml CollectionElemName=' ListOfCRMDesktopFul fillmentCenterIO' >

<prohibit_operation AnyMod=' yes' ErrMsg=' #mod_operation_is_prohibited_err_msg' />

<extra_command_options>
  <option Name=' PrimaryKey1M' Value=' Id' />
  <option Name=' ForeignKey1M' Value=' Id' />
  <option Name=' Cardinality' Value=' 1M' />
  <option Name=' ServerServiceVersion' Value=' 2' />
</extra_command_options>

<field Name=' Conflict Id' Label=' Conflict Id' DataType=' DTYPE_ID'
IsFilterable=' no' IsHidden=' yes' IOElemName=' ConflictId' />

<field Name=' Created' Label=' Created' DataType=' DTYPE_DATETIME'
IOElemName=' Created' />

<field Name=' Created By' Label=' Created By' DataType=' DTYPE_ID' IsFilterable=' no'
IsHidden=' yes' IOElemName=' CreatedBy' />

<field Name=' DS Updated' Label=' DS Updated' DataType=' DTYPE_DATETIME'
IsFilterable=' no' IsHidden=' yes' IsTimestamp=' yes' IOElemName=' DBLastUpd' />

<field Name=' Id' Label=' Id' IsPrimaryKey=' yes' DataType=' DTYPE_ID'
IsFilterable=' no' IsHidden=' yes' IsPartOfUserKey=' yes' IOElemName=' Id' />

<field Name=' Mod Id' Label=' Mod Id' DataType=' DTYPE_ID' IsFilterable=' no'
```

```

IsHidden=' yes'  IsObjectName=' Modified'  />

<field Name=' Name'  Label =' Name'  DataType=' DTYPE_TEXT'  IsPartOfUserKey=' yes'
IsObjectName=' Name'  />

<field Name=' Updated'  Label =' Updated'  DataType=' DTYPE_DATETIME'  IsHidden=' yes'
IsObjectName=' Updated'  />

<field Name=' Updated By'  Label =' Updated By'  DataType=' DTYPE_ID'  IsFilterable=' no'
IsHidden=' yes'  IsObjectName=' UpdatedBy'  />
</object>

```

- 3 Create the basic mapping object to support the new integration object. You use an XML editor to open the Ln_siebel_basic_mapping.xml file and then add the following code:

```

<type id="Fulfillment Center">
  <field id="Name">
    <reader>
      <lotus_std>
        <lotus_field id="LastName" />
        <convertor>
          <string />
        </convertor>
      </lotus_std>
    </reader>
    <writer>
      <lotus_std>
        <lotus_field id="LastName" />
        <convertor>
          <string />
        </convertor>
      </lotus_std>
    </writer>
  </field>
</type>

```

When Siebel CRM Desktop synchronizes data it uses this code to determine where to store mapping data in IBM Notes.

Modifying the Remaining Siebel CRM Desktop Objects

This task is a step in [“Process of Creating Dynamic Picklists That Use Custom Objects” on page 239](#).

The remaining tasks you must perform are nearly identical to the tasks you perform to add a dynamic picklist. The main difference is you must use the JVD Fulfillment Center field name. For more information, see [“Process of Creating Dynamic Picklists” on page 235](#).

To modify the remaining Siebel CRM Desktop objects

- 1 Extend the integration object with the new industry field. Do [“Modifying Siebel CRM Objects to Support Static Picklists” on page 229](#) with the following modifications:

- In the Integration Component Fields list, add a new field using values from the following table.

| Property | Value |
|--------------------|------------------------|
| Name | JVD Fulfillment Center |
| Data Type | DTYPE_TEXT |
| Length | 50 |
| External Sequence | 237 |
| External Name | JVD Fulfillment Center |
| External Data Type | DTYPE_TEXT |
| XML Sequence | 237 |
| XML Tag | JVDFulfillmentCenter |

- Do not add an integration component field user property.
- 2 Add the picklist field to the metadata. Do [“Adding Fields to the Metadata to Support Static Picklists” on page 230](#) with the following modifications:

- a Add the following code:

```
<field Name=' JVD Ful fillment Center' Label =' JVD Ful fillment Center'
DataType=' DTYPE_TEXT'
HasPicklist=' yes' PicklistStatic=' yes' PicklistType=' PickList_Ful fillmentC
enter' IObjectID=' JVDFul fillmentCenter' />
```

- b Create the new PickList_Industry picklist type. You add the following code immediately after the code you added in [Step a](#):

```
<picklist Type=' PickList__Ful fillmentCenter'
SrcObjectType=' Ful fillment Center' ValueFieldName=' Name'
LabelFieldName=' Name' />
```

- 3 Add the JVD Industry field to the basic mapping. Do [“Adding Fields to the Basic Mapping to Support Static Picklists” on page 231](#), but use the following code:

```
<field id="JVD Ful fillment Center">
  <reader>
    <lotus_std>
      <lotus_field id="JVDFul fillmentCenter" />
      <convertor>
        <string />
      </convertor>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="JVDFul fillmentCenter" />
      <convertor>
        <string />
      </convertor>
    </lotus_std>
  </writer>
</field>
```

```

        </convertor>
      </lotus_std>
    </writer>
  </field>

```

Bold text identifies the code that is different for a dynamic picklist.

- 4 Create the type that stores the list of values. Do [“Modifying the Basic Mapping to Store Values for Static Picklists” on page 232](#), but use the following code:

```

<type id="OpportunityJVD FulfillmentCenterPicklist"
predefined_folder="1" ver="1">
  <form
    message_class="IPM.Contact.SBL.OpportunityJVD_FulfillmentCenterPicklist"></
  form>

```

Bold text identifies the code that is different for a dynamic picklist. For more information about doing this step, see [Step 4 on page 232](#).

- 5 Open IBM Domino Designer, open the (SBL)form:opportunity form, and then create the JVD Fulfillment Center combobox.

Configure the combobox to use the following source:

```
OpportunityJVD FulfillmentCenter
```

For more information, see [“Opening IBM Domino Designer” on page 111](#).

- 6 Upload and publish your work. Do [“Uploading and Testing Your Static Picklist” on page 233](#).

- 7 Test your work:

- a Open the client and then navigate to the opportunity form.
- b Make sure the form displays the Fulfillment Center picklist.
- c Choose the Fulfillment Center picklist and then make sure it displays the appropriate values.

The picklist must display all the fulfillment centers that are appropriate for the opportunity that Siebel CRM Desktop currently displays. For example:

- Concord Fulfillment Center
- Copy Center
- Marketing Department
- And so on

Process of Creating Dynamic Picklists That Use a SalesBook Control

This topic describes how to create a dynamic picklist that uses a salesbook control. You do the following work to add a dynamic picklist that uses a salesbook control:

- 1 [Modifying Siebel CRM Objects to Support a Dynamic Picklist That Uses a SalesBook Control on page 248](#)

- 2 [Modifying the Metadata on page 249](#)
- 3 [Modifying the Basic Mapping and Connector Configuration on page 250](#)
- 4 [Modifying the Business Logic and Testing Your Work on page 252](#)

For more information, see [“Overview of Customizing Picklists” on page 209](#).

Modifying Siebel CRM Objects to Support a Dynamic Picklist That Uses a SalesBook Control

This task is a step in [“Process of Creating Dynamic Picklists That Use a SalesBook Control” on page 247](#).

In this topic you modify Siebel CRM objects to support a dynamic picklist that uses a salesbook control.

To modify Siebel CRM objects to support a dynamic picklist that uses a salesbook control

- 1 Do all the work described in [“Process of Creating Dynamic Picklists That Use Custom Objects” on page 239](#).

To add a dynamic picklist that uses a salesbook control, you reuse a number of the objects that you configure when you create a dynamic picklist that uses custom objects.

- 2 Activate the following integration component fields that you deactivated in [Step 7 on page 241](#):

- Description
- Main Fax Number
- Main Phone Number

- 3 Set the Data Type property for the following integration component fields to DTYPE_TEXT:

- Main Fax Number
- Main Phone Number

If the Data Type property for these fields is not DTYPE_TEXT, then Siebel CRM Desktop cannot synchronize data for these fields.

- 4 Add a new integration component field to the CRMDesktopFulfillmentCenterIO integration object using values from the following table.

| Property | Value |
|--------------------|---------------------------|
| Name | JVD Fulfillment Center Id |
| Data Type | DTYPE_ID |
| External Sequence | 239 |
| External Name | JVD Fulfillment Center Id |
| External Data Type | DTYPE_ID |

| Property | Value |
|--------------|------------------------|
| XML Sequence | 239 |
| XML Tag | JVDFulfillmentCenterId |

- 5 Deploy your changes to the Siebel Runtime Repository.

Modifying the Metadata

This task is a step in [“Process of Creating Dynamic Picklists That Use a SalesBook Control” on page 247](#).

In this topic you modify the metadata.

To modify the metadata

- 1 Open the siebel_meta_info.xml file and then add the following new fields to the Fulfillment Center type:

```
<field Name='Description' Label='Description' DataType='DTYPE_TEXT'
IsPartOfUserKey='no' IOElementName='Description' />
```

```
<field Name='Main Fax Number' Label='Main Fax Number' DataType='DTYPE_PHONE'
IsPartOfUserKey='no' IOElementName='MainFaxNumber' />
```

```
<field Name='Main Phone Number' Label='Main Phone Number' DataType='DTYPE_PHONE'
IsPartOfUserKey='no' IOElementName='MainPhoneNumber' />
```

For more information about adding fields to the metadata, see [“Adding Fields to the Metadata to Support Static Picklists” on page 230](#).

- 2 Remove the JVD Fulfillment Center field definition that you added in [Step 2 on page 246](#).

This field represents data that Siebel CRM gets through a join. The Fulfillment Center object includes this data so it is not necessary to synchronize it from the opportunity object.

- 3 Add a field definition for the JVD Fulfillment Center Id field. You add the following code:

```
<field Name='JVD Fulfillment Center Id' Label='JVD Fulfillment Center Id'
DataType='DTYPE_ID' IsFilterable='no' IsRefObjId='yes' RefObjTypeId='Fulfillment
Center' RefExposedToUI='no' IOElementName='JVDFulfillmentCenterId' />
```

where:

- **IsRefObjId** specifies that CRM Desktop uses this field to create a relation with another object in CRM Desktop.
- **RefObjTypeId** specifies the type of object that includes the relation. For example, Fulfillment Center.
- **RefExposedToUI** instructs CRM Desktop to display the related object in the client as a separate object.

Modifying the Basic Mapping and Connector Configuration

This task is a step in [“Process of Creating Dynamic Picklists That Use a SalesBook Control” on page 247](#).

In this topic you modify the basic mapping and connector configuration.

To modify the basic mapping and connector configuration

- 1 Allow Siebel CRM Desktop to store values for the new fields you added in [Step 1 on page 249](#). You add the following code to the Fulfillment Center object in the Ln_siebel_basic_mapping.xml file. This code maps telephone number fields to existing IBM Notes fields and creates a custom field for the Description field:

```
<field id="Main Phone Number">
  <reader>
    <lotus_std>
      <lotus_field id="BusinessTelephoneNumber" />
      <convertor>
        <string />
      </convertor>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="BusinessTelephoneNumber" />
      <convertor>
        <string />
      </convertor>
    </lotus_std>
  </writer>
</field>
<field id="Main Fax Number">
  <reader>
    <lotus_std>
      <lotus_field id="BusinessFaxNumber" />
      <convertor>
        <string />
      </convertor>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="BusinessFaxNumber" />
      <convertor>
        <string />
      </convertor>
    </lotus_std>
  </writer>
</field>
<field id="Description">
  <reader>
    <lotus_std>
```

```

        <lotus_field id="Description" />
        <convertor>
            <string />
        </convertor>
    </lotus_std>
</reader>
<writer>
    <lotus_std>
        <lotus_field id="Description" />
        <convertor>
            <string />
        </convertor>
    </lotus_std>
</writer>
</field>

```

- 2 Locate the following type definition for the opportunity object:

```
<type id="Opportunity"
```

- 3 Add the following code to the object you located in [Step 2](#):

```

<field id="JVD Fulfillment Center Id">
    <reader>
        <lotus_std>
            <lotus_field id="JVDFulfillmentCenterID" />
            <convertor>
                <binary_hexstring />
            </convertor>
        </lotus_std>
    </reader>
    <writer>
        <multiwriter>
            <lotus_field id="JVDFulfillmentCenterID" />
            <linked_fields>
                <link_src_field id="Name" dest_field="JVD Fulfillment Center">
                    <lotus_std>
                        <convertor>
                            <string />
                        </convertor>
                    </lotus_std>
                </link>
            </linked_fields>
        </multiwriter>
    </writer>
</field>

```

The writer statement in this code includes a link field that allows CRM Desktop to get the value for the JVD Fulfillment Center field in the Opportunity object from the Name field of the Fulfillment Center object. This is similar to how Siebel CRM uses a pick map. If Siebel CRM gets values for multiple fields through a join, then you can add multiple fields in the link_fields section. These link_fields must exist in the basic mapping but you do not need to add them to the metadata.

- 4 Use an XML editor open the Ln_connector_configuration.xml file.

- 5 Locate the following definition for the Opportunity object:

```
<type id="Opportunity">
```

- 6 Add the following code to the definition you located in [Step 5](#):

```
<link>JVD Fulfillment Center Id</link>
```

This code specifies the JVD Fulfillment Center Id field as a link on the Opportunity object. In the metadata you specify that the JVD Fulfillment Center Id field is related to another object in CRM Desktop. CRM Desktop uses that relation during synchronization.

Modifying the Business Logic and Testing Your Work

This task is a step in ["Process of Creating Dynamic Picklists That Use a SalesBook Control" on page 247](#).

In this topic you modify the business logic and test your work.

To modify the business logic and test your work

- 1 Add a direct link description in the Business logic:
 - a Open the SBL.BusinessLogic script library in the IBM Domino Designer.
 - b Locate the CreateMetaScheme function.

This function sets up the relationships between objects.
 - c Locate the Opportunity section in the CreateMetaScheme function.
 - d Create the link definition for the Fulfillment Center object relation with the Opportunity object.

The following helper function should be used:

```
MetaSchemeHelper.AddDirectLink(linkFrom As String, linkTo As String, fieldName As String, fieldStatusName As String, tag As String, removeOnUnlink as Boolean)
```

where:

- *linkFrom* specifies the object type where the link field resides.
- *linkTo* specifies the object type that the link field references.
- *fieldName* specifies the name of the field which stores the ID for the linked object.
- *fieldStatusName* specifies the name of the field which stores a Status of linked object (Unsaved or Deleted).
- *tag* specifies the tag name for link.
- *removeOnUnlink* specifies that the linked document should be removed in case of unlinking from the parent object (True) or remains in the local storage as a separate object (False).

In this example, you add the following code:

```
Call Helper.AddDirectLink("Opportunity", "Fulfillment Center",
"JVDFulfillmentCenterId", "ObjectStatus", LINK_TAG_DIRECT, False);
```

e Save and close the SBL.BusinessLogic script library.

2 Create a new view design element for SalesBook and lookup purposes:

a Create a new view with the following name in IBM Domino Designer:

(SBL)\salesbook:FulfillmentCenters

b Enter the following View Selection formula:

```
SELECT CRMDType = "Fulfillment Center"
```

c Modify parameters of the first view column:

Title: display_name

Sort: None

Column Value: @Trim(@ReplaceSubstring(Name; "|" : @NewLine; "|" : " ")) + "|" +
@Text(@DocumentUniqueID)

Hide Column: yes

d Add the second column with the following parameters:

Title: Fulfillment Center Name

Sort: Ascending

Column Value: Name

Hide Column: no

e Enter the following code into the Queryopendocument event:

```
Sub Queryopendocument(Source As Notesui view, Continue As Variant)
Continue = False
End Sub
```

f Save and close the view design element.

3 Add the picklist control to the Opportunity form handler:

a Open the SBL.Forms script library in the IBM Domino Designer.

b Locate the PostOpen procedure in the FormOpportunityEx class.

c Input the following code anywhere before calling InitDescriptors:

```
Dim FulfillmentController As New
DescriptorControllerLookup("JVDFulfillmentCenterId", "Fulfillment Center",
LINK_TAG_DIRECT, "sbFulfillmentButton", "(SBL)\salesbook:FulfillmentCenters ",
"")
With FulfillmentController
.Caption = "Fulfillment Centers"
.ChooseFrom = "Select from Fulfillment Centers"
End With
Set Me.addDescriptor = FulfillmentController
```

d Save and close the SBL.Forms library

4 Add the picklist control to the Opportunity form layout:

- a Open the (SBL)form:opportunity form in IBM Domino Designer.
- b Add a Fulfillment Center label where you need it on the form layout.
- c Add a new field with the following parameters:

Name: JVDFullmentCenterId
Editable: yes
Type: Dialog List
Show field delimiters: no
Allow values not in list: no

- d Input the following Choices formula for this field:

```
sblist:= @DbColumn("": "NoCache"; ""; "(SBL)\salesbook: FulfillmentCenters ";
1);
thisV:= @ThisValue;
sblistsize:= 2 + 2 * @Elements(sblist) + 2 * @Sum(@Length(sblist)) -
@Sum(@Length(@Ascii(sblist; [AllInRange])));
maxlistsiz:= 64000;
@if(sblistsize > maxlistsiz; @True; @Return(sblist));
thisLabel:= @If(thisV = ""; ""; @Trim(@Left(sblist; thisV)));
nmax:= @Max(@Length(@Trim(@Left(sblist; "|"))));
n:= 0;
@DoWhile(n:= n + 1;
abcLabel:= @If(thisV = ""; ""; @Left(thisLabel; n));
abcN:= @If(thisV = ""; 1; n + 1);
abclist:= @Unique(@Left(sblist; abcN));
abclist:= @If(thisV = ""; abclist; @Transform(abclist; "entry";
@if(@Matches(entry; @Left(abcLabel; 1) + "*" ); entry; @Nothing));
plist:= @Transform(sblist; "entry"; @If((abcLabel != "") & @Matches(entry;
abcLabel + "*"); entry; @IsMember(@Left(entry; abcN); abclist); @Do(abclist
:= @Replace(abclist; @Left(entry; abcN); @Nothing); entry; @Nothing));
plistsize:= 2 + 2 * @Elements(plist) + 2 * @Sum(@Length(plist)) -
@Sum(@Length(@Ascii(plist; [AllInRange])));
(plistsize > maxlistsiz) & (n < nmax));
plist
```

- e Enter the following code into the Onchange event of the field:

```
Sub Onchange(Source As Field)
Call cOnChange(m_FormHandler, "JVDFullmentCenterId")
End Sub
```

- f Add the new button with the following IBM Notes script code in the Click event:

```
Sub Click(Source As Button)
Call cOnClick(m_FormHandler, "sbFullmentButton")
End Sub
```

- g Save and close the Opportunity form.

5 Upload and publish your work. Do ["Uploading and Testing Your Static Picklist"](#) on page 233.

6 Test your work:

- a Open the client and then navigate to the Opportunity form.

b Make sure the form displays the Center picklist.

c Type text into the Center picklist.

As you enter each character of text the Center picklist must display different values in the autocomplete entries. These entries must change according to the character you enter. For example, if you enter the letter B, then the field must display entries that begin with the letter B.

d Click the button that opens the SalesBook dialog box.

CRM Desktop must display the SalesBook dialog box. It must include a list of fulfillment centers. For example:

- ❑ Concord Fulfillment Center
- ❑ Copy Center
- ❑ Marketing Department
- ❑ And so on

Process of Creating Hierarchical Picklists

To create a hierarchical picklist, you do the following:

- 1 [Modifying Siebel CRM Objects to Support Hierarchical Picklists on page 256](#)
- 2 [Modifying the Metadata to Support Hierarchical Picklists on page 258](#)
- 3 [Modifying the Basic Mapping and Forms to Support Hierarchical Picklists on page 259](#)
- 4 [Linking Fields and Testing Your Hierarchical Picklist on page 261](#)

This topic describes how to configure Siebel CRM Desktop to display a hierarchical picklist. In this example, two fields use static picklists. The values displayed in the second picklist depend on the value that the user chooses in the first picklist. For more information about the hierarchical picklist, see *Configuring Siebel Business Applications*.

The example in this topic adds a hierarchical picklist to the Opportunity form in CRM Desktop. Assume that a hierarchical list of values named JVD_HIER already exists in Siebel CRM. [Table 18](#) describes the properties of this hierarchical list of values.

Table 18. Properties of the JVD_HIER Hierarchical List of Values

| Type | Display Value | Parent LIC |
|----------|---------------------|------------|
| JVD_HIER | Child 1 of Parent 1 | Parent 1 |
| JVD_HIER | Child 1 of Parent 2 | Parent 2 |
| JVD_HIER | Child 2 of Parent 1 | Parent 1 |
| JVD_HIER | Child 2 of Parent 2 | Parent 2 |

Table 18. Properties of the JVD_HIER Hierarchical List of Values

| Type | Display Value | Parent LIC |
|----------|---------------|----------------|
| JVD_HIER | Parent 1 | Not applicable |
| JVD_HIER | Parent 2 | Not applicable |

Modifying Siebel CRM Objects to Support Hierarchical Picklists

This task is a step in [“Process of Creating Hierarchical Picklists” on page 255](#).

In this topic you modify Siebel CRM objects to support a hierarchical picklist.

To modify Siebel CRM objects to support a hierarchical picklist

- 1 Open Siebel Tools.
- 2 In the Object Explorer, click Pick List.
- 3 In the Picklists list, create two new picklists using values from the following table.

| Name | Business Component | Search Specification |
|---------------------------|--------------------|----------------------|
| JVD Hierarchical - Child | PickList Generic | Not applicable |
| JVD Hierarchical - Parent | PickList Generic | '[Parent] Is Null' |

- 4 In the Object Explorer, click Business Component.
- 5 In the Business Components list, query the Name property for Opportunity.
- 6 In the Object Explorer, expand the Business Component tree and then click Field.
- 7 In the Fields list, create two new fields using values from the following table.

| Name | Join | Column | Type | Picklist |
|-----------------|----------|-----------|------------|---------------------------|
| JVD Hier Child | S_OPTY_X | ATTRIB_47 | DTYPE_TEXT | JVD Hierarchical - Child |
| JVD Hier Parent | S_OPTY_X | ATTRIB_46 | DTYPE_TEXT | JVD Hierarchical - Parent |

- 8 Extend the integration object to expose the new field:
 - a In the Object Explorer, click Integration Object.
 - b In the Integration Objects list, query the name property for CRMDesktopOpportunityIO.
 - c In the Object Explorer, expand the Integration Object tree and then click Integration Component.
 - d In the Integration Components list, query the External Name Context property for Opportunity.

- e In the Object Explorer, expand the Integration Component tree and then click Integration Component Field.
- f In the Integration Component Fields list, add a new field using values from the following table.

| Property | Value |
|--------------------|-----------------|
| Name | JVD Hier Parent |
| Data Type | DTYPE_TEXT |
| Length | 30 |
| External Sequence | 233 |
| External Name | JVD Hier Parent |
| External Data Type | DTYPE_TEXT |
| XML Sequence | 233 |
| XML Tag | JVDHierParent |

- g In the Object Explorer, expand the Integration Component Field tree and then click Integration Component Field User Prop.
- h In the Integration Component Field User Props list, add a new user property using values from the following table.

| Property | Value |
|----------|----------|
| Name | PICKLIST |
| Value | Y |

- i In the Object Explorer, click Integration Component Field.
- j In the Integration Component Fields list, add a new field using values from the following table.

| Property | Value |
|--------------------|----------------|
| Name | JVD Hier Child |
| Data Type | DTYPE_TEXT |
| Length | 30 |
| External Sequence | 234 |
| External Name | JVD Hier Child |
| External Data Type | DTYPE_TEXT |
| XML Sequence | 234 |
| XML Tag | JVDHierChild |

- k In the Object Explorer, click Integration Component Field User Prop.

- I In the Integration Component Field User Props list, add a new user property using values from the following table.

| Property | Value |
|----------|----------|
| Name | PICKLIST |
| Value | Y |

- 9 Deploy your changes to the Siebel Runtime Repository.

Modifying the Metadata to Support Hierarchical Picklists

This task is a step in ["Process of Creating Hierarchical Picklists" on page 255](#).

In this topic you modify the metadata to support a hierarchical picklist.

To modify the metadata to support a hierarchical picklist

- 1 Use an XML editor open the siebel_meta_info.xml file.
- 2 Locate the following object:
TypeId="Opportunity"
- 3 Add new fields to the Opportunity object. You add the following code immediately following the object you located in [Step 2](#):

```
<field Name='JVD_Hier_Parent' Label='JVD_Hier_Parent'
DataType='DTYPE_TEXT' HasPicklist='yes'
PicklistStatus='yes'
PicklistCollectionType='JVD_HIER'
PicklistType='List_of_Values_Parent'
ObjectName='JVD_HierParent' />

<field Name='JVD_Hier_Child' Label='JVD_Hier_Child'
DataType='DTYPE_TEXT' HasPicklist='yes'
PicklistStatus='yes'
PicklistCollectionType='JVD_HIER'
PicklistType='Picklist_Hierarchical_Child'
ObjectName='JVD_HierChild' />
```

- 4 Locate the following object:
Picklist TypeId="Picklist_Hierarchical "
- 5 Add the child picklist. Add the following code after the object you located in [Step 4](#):

```
<picklist
TypeId='Picklist_Hierarchical_Child'
SrcObjectType='Picklist_Hierarchical '
```

```

CollectionTypeFldName=' Type'
ValueFldName=' Value'
LabelFldName=' Value'
LangFldName=' Language' >
<extra_src fldname Visible=' true' >
  Parent
</extra_src fldname>
<master_filter_expr>
  <![CDATA[
    NOT [Parent Id] Is Null
  ]]>
</master_filter_expr>
</picklist>

```

The following filter makes sure that Siebel CRM Desktop gets only the four values that are applicable as child values:

```
NOT [Parent Id] Is Null
```

The following field sets up filtering for items that you configure later in this procedure:

```

<extra_src fldname Visible=' true' >
  Parent
</extra_src fldname>

```

- 6 Add the parent picklist. You add the following code:

```

<picklist
  TypeId=' List_Of_Values_Parent'
  SrcObjectType=' List_Of_Values'
  CollectionTypeFldName=' Type'
  ValueFldName=' Value'
  LabelFldName=' Value'
  LangFldName=' Language' >
  <master_filter_expr>
    <![CDATA[
      [Active] = 'Y' AND [Parent Id] Is Null
    ]]>
  </master_filter_expr>
</picklist>

```

The following filter makes sure that Siebel CRM Desktop gets the values that are allowed as parent values:

```
[Active] = 'Y' AND [Parent Id] Is Null
```

Modifying the Basic Mapping and Forms to Support Hierarchical Picklists

This task is a step in [“Process of Creating Hierarchical Picklists” on page 255](#).

In this topic you modify the basic mapping and forms to support a hierarchical picklist.

To modify the basic mapping and forms to support a hierarchical picklist

- 1 Add the parent field to the basic mapping. Do [“Adding Fields to the Basic Mapping to Support Static Picklists” on page 231](#), but use the following code:

```
<field id="JVD Hier Parent">
  <reader>
    <lotus_std>
      <lotus_field id="JVDHierParent" />
      <convertor>
        <string />
      </convertor>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="JVDHierParent" />
      <convertor>
        <string />
      </convertor>
    </lotus_std>
  </writer>
</field>
```

- 2 Add the child field to the basic mapping. Do [“Adding Fields to the Basic Mapping to Support Static Picklists” on page 231](#), but use the following code:

```
<field id="JVD Hier Child">
  <reader>
    <lotus_std>
      <lotus_field id="JVDHierChild" />
      <convertor>
        <string />
      </convertor>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="JVDHierChild" />
      <convertor>
        <string />
      </convertor>
    </lotus_std>
  </writer>
</field>
```

- 3 Create the type that stores the list of values for the parent picklist. Do [“Modifying the Basic Mapping to Store Values for Static Picklists” on page 232](#), but use the following code:

```
<type id="OpportunityJVD Hier ParentPicklist">
```

Bold text identifies the code that is different for a hierarchical picklist. For more information about doing this step, see [Step 4 on page 232](#).

- 4 Create the type that stores the list of values for the child picklist. Do [“Modifying the Basic Mapping to Store Values for Static Picklists” on page 232](#), but use the following code:

```
<type id="OpportunityJVD Hier ChildPicklist"
predefined_folder="1" ver="1">
<form message_class="IPM.Contact.SBL.OpportunityJVD_Hier_ChildPicklist"></form>
```

Bold text identifies the code that is different for a hierarchical picklist. For more information about doing this step, see [Step 4 on page 232](#).

- 5 Allow the child picklist to store the value from the parent field. You add the following code to the OpportunityJVD Hier ChildPicklist type that you created in [Step 4 on page 260](#):

```
<field id="Parent">
  <reader>
    <lotus_std>
      <lotus_field id="Parent" />|
      <convertor>
        <string />
      </convertor>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="Parent" />
      <convertor>
        <string />
      </convertor>
    </lotus_std>
  </writer>
</field>
```

- 6 Open IBM Domino Designer, and then open the following form:

(SBL)form: opportunity

For more information, see ["Opening IBM Domino Designer" on page 111](#).

- 7 Create the JVD Hier Parent combobox.
- 8 Create the JVD Hier Child combobox.

Linking Fields and Testing Your Hierarchical Picklist

This task is a step in ["Process of Creating Hierarchical Picklists" on page 255](#).

If the user chooses a value in the parent picklist, then this value filters the values that CRM Desktop displays in the child picklist. To set up this relationship in:

- **Siebel Tools.** You set up a constraint on the pickmap.
- **Siebel CRM Desktop.** You write a Lotus Script code in the Opportunity form handler and Opportunity form.

This code implements the business logic for the Opportunity form. The SBL.Forms library includes similar code for each form that Siebel CRM Desktop displays.

If you do not establish this relationship between the parent picklist and the child picklist, then the parent dropdown list displays all parent values as options and the child dropdown list displays all child values.

To link fields and test your hierarchical picklist

1 Open the SBL.Forms script library in IBM Domino Designer.

2 Add a new class:

```
' /**
' * Callback class for clearing fields
' */
Public Class CallbackClearFields As CallbackObject
Private m_formEx As FormEx
Private m_fieldNames As ArrayEx

' /**
' * Constructor
' * @param formEx - current formEx object
' * @param fieldNames - names of fields which must be cleared
' */
Sub New(formEx As FormEx, fieldNames As ArrayEx)
Set m_formEx = formEx
Set m_fieldNames = fieldNames
End Sub

' /**
' * Invokes callback: clear all defined fields
' * @param params - arguments (not used)
' * @return Variant - result of invoking (not used)
' */
Public Function invoke(params As DynamicArguments) As Variant
On Error Goto catch
Dim i As Integer
Dim control As ControlEditable

For i = 0 To m_fieldNames.Length - 1
Set control = m_formEx.FormManager.GetControl(Cstr(m_fieldNames.Item(i)))

If Not control Is Nothing Then
control.Value = ""
Else
m_formEx.DocumentEx.Property(Cstr(m_fieldNames.Item(i))) = ""
End If
Next
Call m_formEx.Form.Refresh()

finally:
Exit Function
catch:
LogMsg "CallbackClearFields.invoke", "Exception " & Err & ": " & Error & " in
line " & Cstr(Erl), LOG_LEVEL_ERROR
```

```

        Resume finally
    End Function
End Class

```

- 3 Locate the InitControls procedure in the FormOpportunityEx class and add the following code:

```

Dim controlChannelType As New ControlEditable("JVDHierParent", Me.Form)
Set callback = New CallbackClearFields(Me, NewSmartArray.Add("JVDHierChild"))
Call controlChannelType.OnChange.Connect(callback)
Call Me.FormManager.RegisterControl(controlChannelType)

```

- 4 Save and close the SBL.Forms script library.

- 5 Open the (SBL)form:opportunity form in IBM Domino Designer.

- 6 Locate the JVDHierParent field and input the following code in the Onchange event:

```

Sub Onchange(Source As Field)
    Call cOnChange(m_FormHandler, "JVDHierParent")
End Sub

```

- 7 Locate the JVDHierChild field and modify the following parameter properties:

Refresh choices on document refresh - yes
 Note that field's choices parameters should contain valid @-formulas to make them dependent.

Example of JVDHierParent field's choices formula:

```

key:="JVD Hier Parent";
@DbLookup("": "NoCache"; ""; "SBLPicklistsOpportunity"; key; 3; [FailSilent])

```

Example JVDHierChild field's choices formula:

```

key := JVDHierParent;
@DbLookup("": "NoCache"; ""; "SBLPicklistsOpportunity"; key; 3; [FailSilent])

```

where:

SBLPicklistsOpportunity - name or alias of the view which contains correspondent picklist records

- 8 Save and close the Opportunity form.

- 9 Upload and publish your work. Do ["Uploading and Testing Your Static Picklist"](#) on page 233.

- 10 Test your work:

- a Open the client and then navigate to the Opportunity form.
- b Make sure the form displays the following picklists:
 - ❑ Hier Parent
 - ❑ Hier Child
- c Choose the Hier Parent picklist and make sure it includes the following values:
 - ❑ Parent 1
 - ❑ Parent 2

- d** Choose Parent 1 in the Hier Parent picklist.
Siebel CRM Desktop should automatically change the value for the Hier Child picklist to Child 1 of Parent 1.
- e** Choose the Hier Child picklist and make sure it includes the following values:
 - ❑ Child 1 of Parent 1
 - ❑ Child 2 of Parent 1
- f** Choose Parent 2 in the Hier Parent picklist.
Siebel CRM Desktop should automatically change the value for the Hier Child picklist to Child 1 of Parent 2.
- g** Choose the Hier Child picklist and make sure it includes the following values:
 - ❑ Child 1 of Parent 2
 - ❑ Child 2 of Parent 2

Configuring Lists of Values to Support Multiple Languages

The predefined Siebel CRM Desktop runs in the language that IBM Notes uses when you install CRM Desktop. This configuration works if IBM Notes and the Siebel object manager use the same language. For example, you can run an English IBM Notes client that connects to an English object manager, or a German IBM Notes client that connects to a German object manager. This configuration does not work in a Siebel deployment that uses multiple languages. For example, you cannot use the predefined configuration to run an English object manager with English IBM Notes clients and German IBM Notes clients. This topic describes how to configure CRM Desktop so that you can run an English object manager with multiple languages.

To configure lists of values to support multiple languages

- 1** Open Siebel Tools.
- 2** Create the new business components that will handle the multilingual values:
 - a** In the Object Explorer, click Business Component.
 - b** In the Business Components list, locate the List Of Values business component.
 - c** Click Edit and then click Copy Record.

- d Set properties for the copy you made in [Step c](#) using values from the following table.

| Property | Value |
|----------------------|---|
| Name | CRM Desktop List Of Values |
| Search Specification | <p>([Class Code] <> 'CLASS' OR [Class Code] IS NULL) AND [Language]= Language()</p> <p>This search specification makes sure this business component only returns values where the language of these values match the language that the object manager uses.</p> |

- e In the Business Components list, locate the PickList Hierarchical business component.
- f Click Edit and then click Copy Record.
- g Set properties for the copy you made in [Step f](#) using values from the following table.

| Property | Value |
|----------------------|-----------------------------------|
| Name | CRM Desktop PickList Hierarchical |
| Search Specification | [Language] = Language() |

- 3 Create business objects for the business components that you created in [Step 2](#):

- a In the Object Explorer, click Business Object.
- b Right-click in the Business Objects list, click New Record, and then set properties using values from the following table.

| Property | Value |
|----------|----------------------------|
| Name | CRM Desktop List Of Values |

- c In the Object Explorer, expand the Business Object tree and then click Business Object Component.
- d Right-click in the Business Object Components list, click New Record, and then set properties using values from the following table.

| Property | Value |
|----------|----------------------------|
| Name | CRM Desktop List Of Values |

- e Right-click in the Business Objects list, click New Record, and then set properties using values from the following table.

| Property | Value |
|----------|-----------------------------------|
| Name | CRM Desktop PickList Hierarchical |

- f Right-click in the Business Object Components list, click New Record, and then set properties using values from the following table.

| Property | Value |
|----------|-----------------------------------|
| Name | CRM Desktop PickList Hierarchical |

4 Update the integration objects:

- a In the Object Explorer, click Integration Object.
- b In the Integration Objects list, locate the CRMDesktopListOfValuesIO integration object and then modify the properties of this integration object using value from the following table.

| Property | Value |
|----------|----------------------------|
| Name | CRM Desktop List Of Values |

- c In the Object Explorer, expand the Integration Object tree and then click Integration Component.
- d In the Integration Components list, query the Name property for the following value:

List Of Values

- e Modify the properties of the integration component you located in [Step d](#) using value from the following table.

| Property | Value |
|-----------------------|----------------------------|
| Name | CRM Desktop List Of Values |
| External Name Context | CRM Desktop List Of Values |
| External Name | CRM Desktop List Of Values |

- f In the Integration Objects list, locate the CRMDesktopPickListHierarchicalIO integration object and then modify the properties of this integration object using value from the following table.

| Property | Value |
|---------------|-----------------------------------|
| External Name | CRM Desktop PickList Hierarchical |

- g In the Integration Components list, query the Name property for the following value:

PickList Hierarchical

- h** Modify the properties of the integration component you located in [Step g](#) using value from the following table.

| Property | Value |
|-----------------------|-----------------------------------|
| Name | CRM Desktop PickList Hierarchical |
| External Name Context | CRM Desktop PickList Hierarchical |
| External Name | CRM Desktop PickList Hierarchical |

- 5** Deploy your changes to the Siebel Runtime Repository
- 6** Update the CRM Desktop package so that it does not get the list of values in a certain language but instead accepts the language that the Siebel object manager returns:
 - a** Use an XML editor open the siebel_meta_info.xml file.
 - b** Remove the LangFldName='Language' attribute from the following PickList objects:
 - ❑ StatePickList
 - ❑ PickList_Hierarchical
 - ❑ List_Of_Values
 - ❑ AccountNoteType
 - ❑ RevenueType

11 Customizing Multi-Value Groups

This chapter describes how to customize multi-value groups. It includes the following topics:

- [Process of Creating MVG Fields on page 269](#)

Process of Creating MVG Fields

This topic describes how to create an MVG (multi-value group) field. You do the following work to add an MVG field:

- 1 [Identifying Predefined MVG Objects in Siebel CRM on page 269](#)
- 2 [Process of Making Siebel CRM Data Available to Add an MVG on page 271](#)
- 3 [Process of Modifying the Customization Package to Add an MVG on page 276](#)
- 4 [Publishing and Testing a Custom MVG Field on page 279](#)

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 IBM Notes_view control.

The example in this topic adds an MVG control to the Opportunity form. This MVG displays an association between an opportunity and channel partners.

For more information about MVGs, see *Configuring Siebel Business Applications*.

Identifying Predefined MVG Objects in Siebel CRM

This task is a step in [“Process of Creating MVG Fields” on page 269](#).

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 click About View.

The About View dialog box lists the applets in the order in that Siebel Call Center displays them.

f Note the applet name that Siebel Call Center uses to display 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 Projects in Siebel Tools” on page 163](#).

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 | <p>S_OPTY_ORG</p> <p>If the Inter Table property:</p> <ul style="list-style-type: none"> ■ 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 | <p>Channel Partner</p> <p>The business component in the Child Business Component property must be displayed.</p> |

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 Creating MVG Fields” on page 269](#).

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 272](#)
- 2** [Creating an Integration Component for the Channel Partner MVG on page 273](#)
- 3** [Extending an Integration Object for the Primary Id Field on page 275](#)

This topic describes how to make sure Siebel CRM data is available to Siebel CRM Desktop. Some Siebel CRM data is available without customizing 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 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 271](#).

In this topic, you create a new integration object that makes the channel partner data available to the EAI Siebel Adapter. The work you do 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 27](#).

To create an integration object for the channel partner MVG

- 1 In Siebel Tools, choose the File Menu and then click New Object.
- 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 271](#).

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 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 163](#).
- 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.
CRM Desktop adds the CRMDesktopOpportunityIO integration object to the repository when you install CRM Desktop on the Siebel Server. You must install it 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 220. |
| XML Sequence | 10002 For more information, see “Requirements for the Sequence Property” on page 220. |
| 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 271.

In this topic, you make the Primary Id field available to Siebel CRM Desktop. You make the primary on the opportunity available so that 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 276](#).

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 272](#).

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 220 . |
| XML Sequence | 139 For more information, see “Requirements for the Sequence Property” on page 220 . |
| 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 Creating MVG Fields” on page 269](#).

To modify the customization package to add an MVG, you do the following:

- 1 [Adding a Custom Object on page 276](#)
- 2 [Adding the MVG Link on page 277](#)
- 3 [Adding the Primary Field on page 277](#)
- 4 [Adding a Field on page 278](#)
- 5 [Customizing the Validation Message and Labels on page 279](#)

Adding a Custom Object

This task is a step in [“Process of Modifying the Customization Package to Add an MVG” on page 276](#).

To add a custom object to Siebel CRM Desktop, you display the object in Siebel CRM and then modify customization package XML files. The example in this topic makes available and then adds 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 272.](#)
- 2 Add a custom object type. You modify the siebel_meta_info.xml file.
For more information, see ["Code That Adds a Custom Object Type" on page 280.](#)
- 3 Map objects. You modify the Ln_siebel_basic_mapping.xml file.
For more information, see ["Code That Maps a Custom Object" on page 281.](#)
- 4 Configure synchronization for the custom object. You modify the Ln_connector_configuration.xml file.

For more information, see ["Code That Configures Synchronization for a Custom Object" on page 282.](#)

Adding the MVG Link

This task is a step in ["Process of Modifying the Customization Package to Add an MVG" on page 276.](#)

In this topic, you add the MVG link to the business logic file.

To add the MVG link

- 1 Use a Domino Designer to open the SBL.BusinessLogic script library.
For more information, see ["Files in the Customization Package" on page 355](#)
- 2 Locate the following function:
`CreateMetaScheme`
- 3 Add the code to add a new association.
For more information, see ["Code That Adds a New Association" on page 282.](#)

Adding the Primary Field

This task is a step in ["Process of Modifying the Customization Package to Add an MVG" on page 276.](#)

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 Add the primary field. You do the following:

a Use an XML editor 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='Channel Partner' IOElementName='PrimaryPartnerId' />
```

c Save and then close the siebel_meta_info.xml file.

d Use an XML editor to open the Ln_siebel_basic_mapping.xml file:

e Add the following code to the Opportunity type tag:

```
<field id="Primary Partner Id">
  <reader>
    <lotus_std>
      <lotus_field id="PrimaryPartnerId" />
      <converter>
        <binary_hexstring />
      </converter>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="PrimaryPartnerId" />
      <converter>
        <binary_hexstring />
      </converter>
    </lotus_std>
  </writer>
</field>
```

f Save and then close the Ln_siebel_basic_mapping.xml file.

2 Add the link:

a Use an XML editor open the Ln_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 Ln_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 276.

This topic describes how to add the ChannelPartnerStatus field.

To add a field

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

2 Add the following code to the Opportunity.Channel_Partner.Association type:

```

<field id="Channel PartnerStatus">
  <reader>
    <lotus_std>
      <lotus_field id="Channel PartnerStatus" />
      <convertor>
        <string />
      </convertor>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="Channel PartnerStatus" />
      <convertor>
        <string />
      </convertor>
    </lotus_std>
  </writer>
</field>

```

- 3 Save and then close the Ln_siebel_basic_mapping.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 276](#).

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 open the Ln_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 Ln_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 Creating MVG Fields” on page 269](#).

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 [“Using the Windows Registry to Control Siebel CRM Desktop” on page 103](#).

2 Test your changes:

- a** Open the 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 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 CRM Desktop automatically displays records in accordance with the letters you enter.
- f** Verify the salesbook control. You click the Salesbook icon and then verify that 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 That Adds a Custom Object Type on page 280](#)
- [Code That Maps a Custom Object on page 281](#)
- [Code That Configures Synchronization for a Custom Object on page 282](#)
- [Code That Adds a New Association on page 282](#)

Code That Adds 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='Channel Partner' Label='Channel Partner' Label Plural='Channel Partners' EnableGetIdsBatching='true' ViewMode='Sales Rep' IntObjName='Channel Partner' SiebMsgXmlElementName='Channel Partner' SiebMsgXmlElementName='ListOfChannel Partner' >

<field Name='DS Updated' Label='DS Updated' DataType='DTTYPE_DATETIME' IsFilterable='no' IsHidden='yes' IOElementName='DSUpdated' />

<field Name='Id' Label='Id' IsPrimaryKey='yes' DataType='DTTYPE_ID' IsFilterable='no' IsHidden='yes' IOElementName='Id' />
```



```

<field Name=' Name' Label =' Name' DataType=' DTYPE_TEXT' IsPartOfUserKey=' yes'
IOElementName=' Name' />

<field Name=' Location' Label =' Location' DataType=' DTYPE_TEXT' IsPartOfUserKey=' yes'
IOElementName=' 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 Ln_connector_configuration.xml file.

Code That Maps a Custom Object

You can map a field of a Siebel CRM object to the IBM Notes 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 IBM Notes
- Map the mapLocation field in IBM Notes to the Location field in Siebel CRM

To map objects, you add the following code to the Ln_siebel_basic_mapping.xml file:

```

<type id="Channel Partner">
  <field id="Name">
    <reader>
      <lotus_std>
        <lotus_field id="Name" />
        <convertor>
          <string />
        </convertor>
      </lotus_std>
    </reader>
    <writer>
      <lotus_std>
        <lotus_field id="LastName" />
        <convertor>
          <string />
        </convertor>
      </lotus_std>
    </writer>
  </field>
  <field id="Location">
    <reader>
      <lotus_std>
        <lotus_field id="Location" />
        <convertor>
          <string />
        </convertor>
      </lotus_std>
    </reader>
  </field>

```

```

    <writer>
      <lotus_std>
        <lotus_field id="Location" />
        <convertor>
          <string />
        </convertor>
      </lotus_std>
    </writer>
  </field>
</type>

```

Note the following requirements:

- The value for the *type id* tag in the Ln_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 value for the form tag must equal the form alias. This example does not require a form, so the form tag is empty.

Code That Configures Synchronization for a Custom Object

To configure synchronization for a custom object, you add the following code to the Ln_connector_configuration.xml file:

```

<type id="Channel Partner">
  <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 That Adds a New Association

To add a new association, you add the following code to the CreateMetaScheme function in the SBL.BusinessLogic script library:

```

Call Helper.AddMVGLink("Opportunity", "Channel Partner", _
Opportunity.Channel_Partner.Association", _
CRMDLeftId", "CRMDRightId", _
CRMDLeftStatus", "CRMDRightStatus", _
LINK_TAG_MVG, _
"PrimaryPartnerId", "")

```

Table 19 describes the important attributes you can use with the AddMVGLink function.

Table 19. Attributes in the Code That 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. |
| associationType | The Id of the association type described in the siebel_meta_info.xml file and the Ln_siebel_basic_mapping.xml file. |
| leftIdField | The field of the association object that contains the ID of the left object. |
| rightIdField | The field of the association object that contains the ID of the right object. |
| leftStatusField | The field of the association that contains the status of the left object. |
| rightStatusField | The field of the association that contains the status of the right object. |
| tag | The tag name of the link. |
| leftObjPrimaryField | The field of the left object which contains the primary ID for the right object. The field can contain an empty value. |
| rightObjPrimaryField | The field of the right object which contains the primary ID for the left object. The field can contain an empty value. |

12 Customizing Authentication

This topic describes how to customize authentication for Siebel CRM Desktop. It includes the following topics:

- [Overview of Customizing Authentication on page 285](#)
- [Installing CRM Desktop SSO on page 290](#)
- [About CRM Desktop SSO Architecture on page 297](#)
- [CRM Desktop SSO Objects You Can Customize on page 308](#)

Overview of Customizing Authentication

This topic describes an overview of single sign on for Siebel CRM Desktop. It includes the following topics:

- [Authentication That Comes Predefined with Siebel CRM Desktop on page 286](#)
- [Types of Authentication That You Can Use With CRM Desktop SSO on page 287](#)
- [Single Sign On Services That CRM Desktop SSO Supports on page 290](#)

CRM Desktop SSO (CRM Desktop Single Sign On) is a single sign on feature that allows the user to log in to CRM Desktop one time and use CRM Desktop features without being prompted to log in again to gain access to features that use access control. It allows you to implement single sign on for the CRM Desktop client. It uses the Web transport protocol and the HTTP or HTTPS protocol. It supports standard Web browser functionality, such as HTTP forms, cookies, and process redirects. It provides the following capabilities:

- Customizable architecture. You can install CRM Desktop SSO as an add-on. Beginning with Siebel CRM Desktop version 3.1, the CRM Desktop installer automatically includes CRM Desktop SSO.
- Supports your existing Web single sign on feature. You can customize CRM Desktop SSO so that it works in your network topology and for mobile or remote access.
- Supports your custom user interface. You can use CRM Desktop SSO that is automated and transparent to the user. You can also use it with a custom authentication screen that you already use, perhaps that includes your company branding.
- Supports typical login name and password authentication or custom authentication that requires more than just a login name and password.
- Handles single sign on, session, and network errors in a way that is transparent to the user.

Unless noted otherwise, support for features that this chapter describes begins with Siebel CRM Desktop version 3.1.

Authentication That Comes Predefined with Siebel CRM Desktop

This topic describes authentication that comes predefined with Siebel CRM Desktop.

Direct Connection

Siebel CRM Desktop can connect to an unprotected EAI (Enterprise Application Integration) endpoint for a direct connection. It comes predefined to use direct connection. For more information, see [“About Authentication and Session Management” on page 31](#).

Figure 14 includes the dialog box that CRM Desktop displays for a direct connection. This dialog box allows the user to enter the user name and password.



Figure 14. Dialog Box That CRM Desktop Displays for a Direct Connection

Single Sign On

CRM Desktop SSO can protect EAI on the client. In this situation, CRM Desktop SSO displays a dialog box that is identical to the dialog box in [Figure 14 on page 286](#) except the user sets the Auth Type field to SSO and the Siebel:SSOUser parameter determines if CRM Desktop SSO enables the User Name field. For more information, see [“Registry Keys That Control SSO for Credentials” on page 334](#).

If you configure CRM Desktop SSO to use interactive authentication, then it displays another dialog box after the user clicks Login in the CRM Desktop-Login dialog box. [Figure 15](#) includes an example of this second dialog box. For more information, see [“Types of Authentication That You Can Use With CRM Desktop SSO” on page 287](#).

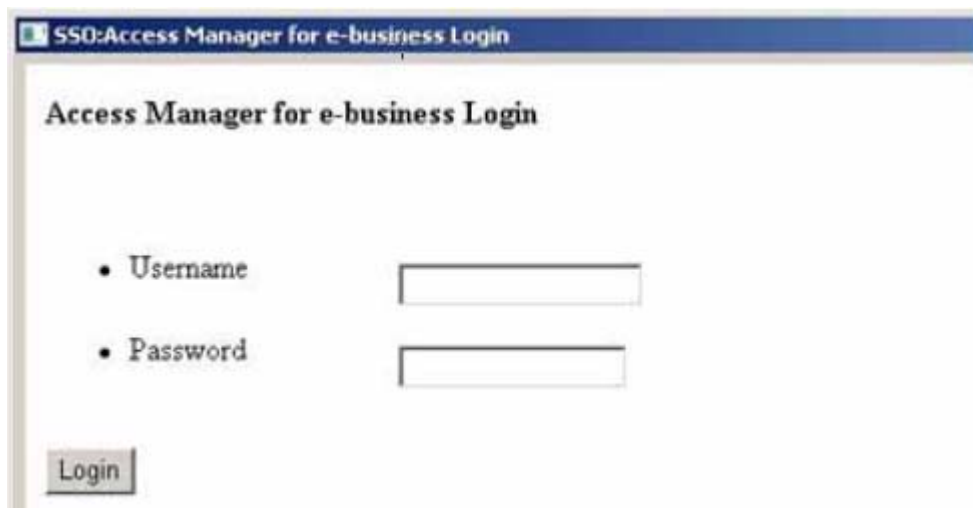


Figure 15. Example Dialog Box That CRM Desktop SSO Displays for Interactive Authentication

Types of Authentication That You Can Use With CRM Desktop SSO

This topic describes the types of authentication that you can use with CRM Desktop SSO.

Interactive Authentication

Interactive authentication is a type of authentication in CRM Desktop SSO that displays a second login dialog box that is native to the client. It displays this native dialog box in the following situations:

- After the user sets values in the CRM Desktop - Login dialog box and then clicks Login.
- The first time CRM Desktop logs in to the Siebel Server after Outlook restarts.
- A previously obtained SSO session expires.

Siebel CRM supports interactive authentication beginning with Siebel CRM Desktop version .

If you enable CRM Desktop SSO, then it uses interactive authentication by default. It is the only SSO authentication that comes predefined with CRM Desktop.

Interactive authentication includes the following functionality:

- **Detect session expiration.** If a CRM Desktop SSO session expires, then CRM Desktop SSO prompts the user to provide credentials and then reestablishes the session.

- **Multifactor authentication.** Supports more than only the login name and password from the the dialog box that is native to the browser, such as requiring code from a security token in addition to the password. The CRM Desktop SSO login dialog box can include more fields, images, a list of questions that the user must answer, ActiveX controls, and other items that your implementation requires for authentication. It can support an input field for an RSA (Rivest Shamir Adleman) token or other information that only the user can provide.
- **Supports Internet Explorer.** The CRM Desktop SSO log-in dialog box is an Internet Explorer ActiveX dialog box.

Interactive authentication provides the following benefits:

- Allows CRM Desktop SSO to support a more complex SSO login
- Allows you to use script to customize an SSO login on the client that supports a nonstandard configuration.
- Interactive SSO provides the following challenges:
- Cannot store credentials in the product configuration that CRM Desktop SSO can use later. This situation might require the user to reenter credentials during the first and subsequent SSO session logins.
- Supports only Internet Explorer version 7 or later.

How Siebel CRM SSO Starts and Stops Interactive Authentication

If the POST request to the EAI web service returns one of the following values, then CRM Desktop SSO starts interactive authentication:

- HTTP Redirect (302)
- HTML content

CRM Desktop SSO does one or more request and reply iterations during an interactive authentication. It monitors these iterations for the presence of one of the following stop conditions. If it encounters one of these conditions, then it stops the iteration:

- If the setting for the EndpointRegExp registry key is:
 - **Not defined.** The URL must match the URL parameter that resides on the Login dialog box. This setting is the default value.
 - **Defined.** The destination URL must match the EndpointRegExp registry setting.
- If the setting for the SuccessLoginRegExp registry key is:
 - **Not defined.** The HTML body must match the expression that the Siebel EAI server returns. This setting is the default value. For more information, see the description about SuccessLoginRegExp in [“Registry Keys That Control SSO for Siebel CRM Desktop” on page 333](#).
 - **Defined.** The HTML body must match the setting of the SuccessLoginRegExp registry key.

The user can also close the dialog box at any time to stop interactive authentication. CRM Desktop interprets this closure and cancels interactive authentication. For more information, see [“Registry Keys That Control SSO for Siebel CRM Desktop” on page 333](#).

Noninteractive Authentication

Noninteractive authentication is a type of authentication in CRM Desktop SSO that uses a separate dialog box as the login window for Siebel CRM Desktop. It is similar to the dialog box that [Figure 16 on page 346](#) displays. It includes the following functionality:

- **Save password.** The user can set the Save Password option on the CRM Desktop - Login dialog box to one of the following values:
 - **Include a check mark.** CRM Desktop SSO stores encrypted credentials in the Windows Registry. It does not prompt the user for credentials the next time the user starts CRM Desktop.
 - **Do not include a check mark.** CRM Desktop SSO prompts the user for credentials the next time the user starts CRM Desktop and after the first SSO session starts. This behavior typically occurs during the first synchronization that occurs after the user restarts CRM Desktop.
- **Detect session expiration.** If a CRM Desktop SSO session expires, then CRM Desktop SSO reestablishes the session without involving the user. CRM Desktop SSO requires user interaction only if the user credentials are not valid.
- **Use only the login name and password.** Noninteractive authentication does not support multifactor authentication, such as requiring code from a security token in addition to the password.
- **Detect invalid user password.** The SSO script can detect if the user enters an invalid password and then react accordingly.
- **Does not include Web pages.** The user interacts only with CRM Desktop Web pages that do not include CRM Desktop SSO.

Benefits and Challenges of Using Noninteractive Authentication

If CRM Desktop SSO uses noninteractive authentication, then SSO script interprets HTTP requests and replies that do not involve the user. It handles HTTP redirects, HTML form submits, automatic submits, and so on. It emulates the user interaction that typically occurs during a login that is native to the Web browser.

Noninteractive SSO provides the following benefits:

- CRM Desktop SSO can log in without user intervention.
- CRM Desktop SSO can reestablish a session automatically. User interaction is required only if a password lockout occurs.

Noninteractive SSO provides the following challenges:

- Requires slightly more customization than interactive authentication.

The implementation can be complex and difficult to scale. If you modify the login process that your company uses, then these modifications might be difficult to implement. For example, if your company must change from a simple username and password sign on to a complex sign on that requires more than these factors. In this situation, you must modify, test, and redeploy the SSO script.

Single Sign On Services That CRM Desktop SSO Supports

Predefined CRM Desktop SSO supports the following Web single sign on services:

- ClearTrust
- IBM Tivoli
- Computer Associates Siteminder
- Ping Federate

Customized CRM Desktop SSO can support other services, such as the following:

- Capgemini SSO
- CAS (Central Authentication Service)
- Cosign

CRM Desktop SSO supports the following operating systems:

- Windows XP SP3
- Windows Vista SP1 and above
- Windows 7

Installing CRM Desktop SSO

This topic describes how to install CRM Desktop SSO. It includes the following topics:

- Setting Windows Registry Keys to Enable CRM Desktop SSO on page 353
- Options for Installing CRM Desktop SSO on page 356
- Removing or Upgrading CRM Desktop SSO on page 359

Starting with Siebel CRM Desktop version 3.1, it is not necessary to install CRM Desktop SSO. It comes predefined starting with these version. For more information about using an installation package, see [“Overview of Installing the CRM Desktop Add-In” on page 81](#).

To install CRM Desktop SSO:

- 1 Verify the network and infrastructure:
 - a Modify the Windows Registry settings, as necessary.
For more information, see [“Using the Windows Command Line to Set Optional Parameters for Siebel CRM SSO” on page 294](#).
 - b Make sure SSO script supports the single sign on capabilities that your implementation requires.
CRM Desktop SSO cannot connect directly to a single sign on system. To operate properly, it requires SSO script that you customize for the target single sign on system. For a list of options, see [“Windows Registry Keys You Must Set to Enable CRM Desktop SSO” on page 330](#).

- c Determine the sequence you will use to install CRM Desktop SSO.

You can install, remove, or upgrade CRM Desktop SSO independent of CRM Desktop. You can install it before or after you install CRM Desktop.

- 2 Make sure you are a member of the Administrators group in Windows XP on the client computer. This membership provides the rights you require to run the executable file that CRM Desktop SSO uses in the installation package.

- 3 Set the Windows Registry keys.

For more information, see [“Setting Windows Registry Keys to Enable CRM Desktop SSO” on page 292](#).

- 4 (Optional) The installer calls the UAC (User Account Control) prompt in Windows Vista or Windows 7. To disable this prompt, you can set the following properties on the operating system on the client computer:

```
ALLUSERS=2
MSIINSTALLPERUSER=1
```

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

To install CRM Desktop SSO, you use an installation package that comes predefined as a single MSI file. This file includes the following data:

- Installation information for CRM Desktop SSO
- CRM Desktop SSO binary files

To copy this file, you can do one of the following

- Manually copy the InvisibleSSOModule.msi file to the client computer.
- Use third-party deployment software to deploy the InvisibleSSOModule.msi file to multiple computers. For more information, see [“Installing Siebel CRM Desktop in the Background” on page 97](#).

- 6 Log in to the client computer.

- 7 If Microsoft Outlook is open, then close it.

- 8 Locate the InvisibleSSOModule.msi installation package.

This location depends on where the user saves the InvisibleSSOModule.msi file. The following directory is a typical location:

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

- 9 Run the InvisibleSSOModule.msi installation package:

- a In the Welcome dialog box, click Next.

- b** In the Customer Information dialog box, specify the user name and organization and then chose to do this install for a single user or for any user who uses this client computer.

If you choose to do this installation for any user who uses this client computer, then you must be a member of the Administrators group. If you are not, then this installation will fail.

It is recommended but not required that you use the same installation configuration for CRM Desktop SSO that you use for CRM Desktop. For example, if you install CRM Desktop for each user, then it is recommended that you install CRM Desktop SSO for each user. If you install CRM Desktop for anyone who uses this computer, then it is recommended that you install CRM Desktop SSO for anyone who uses this computer. If you do not use this configuration, then CRM Desktop SSO might not be available for some users.

- c** In the Destination Folder dialog box, specify the folder where the installer must install CRM Desktop SSO.

You can specify a directory. For more information, see [“Setting the Installation Directory for CRM Desktop SSO” on page 295](#).

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

Because you can install CRM Desktop for multiple users, the user who is currently logged in can view application files that CRM Desktop stores in the default directory described in [“Setting the Installation Directory for CRM Desktop SSO” on page 295](#).

- e** In the InstallShield Wizard dialog box, click Finish.

The installer installs CRM Desktop SSO. The next time CRM Desktop starts it loads CRM Desktop SSO according to the Windows Registry settings. For more information, see [“For more information, see “Using the Windows Command Line to Set Optional Parameters for Siebel CRM SSO” on page 294.” on page 296](#).

After you complete the installation, CRM Desktop SSO is configured and CRM Desktop uses it when it communicates with the Siebel Server.

- 10** Notify the user that CRM Desktop SSO is installed.

Setting Windows Registry Keys to Enable CRM Desktop SSO

This topic describes Windows Registry settings that you must set if you install Siebel CRM Desktop for Siebel CRM Desktop version 3.1. You do not set registry keys with version 3.1, or later. This description applies only to installing CRM Desktop. It does not apply to using the InvisibleSSOModule.msi installer.

To set Windows Registry keys to enable CRM Desktop SSO

- Enter the required command-line parameters when you run msixec.exe.

For example:

```
msiexec.exe /I CRMDesktop.msi SSOENABLE=1 SSOSCRIPTINCLUDEPATH=%appdata%  
SSOURL=http://myurl
```

For more information, see [“Windows Registry Keys You Must Set to Enable CRM Desktop SSO” on page 330](#).

Options for Installing CRM Desktop SSO

This topic describes options for installing CRM Desktop SSO. It includes the following topics:

- [Installing CRM Desktop SSO If You Use Autoupdate on page 293](#)
- [Installing CRM Desktop SSO If You Do Not Use Autoupdate on page 294](#)
- [Using the Windows Command Line to Set Optional Parameters for Siebel CRM SSO on page 294](#)
- [Abbreviating the Installation Procedure on page 295](#)
- [Setting the Installation Directory for CRM Desktop SSO on page 295](#)

Siebel CRM Desktop version 3.1 and later comes predefined with these options enabled. It is only necessary to enable these options if you are using version 3.1 or earlier.

Installing CRM Desktop SSO If You Use Autoupdate

Autoupdate is a CRM Desktop SSO feature that automatically updates the SSO script if any change occurs to this script. CRM Desktop SSO scripts must reside in a single ZIP file. The CRM Desktop SSO installer uses the SSOURL parameter to determine the file path where these scripts reside. CRM Desktop SSO uses this location to download these scripts during installation and later during updates. This configuration allows you to deploy CRM Desktop SSO across your enterprise. It makes sure that the scripts on each client are consistent and valid. If autoupdate is enabled, then CRM Desktop SSO deploys the SSO script as a single ZIP file.

To install CRM Desktop SSO if you use autoupdate

- 1 Enter the following parameter on the msixexec command line anywhere after CRMDesktop.msi :

```
SSOENABLE=1 SSOURL=URL
```

For example:

```
msiexec.exe /I CRMDesktop.msi SSOENABLE=1 SSOURL=http://myurl
```

You can also set the following optional parameters:

```
SSOSCRIPTINCLUDETEMPLATE = template
SSOCHECKINTERVAL = new interval
SSOSCRIPTFILENAME = file name
```

For more information, see [“Using the Windows Command Line to Set Optional Parameters for Siebel CRM SSO” on page 294](#).

Installing CRM Desktop SSO If You Do Not Use Autoupdate

If you do not use autoupdate, then you configure CRM Desktop SSO to read scripts from a fixed location that you specify on the local drive. If autoupdate is not enabled, then CRM Desktop SSO deploys the SSO script as separate JavaScript files.

To install CRM Desktop SSO if you do not use autoupdate

- 1 Enter the following command on the msixec command line anywhere after CRMDesktop.msi :

```
SSOENABLE=1 SSOUPDATEDISABLE=1 SSOSCRIPTINCLUDEPATH=path_to_scripts
```

For example:

```
msiexec.exe /I CRMDesktop.msi SSOENABLE=1 SSOUPDATEDISABLE=1  
SSOSCRIPTINCLUDEPATH=%appdata%
```

You can also set the following parameters optional:

```
SSOCHECKINTERVAL = new interval  
SSOSCRIPTFILENAME = file name
```

For more information, see [“Using the Windows Command Line to Set Optional Parameters for Siebel CRM SSO” on page 294](#).

- 2 If you use MST or a prepackaged MSI, the you must make sure that the path that you specify for the SSOSCRIPTINCLUDEPATH parameter matches the path where you install the InvisibleSSOModule.msi file.

Using the Windows Command Line to Set Optional Parameters for Siebel CRM SSO

This topic describes how to use the Windows command line to set optional parameters for Siebel CRM SSO. CRM Desktop SSO supports all parameters that you can set in the Windows Installer msixec command line. This option is not available starting with Siebel CRM Desktop version . CRM Desktop SSO comes predefined starting with these versions. For more information, see [“Using the Windows Command Line to Set Optional Parameters” on page 98](#).

To use the Windows command line to set optional parameters

- 1 On the client computer, open the Windows command line interface.
- 2 Navigate to the directory that contains the InvisibleSSOModule.msi file.

For example:

```
c: \Documents And Settings\username\Desktop
```

- 3 Enter the Windows Installer command using the following format:

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

where:

- *optional_parameter* is a parameter that the installer can run. For example:

```
msiexec.exe /I InvisibleSSOModule.msi INSTALLDIR=c:\My_Custom_Directory
```

Note the following conditions:

- You must specify each optional parameter in the same command line after the name of the InvisibleSSOModule.msi file.
 - To separate each optional parameter, you must enter a space.
 - You can arrange optional parameters in any order.
- 4 Press Enter.
 - 5 The CRM Desktop SSO setup wizard displays the welcome dialog box.

Abbreviating the Installation Procedure

To automatically run the windows that normally require user action, you can use the optional QR parameter. If you use it, then the InvisibleSSOModule.msi installation package does not display dialog boxes that require user action. This option is not available starting with Siebel CRM Desktop version . CRM Desktop SSO comes predefined starting with these versions.

To abbreviate the installation procedure

- Append the QR parameter to the msiexec command.

For example:

```
msiexec.exe /I InvisibleSSOModule.msi INSTALLDIR=c:\My_Custom_Directory /QR
```

For more information, see [“Using the Windows Command Line to Set Optional Parameters for Siebel CRM SSO” on page 294](#).

Setting the Installation Directory for CRM Desktop SSO

The InvisibleSSOModule.msi installation package saves binary files during installation in one of the following directories, by default:

- Installation directory for each user on Windows 7 or Vista. For Windows XP, it installs the binary files into the following folder:

```
C:\Users\user\AppData\Roaming\
```

- Installation directory for each computer:

```
C:\Program Files\InvisibleCRM\SSO\
```

To change this directory, the user can choose a different location in the InstallShield wizard or you can use the INSTALLDIR parameter.

If you install CRM Desktop SSO for any user who uses this computer, then you must make sure that all users can read this directory. Predefined CRM Desktop SSO suggests a different directory depending on if you install for each user or for anyone who uses this computer.

This option is not available starting with Siebel CRM Desktop version 3.1. CRM Desktop SSO comes predefined starting with these versions.

To set the installation directory for CRM Desktop SSO

- Enter the following parameter on the msexec command line anywhere after the InvisibleSSOModule.msi name parameter:

```
INSTALLDIR=directory_path
```

For example:

```
msiexec.exe /I InvisibleSSOModule.msi INSTALLDIR=c:\My_Custom_Directory /QR
```

For more information, see [“Using the Windows Command Line to Set Optional Parameters for Siebel CRM SSO” on page 294](#).

Removing or Upgrading CRM Desktop SSO

This topic describes how to remove or upgrade CRM Desktop SSO for versions that occur earlier than Siebel CRM Desktop version 3.1. For information about removing multiple users, see [“Removing the CRM Desktop Add-In for Multiple Users” on page 114](#).

Removing CRM Desktop SSO for a Single User

This topic describes how to remove CRM Desktop SSO for a single user.

To remove CRM Desktop SSO for a single user

- 1 Log on to the computer where CRM Desktop SSO is installed.
- 2 Synchronize and back up all personal data:
 - a In Microsoft Outlook, perform synchronization.
 - b Backup personal data.

It is recommended that the user use the export feature in Microsoft Outlook to export personal data to a file.
- 3 Remove CRM Desktop SSO:
 - a In Microsoft Windows, click the Start menu, choose Settings, and then click Control Panel.
 - b In the Control Panel, right-click Add or Remove Programs and then choose Open.
 - c In the Add or Remove Programs dialog box, in the Currently Installed Programs window, click Invisible SSO Module and then click Remove.

CRM Desktop removes the registry settings and the files that CRM Desktop SSO uses.

Upgrading CRM Desktop SSO

This topic describes how to upgrade CRM Desktop SSO.

To upgrade CRM Desktop SSO

1 Remove CRM Desktop SSO.

For more information, see [“Removing CRM Desktop SSO for a Single User” on page 296.](#)

2 Install CRM Desktop SSO.

For more information, see [“Installing CRM Desktop SSO” on page 290.](#)

About CRM Desktop SSO Architecture

This topic describes the architecture that CRM Desktop SSO uses. It includes the following topics:

- [Architecture That CRM Desktop SSO Uses on page 298](#)
- [Flow That CRM Desktop SSO Uses During Authentication on page 299](#)
- [Flow That the CRM Desktop SSO DLL Uses on page 301](#)
- [Architecture That an SSO Session Uses on page 303](#)
- [How CRM Desktop SSO Handles Errors on page 306](#)
- [Modifying SSO JavaScript on page 307](#)

Architecture That CRM Desktop SSO Uses

Figure 16 illustrates the architecture that CRM Desktop SSO uses.

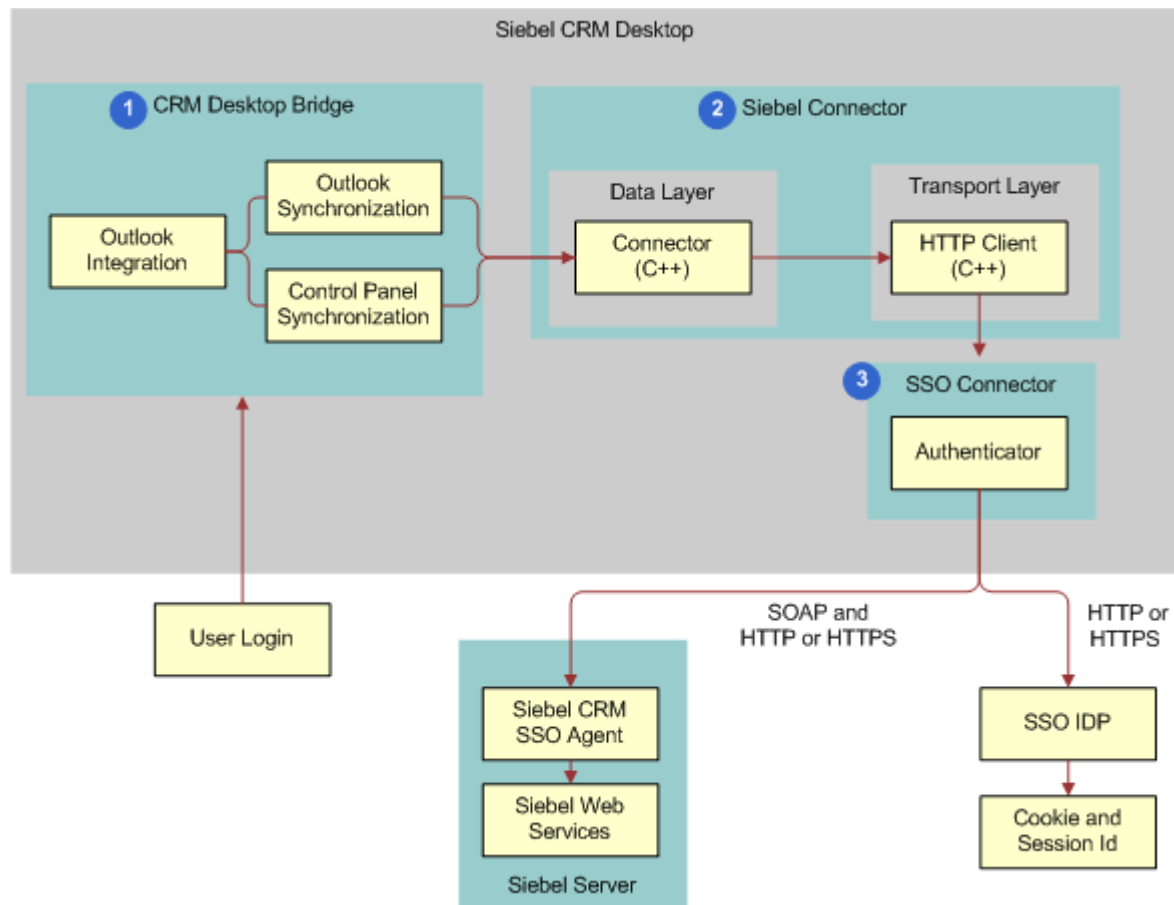


Figure 16. Architecture That CRM Desktop SSO Uses

Explanation of Callouts

The architecture that CRM Desktop SSO uses includes the following items:

- 1 CRM Desktop Bridge.** Uses a customization package that allows CRM Desktop to synchronize data. This synchronization uses the credentials that the user provides.
- 2 Siebel Connector.** Communicates information between CRM Desktop and Siebel CRM. It communicates through the SSO Connector to synchronize data, get the customization package, and open objects in Outlook
- 3 SSO Connector.** A set of JavaScript files that CRM Desktop SSO deploys to the client during installation. CRM Desktop uses the SSO Connector as a proxy to emulate a direct communication channel to the Siebel Connector. It uses SSO script. It includes the following items:

- **SSO session.** Includes state information and code that handles the request and reply that allows CRM Desktop SSO to establish, monitor, and exchange data between the connector and the Siebel Server. The Outlook Bridge uses the connector to start this session. To start multiple SSO sessions, this bridge can create a separate connector instance for each session.
 - **SSO session data.** Includes a global JavaScript context and other state information that the SSO script uses to track an SSO session.
 - **SSO session handler.** Each handler handles communication for a single SSO session.
- 4 CRM Desktop **SSO Proxy**. Proxy for the CRM Desktop SSO Connector that handles all requests from this connector and provides replies back to this connector.

Flow That CRM Desktop SSO Uses During Authentication

Figure 17 illustrates the flow that CRM Desktop SSO uses during authentication.

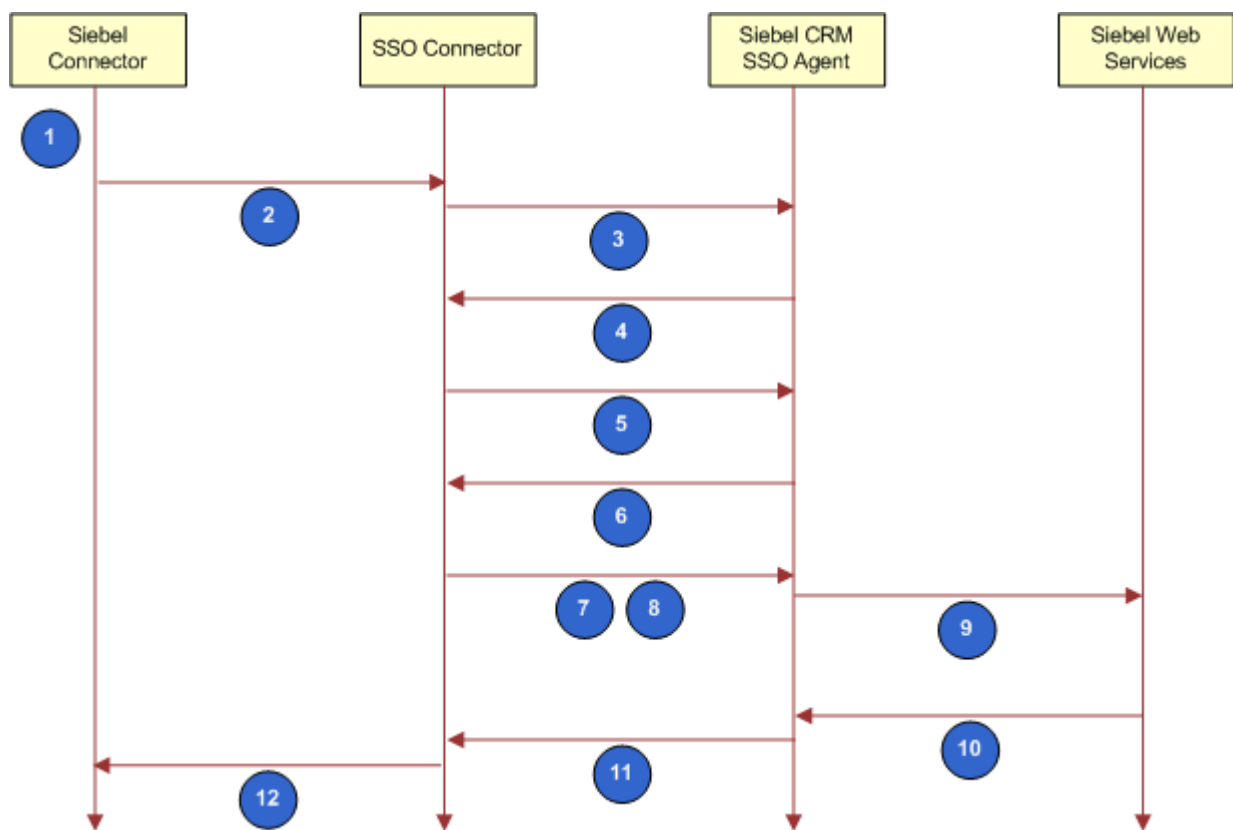


Figure 17. Flow That CRM Desktop SSO Uses During Authentication

Explanation of Callouts

The architecture for CRM Desktop SSO does the following during authentication:

- 1 User opens Outlook and then enters user name and password or uses credentials that CRM Desktop saved during a prior session.
- 2 Siebel Connector sends SOAP request with saved credentials to the SSO Connector. CRM Desktop SSO is a plug-in to CRM Desktop and acts as a local HTTP proxy. If the agent SessionID cookie that Siebel CRM SSO uses is set, then flow continues to [Step 9](#).
- 3 SSO Connector attempts to send a request to the SSO Agent.
- 4 If the Agent SessionID cookie that Siebel CRM SSO uses is not set, or if this cookie is expired, then the SSO Agent sends a request for authentication in an HTTP redirect form or in an HTML form. This HTML form allows the user to reenter authentication information.
- 5 SSO Connector detects a request for authentication and then starts interactive or noninteractive authentication.
- 6 SSO connector sends HTTP request to the SSO Agent.
- 7 The SSO Agent sends a reply to the client and does something depending on the following authentication that CRM Desktop uses:
 - **Interactive authentication.** The user must enter authentication information and then start the next step, for example, by clicking Login, and so forth.
 - **Noninteractive authentication.** The SSO Connector interprets the HTML reply.

[Step 6](#) and [Step 7](#) might repeat multiple times until authentication successfully finishes. CRM Desktop SSO considers this authentication successful if the HTTP can redirect to the original Siebel EAI address. When it meets this criteria is met, The SSO connector can use the session cookies when authentication successfully finishes.
- 8 The SSO Connector sends the original SOAP request and the session cookies to the SSO Agent.
- 9 The request now includes valid session information so the SSO Agent sends the original SOAP request to Siebel Web Services.
- 10 Siebel Web Services sends a reply to the SSO Agent.
- 11 The SSO Agent sends this reply to the SSO Connector. If the Siebel Server does not reply with HTTP 200 or HTTP 500, or if the reply does not include XML content, then the session is not valid and CRM Desktop SSO goes to [Step 5](#). The presence of XML content indicates that the user has logged in into the native Web SSO that the browser uses.
- 12 The SSO Connector sends a reply to the Siebel Connector for processing. The SSO Connector can store an Agent SessionID cookie while Outlook runs. It can reuse this cookie in subsequent connection attempts. If this cookie expires, then Siebel CRM SSO requests the user to log in again.

Flow That the CRM Desktop SSO DLL Uses

Figure 18 illustrates the flow that the CRM Desktop SSO DLL uses.

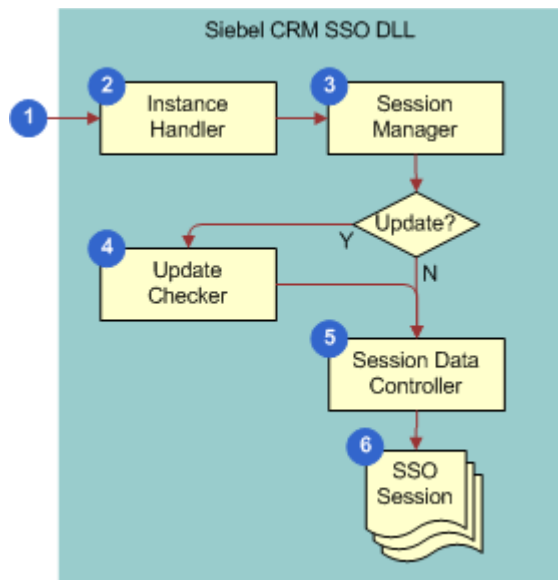


Figure 18. Flow That the CRM Desktop SSO DLL Uses

Explanation of Callouts

The CRM Desktop SSO DLL does the following:

- 1 CRM Desktop SSO loads the DLL.
- 2 The Instance Handler loads and initializes the DLL.
- 3 The Session Manager starts and maintains SSO sessions.
- 4 If you enable autoupdate, then the Update Checker automatically updates the script for each new session instance. For more information, see ["Installing CRM Desktop SSO If You Use Autoupdate" on page 293](#).
- 5 The Session Data Controller stores the names and values of parameters that the SSO sessions share. It allows data exchange between SSO sessions and provides a single location to store data that these sessions can share. For more information, see ["About Authentication Sessions and Data Exchange Sessions" on page 301](#).
- 6 The SSO sessions are a collection of SSO sessions that are currently active. For more information, see ["Architecture That an SSO Session Uses" on page 303](#).

About Authentication Sessions and Data Exchange Sessions

CRM Desktop SSO can create the following types of sessions:

- **Authentication session.** Starts if the user must change the login name and password or if CRM Desktop SSO requests the user to reenter the password to confirm these credentials. Note the following:
 - The SSO script does not use any cached information from a previous SSO session during an authentication session.
 - In some situations the SSO script cannot prevent Internet Explorer from allowing the user to access CRM Desktop without entering credentials. This situation typically occurs if CRM Desktop SSO uses a persistent cookie to identify the Web SSO user session. To allow the user to modify credentials when CRM Desktop SSO uses a persistent cookie, the user must use Internet Explorer to log out from CRM Desktop. This log out removes the persistent cookie. An authentication session prompts the user for a user name and password the next time the user attempts to connect to the Siebel Server from CRM Desktop.
- **Data exchange session.** Starts during a normal operation, such as synchronization, opening the Control Panel, and so on. CRM Desktop SSO can create multiple data exchange sessions. To avoid displaying unnecessary login prompts, the SSO Connector caches any session cookies that exist in the shared session cache or that reside in the cookie cache that Internet Explorer uses.

Architecture That an SSO Session Uses

Figure 19 illustrates the architecture that an SSO session uses.

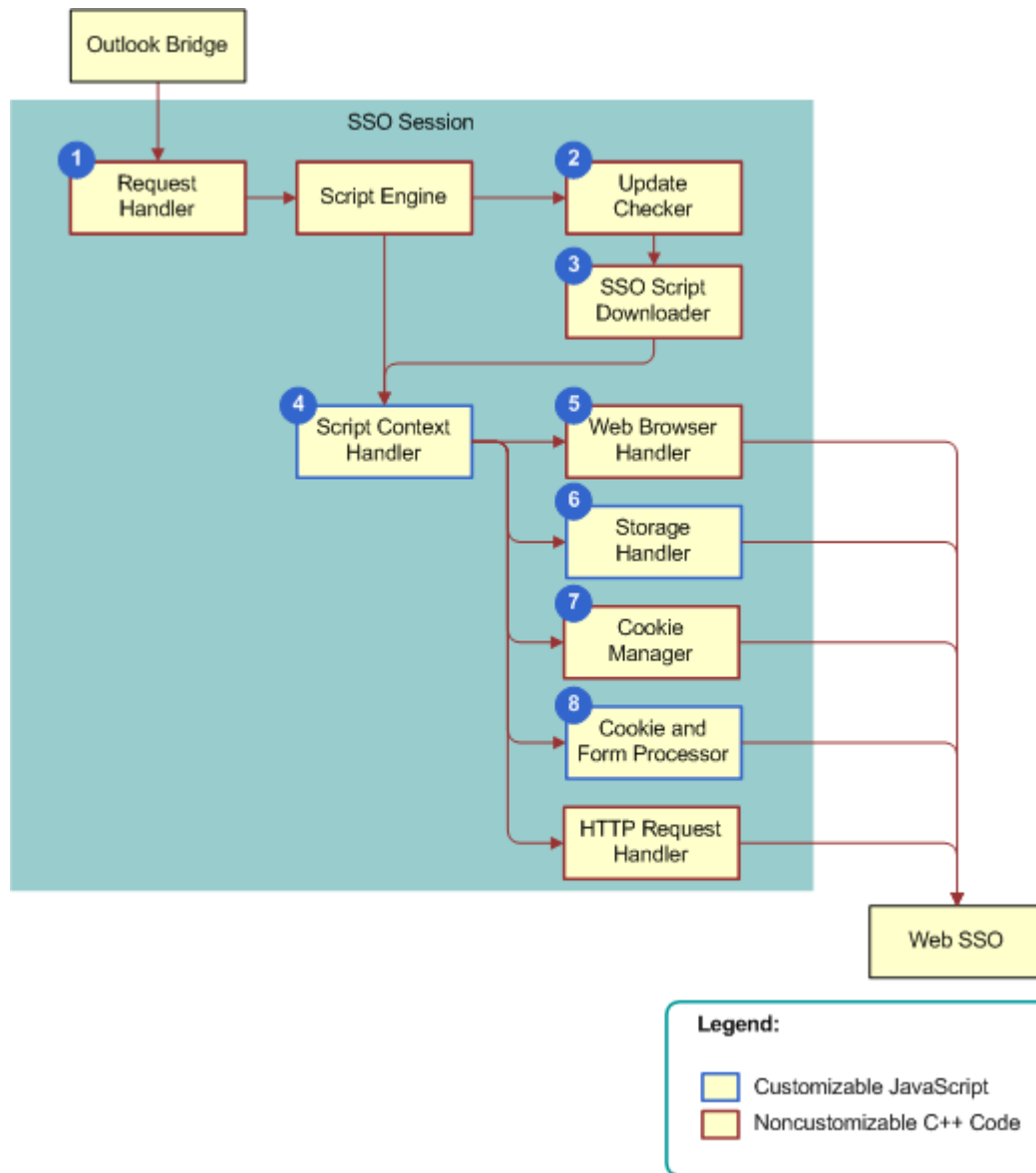


Figure 19. Architecture That an SSO Session Uses

Explanation of Callouts

The architecture that an SSO session uses includes the following items:

- 1 **Request handler.** Accepts each request from the Outlook Bridge, wraps the request in an object that an SSO script can access, routes the request through SSO script, and then sends the reply from the SSO to the client. It initializes the script, registers the request handler to handle connector requests and initializes the request handler. For example, it reads configuration settings, initializes global variables, and so forth. It runs the function that the request handler code registers for callback. For each incoming connector request, the SSO script establishes or reuses an SSO session with the Siebel Server and sends a reply to this server. For more information, see [“Request Handler Function” on page 312](#).
- 2 **Update checker.** Calls the SSO script that runs autoupdate. It does this work the first time this session uses this script. For more information, see [“Installing CRM Desktop SSO If You Use Autoupdate” on page 293](#).
- 3 **SSO script downloader.** Downloads and prepares the SSO script.
- 4 **Script context handler.** Uses one instance of a Microsoft ActiveScript engine that runs SSO script and sends a reply to a request handler notification.
- 5 **Web Browser handler.** Displays an interactive login prompt to user, interprets the login information that this user enters, and notifies the SSO script.
- 6 **Storage handler.** Handles persistent and session storage.
- 7 **Cookie manager.** Gets and sets Internet Explorer cookies.
- 8 **Cookie and form processor.** Processes cookies, forms, and redirects. For more information, see [“Cookie Handling” on page 305](#).

To customize the following items, you can reuse SSO objects or you can use your own set of common code. For more information, see [“CRM Desktop SSO Objects You Can Customize” on page 308](#).

- Script context handler
- Storage handler
- Cookie and form processor

You cannot use JavaScript to customize items in [Figure 19](#) that use C++ code, but you can change registry settings that affect these items.

SSO Script Lifecycle

If CRM Desktop SSO is enabled, and if the first connector starts, then CRM Desktop SSO loads the SSO module into Siebel CRM Desktop and it remains loaded until CRM Desktop closes.

The SSO script context includes all JavaScript global variables and state information. It is part of the SSO session data. The SSO Session Manager creates it and it exists until the SSO session ends.

Requests to start or end a session depend on the connector lifetime. CRM Desktop SSO starts a new session when it starts each new connector instance. If it ends a connector instance, then it also ends the SSO script context.

SSO Script Autoupdate

If SSO Script Autoupdate is enabled, then this Autoupdate determines if updated SSO script is available. If updated script is available, then CRM Desktop SSO loads this updated script instead of loading the old script. This configuration might result in the memory containing multiple versions of SSO script and SSO script context. When the connector sessions finish, CRM Desktop SSO unloads any old SSO script that exists and replaces it with the updated script.

Sharing Information Between Contexts

CRM Desktop SSO isolates script contexts and makes them independent from each other. To avoid unnecessary reauthentication, a script can handle different SSO sessions that share information. To do this, CRM Desktop SSO uses the `settings_cache` global object to read the configuration from one SSO session and reuse it or modify it in another SSO session.

SSO Script Operation

This topic describes SSO script operation.

Initialization

CRM Desktop SSO initializes SSO script when it creates a new SSO session. The initialization code must register a handler for the `request_handler` so that it handles connector requests and does the initialization that makes sure request handling is operational. For example, to set the read configuration settings, initialize global variables, and so forth.

Request Handling

To handle an SSO script request, CRM Desktop SSO runs a function for the `request_handler` callback. SSO script establishes or reuses an SSO session with the Siebel Server and returns a reply from this server for each incoming connector request.

Credentials Handling

CRM Desktop SSO handles credentials in one of the following ways:

- **Noninteractive authentication.** Sets user credentials in the CRM Desktop login dialog box and then communicates them to the SSO script through the `get_sso_username` function and the `get_sso_password` function of the `sso_client` global object.
- **Interactive authentication.** Does not send user credentials to SSO script. This SSO script must make sure that CRM Desktop allows the user to authenticate and that the authentication session runs correctly. It must use the `ia_state` object to capture cookie information and then use this information in the request and reply with the Siebel Server.

Cookie Handling

CRM Desktop SSO uses the WinHTTP protocol to support cookie handling. For more information, see the topic about Manual and Automatic Cookie Handling in the Cookie Handling in WinHTTP topic in the Dev Center - Desktop section of the Microsoft Developer Network web site.

The `execute_request` call returns cookies that the Siebel Server sets as part of the HTTP handling. WinHTTP interprets this call and adds it to the cookie cache that CRM Desktop SSO reuses during subsequent requests. The client can also specify cookies and then add them to a request. Interactive authentication requires special handling of cookies. Noninteractive authentication uses WinHTTP while interactive authentication uses Internet Explorer. CRM Desktop SSO sends all required cookies from the script session to the Internet Explorer session before it starts an interactive authentication. CRM Desktop sends these cookies back to the WinHTTP noninteractive session after interactive authentication finishes.

How CRM Desktop SSO Handles Errors

SSO script avoids creating JavaScript exceptions. If the user enables a JIT Debugger on the client computer, such as Visual Studio, then this debugger might display a separate prompt and debug message for each exception. These prompts can significantly affect the user experience. Siebel CRM SSO uses the following functions to handle exceptions:

- `cpp_exception_occurred`
- `drop_exception`
- `raise_not_logged_in_exception`
- `raise_not_valid_exception`
- `raise_cancel_exception`
- `execute_request`

The `execute_request` function is the only function that can fail with exception. It depends on the network state and the availability of Siebel Servers. If `execute_request` fails, then Siebel CRM SSO returns null instead of a response object to handle logic that does not include exceptions. The SSO Connector must examine the return code, determine if an error occurred, and then take corrective action.

If an exception occurs, then the SSO script does one of the following:

- **Pass error to Siebel Connector.** Recreates the exception when flow returns to the Siebel Connector so that this connector can handle the exception. The SSO Connector script must finish running without calling any other `execute_request` function or calling a function that modifies the exception, such as a `raise_value_exception` or `drop_exception`. It does this to avoid overwriting the original exception message that the script created as a result of the error.
- **Clear the exception.** Use the `drop_exception` function to clear the exception.
- **Run another request.** Use the other `execute_request` function to clear the previous exception.
- **Create an exception:**
 - Use the `raise_not_valid_exception` function. In this situation, the user provided the credentials in the client but they are not valid.
 - Use the `raise_not_logged_in_exception` function. In this situation, the credentials are not complete or are missing.

For more information about these methods, see [“SSO Client Object” on page 308](#).

How the SSO Script Sends an Error to the Connector

The following type of error determines how the SSO script sends an error to the connector:

- **Network error.** If the `execute_request` function returns a value of `Null`, then the SSO script sends the error back to the connector and the connector processes the error.
- **Authentication or SSO script logic error.** The SSO script translates the error to a `not_valid` or a `not_logged_in` error so that the corresponding function in the `sso_client` object can handle the error. It uses the following functions:
 - Uses the `raise_not_logged_in_exception` function for a `not_logged_in` error
 - Uses the `raise_not_valid` function for a `not_valid` error
- **User cancelled interactive authentication.** The SSO script runs the `raise_cancel_exception` function to start a `cancel_exception` exception.

How the SSO Script Logs Errors and Messages

Siebel CRM SSO writes log messages to the CRM Desktop general log which simplifies debugging and gathering diagnostics. The SSO script logs information about the SSO API and application failures. For more information, see [“Logger Object” on page 314](#).

Modifying SSO JavaScript

You can use the API (application programming interface) that comes predefined with CRM Desktop SSO (SSO API) to create a custom script that does the following:

- Get HTML and XML pages
- Submit forms
- Follow redirects
- Support cookies

CRM Desktop SSO uses JavaScript to implement the logic it requires. This script can accommodate some layout or workflow changes but it might be necessary to modify it to meet your deployment requirements. CRM Desktop SSO comes predefined with the following JavaScript files:

- **core.js and utils.js.** A set of reusable functions and building blocks that handle authentication. The `lib.js` file includes detailed documentation in the source code that describes the API that you can use to customize CRM Desktop SSO.
- **sso.js.** An entry point that handles authentication.

You must distribute any update you make by uploading the new script you create to an autoupdate location or you can use external provisioning. For more information, see [“Installing CRM Desktop SSO If You Use Autoupdate” on page 293](#).

CRM Desktop SSO Objects You Can Customize

This topic describes the objects that you can use to customize CRM Desktop SSO. It includes the following topics:

- [SSO Client Object on page 308](#)
- [Logger Object on page 314](#)
- [Settings Cache Object on page 314](#)
- [Settings Object on page 315](#)
- [Request Object on page 315](#)
- [Response Object on page 315](#)
- [Content Object on page 316](#)
- [Header Object on page 317](#)
- [Credentials Object on page 318](#)
- [Interactive State Object on page 319](#)
- [Dialog Object on page 320](#)

SSO Client Object

This topic describes functions that you can use with the SSO client object. It includes the following topics:

- [Create Request Function on page 309](#)
- [Create Response Function on page 309](#)
- [Decode URL Function on page 310](#)
- [Encode URL Function on page 310](#)
- [Exception Occurred Function on page 311](#)
- [Get Platform Cookie Function on page 311](#)
- [Interactive Function on page 311](#)
- [Request Handler Function on page 312](#)
- [Set Platform Cookie Function on page 313](#)
- [Other Functions That the SSO Client Object Includes on page 313](#)

To communicate the credentials that the user enters in the CRM Desktop login dialog box to the SSO script, CRM Desktop SSO uses the `get_sso_username` and `get_sso_password` functions of the `sso_client` object.

The `sso_client` object is a global object that is available anywhere in the main script file. The following registry key identifies this main script file:

```
SSO\ScriptFileName
```

For more information, see ["Windows Registry Keys You Must Set to Enable CRM Desktop SSO" on page 330](#).

Create Request Function

The `create_request` function creates a request and returns a new request object. It uses the following format:

```
create_request(url, method, body, encoding)
```

where:

- *url* is a string that contains the URL that identifies the request endpoint.
- *method* is a string that identifies the HTTP request method. It can include one of the following values:
 - GET
 - POST
- *body* is a string that contains the body of the request.
- *encoding* is a string that identifies how to encode the request body. It can include one of the following values:
 - iso-8859-1
 - utf-8
 - utf-16

For example, the following code creates a simple GET request to a URL:

```
var req = sso_client.create_request("http://siebel\ei_enu", "GET", "", "utf-8");
```

For more information, see ["Request Object" on page 315](#).

Create Response Function

The `create_response` function creates and returns a new response object. It uses the following format:

```
create_response(http_code, http_status, body, encoding)
```

where:

- *http_code* is an integer that identifies an HTTP status code. For example, 200, 500, or 404. For more information, see the topic about status code definitions at the World Wide Web Consortium (W3C) web site.
- *http_status* includes a string that identifies an HTTP status message that corresponds to the value that *http_code* identifies. For example, OK for 200, Not found for 404, and so forth.

- *body* includes a string that is the response body.
- *encoding* includes a string that identifies how to encode the response body. It can include one of the following values:
 - iso-8859-1
 - utf-8
 - utf-16

You must use double quotes to enclose each value. For example:

```
var resp_body = "... response body ...";
var resp = sso_client.create_response(500, "Internal Server Error", resp_body,
"UTF-8");
  resp.get_headers().add_header("Content-Type", "text/xml; charset=utf-8");
  resp.get_headers().add_header("Content-Length", ""+resp_body.length);
```

Decode URL Function

The `decode_url` function decodes a URL. It does the following:

- 1 Splits a URL into components and then reconstructs this URL. It adds any missing components.
- 2 Replaces encoded sequences into their corresponding values. The following format identifies a sequence:

`%xx`

where:

- `%` represents percent encoding that allows you to encode information in a Uniform Resource Identifier (URI).

The `decode_url` function uses the following format:

`decode_url (url)`

where:

- *url* is a string that contains the URL that the `decode_url` function decodes.

Encode URL Function

The `encode_url` function encodes a URL. It replaces each nonstandard character with the encoded value. The following format identifies a sequence:

`%xx`

The `encode_url` function uses the following format:

`encode_url (url)`

where:

- *url* is a string that contains the URL that the `encode_url` function encodes.

Exception Occurred Function

The `cpp_exception_occurred` function is a Boolean function that returns True if the last call to an `execute_request` function resulted in C++ code creating an exception. Any subsequent call to the `execute_request` function resets a previously recorded exception, so you must use this function immediately after the call to the `execute_request` function.

Get Platform Cookie Function

The `get_platform_cookie` function gets the Internet Explorer cookie. It can only get Internet Explorer cookies that are persistent and that are not of type HTTPOnly. It uses the following format:

```
get_platform_cookie(url, name)
```

where:

- *url* is a string that contains the URL that CRM Desktop SSO uses to get the cookie.
- *name* is a string that contains the name of the cookie.

Interactive Function

The interactive function instructs the SSO script to use interactive authentication. It uses callbacks to monitor the state of the interactive session. CRM Desktop SSO runs the statement that occurs immediately after the statement that calls the interactive function only after the interactive authentication finishes. While interactive authentication runs, the SSO script gets notifications from the function that the callback parameter identifies. This callback delivers the information that this script requires to monitor the interactive authentication. It can send a request to stop the interactive authentication when the interactive session finishes.

The interactive function uses the following format:

```
interactive(ia_state, callback)
```

where:

- *ia_state* is a string that identifies the object that contains information about interactive authentication.
- *callback* is a string that identifies the function that CRM Desktop SSO calls on every change that occurs in the `ia_state.status`. This callback must return a Boolean value. If it returns True, then CRM Desktop SSO stops the interactive authentication. *ia_state* is an object that the `sso_client.create_ia_state` function creates and then uses to control how interactive authentication runs. The callback is a function that returns one of the following values:
 - **true**. Interactive authentication finished.
 - **false**. Interactive authentication has not yet finished.

The following example does a simple callback for interactive authentication. It does not do any processing. It always return false to indicate that CRM Desktop SSO must display the native browser dialog box in every subsequent web page where the user navigates:

```
function i_a_callback(i_a_state)
{
    switch (i_a_state.status) {
        case "before":
            // before navigate to url
            break;

        case "finished":
            // page download complete
            break;

        case "cancelled":
            // user closed dialog
            break;
    }
    return false;
}
```

This example includes the following code. This code is part of the request handler function:

```
var i_a_state = sso_client.create_i_a_state();
i_a_state.url = url;
i_a_state.dialog.width = 1024;
i_a_state.dialog.height = 768;
i_a_state.dialog.title = "SSO";
i_a_state.dialog.visible = false;
sso_client.interactive(i_a_state, i_a_callback);
```

Request Handler Function

CRM Desktop SSO routes each request from the Siebel Connector through the `request_handler` function. This function handles requests that occur from the Siebel Connector to the Siebel Server. This `request_handler` function expects a single argument that contains the initial request object from the SSO Connector. It sends a return value to the SSO Connector as if the Siebel Server sent this reply. The Siebel Connector is a proxy that resides between the Siebel Connector and Siebel Web Services. The SSO Agent protects these Web services. This proxy hides the SSO Agent from the Siebel Connector.

The following example includes a CRM Desktop SSO script that passes all requests from the Siebel Connector to the Siebel Server without doing any processing. This example is for illustration purposes only. An actual SSO Connector script includes the `process_request` function to do processing:

```
sso_client.request_handler = process_request;
function process_request(request)
{
    return sso_client.execute_request(request);
}
```

The following line from this code defines the entry point where CRM Desktop SSO registers the handler:

```
sso_client.request_handler = process_request
```


Set Platform Cookie Function

The `set_platform_cookie` function sets the Internet Explorer cookie. It can only get Internet Explorer cookies that are persistent and that are not of type HTTPOnly. It uses the following format:

```
get_platform_cookie(url, name, value)
```

where:

- *url* is a string that contains the URL that CRM Desktop SSO uses to set the cookie.
- *name* is a string that contains the name of the cookie.
- *value* is a string that contains the data that CRM Desktop SSO associates with the cookie.

For example:

```
sso_client.set_platform_cookie("http://example.com", "Cookie_name",  
"Cookie_value");
```

Other Functions That the SSO Client Object Includes

You can use the following functions in the SSO Client object. CRM Desktop supports each of these functions starting with Siebel CRM Desktop version except for the `execute_request` function. Support for the `execute_request` function starts with an earlier release:

- **create_ia_state.** Creates and returns a new `ia_state` object. CRM Desktop SSO uses this object for initial configuration with interactive authentication to send status information. `ia_state` is an object, while `create_ia_state` is a function that creates the `ia_state` object. For more information, see [“Interactive State Object” on page 319](#).
- **execute_request.** Runs the request and returns a reply. It accepts a request from the `request_handler` function or a request that the `create_request` function creates. It uses the following format:


```
execute_request(request_object)
```
- **get_sso_username.** Gets the user name that the user provides in noninteractive authentication. CRM Desktop SSO does not use this function for interactive authentication.
- **get_sso_password.** Gets the password that the user provides in noninteractive authentication. CRM Desktop SSO does not use this function for interactive authentication.
- **drop_exception.** Drops any exceptions that the connector reports. If the script finishes running after CRM Desktop SSO calls this method, then CRM Desktop SSO does not create an exception.
- **raise_not_logged_in_exception.** Drops any `not_logged_in` exceptions that the connector reports and then creates a `not_logged_in` exception with a description that states it cannot process the login because a problem occurred with the Siebel Server or credentials are not valid. CRM Desktop SSO does not stop the session flow. The SSO script must finish running correctly before the connector creates the exception.
- **raise_not_valid_exception.** Drops any `not_valid` exceptions that the connector reports and creates a `not_valid` exception with a description that states the user password is not valid or is missing. CRM Desktop SSO does not stop the session flow. The SSO script must finish running correctly before the connector creates the exception.

- **raise_cancel_exception.** Clears any reported exception. It creates the following exception for the SSO Connector:

`cancelled`

CRM Desktop SSO does not stop the session flow. The SSO script must finish running the `request_handler` after it creates an exception.

Logger Object

The logger object is a global object that is available anywhere in the main script file. It allows CRM Desktop SSO to log messages to a general log. It uses the following format:

`log(component, message, class)`

where:

- *component* is a string that identifies the source of the log message
- *message* is a string that identifies the message that CRM Desktop SSO logs.
- *class* is a string that identifies the error class.

For example:

```
Logger.log ("SSO", "Sample message", 0);
```

Settings Cache Object

The `settings_cache` object is a global object that is available anywhere in the main script file. It is a shared name-value cache. Multiple SSO script instances that run in different threads of a single process can reuse this cache to get session information.

It includes the following function that sets the value of a cache setting:

`set(name, value)`

where:

- *name* is a string that identifies the setting name
- *value* is a string that identifies the setting value

It includes the following function that gets the value of a cache setting:

`get(name)`

where:

- *name* is a string that identifies the setting name

For example:

- The following code stores a value that CRM Desktop SSO gets in another instance of the SSO script:

```
settings_cache.set("MyKey", "MyValue");
```

- The following code gets the stored value. If no value exists, then this code returns an empty string:

```
var v = settings_cache.get("MyKey");
```

Settings Object

The settings object is a global object that is available anywhere in the main script file. It gets the value of a key that CRM Desktop SSO maps from the product key in the Windows Registry. It uses the following format:

```
get(name)
```

where:

- *name* is a string that identifies the name of a registry key

Request Object

The request object represents the initial HTTP request that the connector sends to the SSO script. You can also use it to allow the SSO script to create an instance of a request object that establishes an SSO session for the connector, as necessary. It includes the following functions:

- **get_headers.** Gets the object that represents the HTTP request header. For more information, see [“Header Object” on page 317](#).
- **get_server.** Gets the host part of the request URL. For example, `http://example.com/`.
- **get_object.** Gets the path part of the request URL. For example, `path/page.asp`.
- **get_method.** Gets the HTTP method. For example, GET, POST, and so forth.
- **get_contents.** Gets the content object that contains the request body and that CRM Desktop SSO can use to get the request in plain text. For more information, see [“Content Object” on page 316](#).
- **get_credentials.** Gets the credentials object that contains the security credentials for the HTTP request. These credentials include the login name, password, and other authentication information. For more information, see [“Credentials Object” on page 318](#).

Response Object

The response object contains the HTTP response that CRM Desktop SSO gets from the Siebel Server. It includes the properties that CRM Desktop SSO requires to retrieve the response status, headers, and the body. It includes the following functions:

- **get_status_code.** Gets the HTTP status code for the current response.

- **get_headers.** Gets the header object that contains the HTTP headers for the response. For more information, see [“Header Object” on page 317](#).
- **get_contents.** Gets the content object that contains the response body and that CRM Desktop SSO can use to get the response in plain text. For more information, see [“Content Object” on page 316](#).

Content Object

The content object gets the string content of a request or a response. It includes the following function that gets the text content of a request or response:

```
get_text(encoding)
```

where:

- *encoding* is a string that identifies how the body of the request or response is encoded. It can include one of the following strings:

- iso-8859-1
- utf-8
- utf-16

For example:

```
get_text(i so-8859-1)
```

It includes the following function that sets the text content of a request or response:

```
set_text(text, encoding)
```

where:

- *text* is the text of the body of the request or response.
- *encoding* is a string that identifies how CRM Desktop SSO must encode the body of the request or response. It can include one of the following strings:

- iso-8859-1
- utf-8
- utf-16

For example:

```
set_text("My text string", "i so-8859-1");
```

You must enclose each string in JavaScript quotes, so the following code returns the body of the request converted from utf-8:

```
var body = req.get_contents().get_text("utf-8");
```

CRM Desktop SSO does not allow you to modify the text of the initial request that comes as a parameter of the `request_handler` function. For more information, see [“Request Handler Function” on page 312](#).

For more information, see [“Request Object” on page 315](#) and [“Response Object” on page 315](#).

Header Object

The header object includes the HTTP headers of a request or response. It is read-only for a response and writable for a request. It includes the following functions:

- **`get_header_count`**. Gets the number of headers that the request or response contains.
- **`get_string_value`**. Gets all headers as a single string.

Get Header Name Function

The `get_header_name` function gets the name of a header. It uses the following format:

```
get_header_name(index)
```

where:

- *index* is a numeric value that identifies the header index. This index starts with 0 as the first value. To access the first header, you use an index value of 0. To access the second header, you use an index value of 1, and so forth.

For example, the following code gets the name of the first header:

```
var header = req.get_headers().get_header_name(0);
```

Get Header Value Function

The `get_header_value` function gets the value of a header. It uses the following format:

```
get_header_value(index)
```

where:

- *index* is a numeric value that identifies the header index. This index starts with 0 as the first value. To access the first header, you use an index value of 0. To access the second header, you use an index value of 1, and so forth.

For example, the following code gets the value of the first header:

```
var header_val = req.get_headers().get_header_value(0);
```

Add Header Function

The `add_header` function adds a new header. You can use it only for a request. You cannot use it for a reply. It uses the following format:

```
add_header(name, value)
```

where:

- *name* is a string that contains the header name.
- *value* is a string that contains the header value.

For example, the following code adds the content type for the request:

```
req.get_headers().add_header("Content-Type", "text/html ");
```

For more information, see ["Request Object" on page 315](#) and ["Response Object" on page 315](#).

Credentials Object

The credentials object is a subobject of a request object that specifies the HTTP authorization parameters that CRM Desktop SSO uses. It includes the following functions:

- **get_username.** Gets the user name associated with a request.
- **get_password.** Gets the password associated with a request.
- **get_auth_schemes.** Gets the list of authorization schemes that this request supports, in order of preference. For a list of schemes, see ["Set Authorization Schemes Function" on page 318](#).

Set User Name Function

The set_username function sets the user name for a request. It uses the following format:

```
set_username(username)
```

where:

- *username* is a string that contains the user name.

Set Password Function

The set_password function sets the password for a request. It uses the following format:

```
set_password(password)
```

where:

- *password* is a string that contains the password.

Set Authorization Schemes Function

The set_auth_schemes function sets the authorization schemes that CRM Desktop SSO uses with a request. It uses the following format:

```
set_auth_schemes(auth_schemes)
```

where:

- *auth_schemes* is a string that contains the authorization schemes. It can contain any combination of the following values:
 - **B, b.** Specifies to use basic authorization.
 - **N, n.** Specifies to use NTLM (NT LAN Manager) authorization.
 - **P, p.** Specifies to use passport authorization.
 - **D, d.** Specifies to use digest authorization.

You must use the following format:

- The order that you use is important. For example, if you specify NB, then this request only supports NTLM or basic authentication and it uses NTLM first, if possible.
- JavaScript requires that you use double quotes to enclose each string.
- Values are not case-sensitive.

For example, the following code configures the HTTP client to use NTLM basic authentication if the direct request fails, where NTLM authentication takes priority over basic authentication:

```
req.get_credentials().set_auth_schemes("nb");
```

For information about basic authorization and passport authorization, see the topic about HTTP authentication schemes at the Microsoft Developer Network web site.

Interactive State Object

The interactive state object contains information about the state of the interactive authentication. The `create_ia_state` function of the `sso_client` object creates the interactive state object so that the interactive function of the `sso_client` object can use it when it calls the SSO script. CRM Desktop SSO uses the same interactive state object to do a callback from the code that handles the interactive authentication to the SSO script. For an example that uses a callback, see [“Interactive Function” on page 311](#).

The `ia_state` object includes the following properties:

- **url.** A string that identifies the URL of the page that Internet Explorer displays when Siebel CRM SSO runs the following function:

```
interactive()
```

CRM Desktop SSO provides the current URL that the Web browser object processes. It does this during the callback. CRM Desktop SSO uses `ia_state` later as a callback to JavaScript, and the URL contains the current URL of the page that the browser control processes. For an example of this usage, see [“Example Code That Customizes CRM Desktop SSO” on page 320](#).

- **cookies.** A string that identifies the cookies that are associated with the HTML page that CRM Desktop SSO loads into the control Internet Explorer.
- **headers.** A string that identifies any other HTTP headers that the Browser sends.

- **post_data.** A string that identifies the data that CRM Desktop SSO sends through an HTTP POST function to the Siebel Server. This property exists on form submission. CRM Desktop SSO uses HTML forms to get user information on the Web page and to send this information to the Siebel Server for processing. For example, to get the login name and password that the user enters during login.
- **html_body.** A string that identifies the HTML body of the document that CRM Desktop loads in Internet Explorer.
- **status.** A string that indicates the type of callback that the interactive authentication handler uses. It can include one of the following values:
 - **before.** CRM Desktop SSO sends the callback before it navigates the Web browser to a URL.
 - **finished.** CRM Desktop SSO sends the callback after the page download finishes.
 - **cancelled.** CRM Desktop SSO sends the callback if the user cancels the login and closes the dialog box.

JavaScript requires that you enclose each string in double quotes.

- **title.** A string that contains the title of the dialog box. CRM Desktop SSO displays this title in the dialog box during interactive authentication. The default configuration that the SSO Connector uses adds the following prefix to any page title that the Siebel Server renders:

SSO:

- **dialog.** A string that identifies the dialog object. For more information, see [“Dialog Object” on page 320](#).

Dialog Object

The dialog object is a subobject of the `ia_state` object. It defines the size, position, visibility, and title of the SSO login dialog box that CRM Desktop SSO displays during interactive authentication. It includes the following properties:

- **width.** A number that determines the width of the dialog box, in pixels.
- **height.** A number that determines the height of the dialog box, in pixels.
- **visible.** A Boolean value. If True, then CRM Desktop SSO displays the dialog box.
- **title.** A string that contains the title of the dialog box. CRM Desktop SSO displays this title in the dialog box during interactive authentication.

Example Code That Customizes CRM Desktop SSO

The following code comes predefined with CRM Desktop:

```
include("utils.js", "_utils");
include("core.js", "_core");
_utils.logger = logger;
```



```

_utils.sso_client = sso_client;
_core.sso_client = sso_client;
_core.settings_cache = settings_cache;

var Utils = _utils.Utils;
var Lib = _core.Lib;
var CookieManager = new Lib.CookieManager();
var SSOConfiguration = {
    "CookieBuffer": "DomainCookies",
    "AuthType": settings.get("AuthType"),
    "EndPointRegExp": settings.get("EndPointRegExp"),
    "SuccessLoginRegExp": settings.get("SuccessLoginRegExp")
};
var interactive_params = {
    "InitialWidth": 1024,
    "InitialHeight": 768,
    "InitialTitle": "SSO",
    "InitialTitlePrefix": "SSO",
    "CheckFn": InteractiveCheckFunction
};

var persistent_cookies = true;

sso_client.request_handler = process_request;

function InteractiveCheckFunction (ia_state, ia_result, original_request) {
    var path = (original_request.get_object()).replace(/\\/i, "").split("?");

    if (SSOConfiguration.EndPointRegExp == "") {
        is_original_url = ( (ia_state.url).toLowerCase()
).indexOf((original_request.get_server() + path[0]).toLowerCase() ) != -1;
    } else {
        is_original_url = (new RegExp(SSOConfiguration.EndPointRegExp,
'i')).test(ia_state.url)
    }
    if (!is_original_url && ia_state.status == "before") {
        persistent_cookies = false;
    }
    if (ia_state.status == "finished" && ia_state.html_body != "") {
        if (is_original_url) {
            var regexp = SSOConfiguration.SuccessLoginRegExp == "" ? "FAULTSTRING.
*?10944629" : SSOConfiguration.SuccessLoginRegExp;

            if ( ia_state.html_body.match( new RegExp(regexp,'mi') ) != null ) {
                return true;
            }
        }
        return false;
    }
}

function RequestData(request, clear_session) {
    if (GetCache() != "" && CookieManager.GetAllCookies().length == 0) {
        Utils.SetRequestCookies(request, GetCache());
    }
    var response = Lib.ExecuteRequest(request, CookieManager, clear_session);
}

```

```

if (CookieManager.GetAllCookies().length > 0) {
    UpdateCache(CookieManager.GetAllCookiesAsString());
}
return response;
}

function RedefineInteractiveDescriptor (response, redefine_location) {
    return function (descriptor, ia_state) {
        Utils.Log("Clear browser cookies", "info");
        var cookies = Utils.ParseCookieString(ia_state.cookies);
        var cookie = {};
        for (var i = 0, len = cookies.length; i < len; i++) {
            cookie = Utils.ParseCookie(cookies[i]);
            ia_state.cookies = cookie.Name + " ";
        }
        if (response !== null) {
            if (redefine_location) {
                descriptor.SetEndpoint(Utils.GetSpecifi cHeader(response.get_headers(),
                    "Location")[0]);
            }
        }
        return descriptor;
    }
}

function UpdateCache (value) {
    Utils.Log("Update cache cookies", "info");
    var cached = Utils.ParseCookies(Utils.ParseCookieString(GetCache()));
    var browser = Utils.ParseCookies(Utils.ParseCookieString(value));
    for (var i = 0, iLen = browser.length; i < iLen; i++) {
        isset = false;
        for (var j = 0, jLen = cached.length; j < jLen; j++) {

            if (browser[i].Name == cached[j].Name) {
                isset = true;
                cached[j] = browser[i];
            }
        }
        if(!isset) {
            cached.push(browser[i]);
        }
    }
    settings_cache.set(SSOConfigurati on.CookieBuffer, CookieManager.ConvertCookiesToString(
        cached));
}

function ClearCache () {
    settings_cache.set(SSOConfigurati on.CookieBuffer, "");
}

function GetCache() {
    return settings_cache.get(SSOConfigurati on.CookieBuffer);
}

function process_request(sso_client_request) {
    var ignore_cache = settings.get("IgnoreCache");
    var response;
    if (SSOConfigurati on.AuthType == "NTLM") {
        var creds = sso_client_request.get_credentials();
    }

```

```

creds.set_username('');
creds.set_password('');
creds.set_auth_schemes('n');
response = sso_client.execute_request(sso_client_request);
if (response == null) Utils.Log("No response received", "warning");
} else {
var request_x_type = Utils.GetSpecific-
Header(sso_client_request.get_headers(), "X-CRMD-TYPE");
if (request_x_type == "verify" && ignore_cache == "1") {
ClearCache();
}
response = RequestData(sso_client_request, false);
var status_code = response.get_status_code();
if (request_x_type == "logout") {
response = null;
} else if (status_code == "401" || status_code == "407") {
Utils.Throw("not_val id", "script_not_val id_sso_ntlm_attempt");

Utils.Log("Attempt to use SSO mode with NTLM-protected EAI", "info");
return null;
} else if (status_code == "302" || Utils.Transitions.IsHtml(response)) {
var redefine = status_code == "302" ? true : false;
Utils.Log("Interactive mode initialized", "info");
var result = Lib.RunInteractive(sso_client_request, interactive_params,
CookieManager, RedefineInteractiveDescriptor(response, redefine));
if (result == "success") {
if (persistent_cookies == true && ignore_cache == "1" && settings.
get("UserChanged") == "1") {
Utils.Throw("not_val id", "script_not_val id_clear_cookies");
Utils.Log("Persistent cookies exist", "info");
return null;
} else {
persistent_cookies = true;
Utils.Log("Login successful", "info");
}
} else if (result == "canceled") {
Utils.Log("Login canceled", "info");
sso_client.raise_cancel_exception("User canceled login dialog.");
return null;
}
Utils.Log("Interactive mode finished", "info");
response = RequestData(Utils.CloneRequest(sso_client,
sso_client_request, null), true);
}
}
return response;
}

```


A

Reference Information for Siebel CRM Desktop

This appendix describes reference information for Siebel CRM Desktop. It includes the following topics:

- [Registry Keys You Can Use with Siebel CRM Desktop on page 325](#)
- [Parameters You Can Use with Log Files on page 335](#)
- [Filters in the CRM Desktop Filter - Edit Criterion Dialog Box on page 344](#)
- [Threshold That Siebel CRM Desktop Uses to Display the Confirm Synchronization Tab on page 346](#)
- [Files That the Metadata and Customization Package Contains on page 348](#)
- [IBM Notes Field Types and Equivalent Converter Classes on page 358](#)

Registry Keys You Can Use with Siebel CRM Desktop

This topic describes Windows Registry keys you can use with Siebel CRM Desktop. It includes the following topics:

- [Registry Keys That Affect Siebel CRM Desktop Behavior on page 325](#)
- [Registry Keys That Affect Credentials on page 329](#)
- [Registry Keys That Affect CRM Desktop SSO on page 330](#)

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 [“Using the Windows Registry to Control Siebel CRM Desktop” on page 103](#).

Registry Keys That Affect Siebel CRM Desktop Behavior

[Table 20](#) describes the Windows Registry keys that you can modify to change Siebel CRM Desktop behavior. In the Registry Editor (regedit), you can modify these keys in the following path:

HKEY_CURRENT_USER\Software\Oracle\CRM Desktop for IBM Notes

Table 20. Windows Registry Keys That Affect Behavior of the CRM Desktop Add-In

| Windows Registry Key | Description |
|------------------------------|--|
| AppLanguageID | ID of the current installation package of IBM Notes. |
| customization_path | Path to the customization package files. |
| DestinationLocation | Name of the IBM Notes location where the add-in is installed. |
| DestinationStore | Name of the IBM Notes location where the add-in is installed. |
| DisableLiveUpdate | Specifies if the live update feature is allowed. The following values are valid: <ul style="list-style-type: none"> ■ 0. Live update is allowed. ■ 1. Live update is not allowed. |
| DisableSyncConfirmation | Suppresses confirmation for deleting an object. The following values are valid: <ul style="list-style-type: none"> ■ 1. Suppress confirmation for deleting an object. ■ 0. Do not suppress confirmation for deleting an object. <p>The corresponding attribute in the Ln_connector_configuration.xml file of the customization package automatically overwrites the DisableSyncConfirmation key.</p> |
| 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: <ul style="list-style-type: none"> ■ 1. Estimate automatically. ■ 0. Do not estimate automatically. <p>This entry is not accessible through the administrative interface.</p> |
| HTTPClient:AcceptCompression | A flag that instructs the Web Service Connector to accept zipped HTTP content. This key is not accessible through the administrative interface. |
| HTTPClient:CompressOutgoing | A flag that instructs the Web Service Connector to send zipped HTTP content. This key 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. |

Table 20. Windows Registry Keys That Affect Behavior of the CRM Desktop Add-In

| Windows Registry Key | Description |
|----------------------------|--|
| HTTPClient: SendRetryCount | The count for the connection retries in milliseconds. |
| HTTPClient: SendTimeout | The timeout for the sending requests in milliseconds. |
| LogoutTimeout | The time to wait in milliseconds after CRM Desktop sends the log out request. This parameter stops the session without waiting for the reply from the server. |
| Page: Feedback: AttachLog | Specifies a log for the feedback form. The following values are valid: <ul style="list-style-type: none"> ■ 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: <ul style="list-style-type: none"> ■ 0. Do not use a proxy server. ■ 1. Use a proxy server. |
| RunPeriodicalSyncAlways | Determines if CRM Desktop starts a scheduled synchronization at the scheduled time or waits until IBM Notes is idle. The following values are available: <ul style="list-style-type: none"> ■ 0. Wait until IBM Notes 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. 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. |

Table 20. Windows Registry Keys That Affect Behavior of the CRM Desktop Add-In

| Windows Registry Key | Description |
|--------------------------------|--|
| SharedByDefault: NewItems | <p>Determines how CRM Desktop shares newly created IBM Notes items. The SharedByDefault: NewItems registry key controls the following option:</p> <p>Always share with Siebel new: Calendar Entry, Contacts, To Do items</p> <p>To access this option, the user right-clicks the CRM Desktop icon in the system tray, chooses Options, and then clicks the Advanced tab in the CRM Desktop - Options dialog box.</p> |
| Siebel: HideSavePasswordOption | <p>Determines how CRM Desktop displays the Save Password check box on the login screen. The following values are valid:</p> <ul style="list-style-type: none"> ■ 0. Display the Save Password check box. CRM Desktop displays this check box, by default. ■ 1. Disable the Save Password check box. |
| Siebel: MetaInfoFilePath | <p>Describes the path to the siebel_meta_info.xml file in the customization package.</p> |
| SuppressSyncEstimating | <p>Estimates the number of objects that CRM Desktop will synchronize for the current user. The following values are valid:</p> <ul style="list-style-type: none"> ■ 0. Do estimate the number of the objects. ■ 1. Do not estimate the number of the objects. If SuppressSyncEstimating is 1, then CRM Desktop does not use the logic that the MaximumSyncPassthrough key controls. It uses MaximumSyncPassthrough only after it estimates the number objects it must synchronize. |
| SuppressSyncIssues | <p>If a synchronization problem occurs, then this key determines if CRM Desktop changes the application icon in the system tray to an exclamation point and displays a message. The following values are valid:</p> <ul style="list-style-type: none"> ■ 0. Change the icon in the system tray and display a message. The default value is 0. ■ 1. Do not change the icon in the system tray and do not display a message. |

Registry Keys That Affect Credentials

Table 21 describes the Windows Registry keys that you can modify to change how Siebel CRM Desktop handles credentials. In the Registry Editor (regedit), you can modify these keys in the following path:

HKEY_CURRENT_USER\Software\Oracle\CRM Desktop for IBM Notes\Credentials

Table 21. Windows Registry Keys That Affect Credentials

| Windows Registry Key | Description |
|----------------------|--|
| RememberPassword | <p>Determines how CRM Desktop remembers the password. You can use one of the following values:</p> <ul style="list-style-type: none"> ■ 0. Remember the password for all IBM Notes sessions and add a check mark to the Save Password check box in the CRM Desktop - Login dialog box. ■ 1. Remember the password only for the current IBM Notes session and remove the check mark from the Save Password check box in the CRM Desktop - Login dialog box. |
| Siebel:ComponentName | <p>Siebel Server component name that processes incoming requests. The following value is the default value:</p> <p>eai /enu</p> <p>The Siebel:ComponentName key is appended to the URL. For more information, see “Overriding Windows Registry Keys That Locate the Siebel Server” on page 104.</p> |
| Siebel:LoginName | The name of the CRM Desktop user who is currently logged in. |
| Siebel:Password | Stores the user password in binary format. |
| Siebel:Protocol | <p>Defines the URL protocol. The default is http. The following values are valid:</p> <ul style="list-style-type: none"> ■ http ■ https |
| Siebel:RequestSuffix | <p>Sets the suffix of the URL to the Siebel Server. The following is the default value:</p> <p>?SWEExtSource=WebService&SWEExtCmd=Execute&WSSOAP=1</p> <p>For more information, see “Overriding Windows Registry Keys That Locate the Siebel Server” on page 104.</p> |

Table 21. Windows Registry Keys That Affect Credentials

| Windows Registry Key | Description |
|----------------------|--|
| Siebel:Server | Sets the host name of the URL. The default value is <i>empty</i> . |
| Siebel:ServerPort | Sets the port of the URL. The default value is 80. |

Registry Keys That Affect CRM Desktop SSO

This topic describes the Windows registry keys that affect CRM Desktop SSO. It includes the following topics:

- [“Windows Registry Keys You Must Set to Enable CRM Desktop SSO” on page 330](#)
- [“Registry Keys That Control SSO for Siebel CRM Desktop” on page 333](#)
- [“Registry Keys That Control SSO for Credentials” on page 334](#)

You must not modify the following Windows registry keys:

- **UpdateLastCheck.** A timestamp value in 100 UTC nanosecond units that stores the time and date of the last successful or unsuccessful update attempt.
- **UpdateZipTimestamp.** A timestamp value in 100 UTC nanosecond units that stores the time and date of the last successful update.

To establish an outgoing HTTP connection, CRM Desktop SSO also uses the following registry keys:

- HTTPClient keys
- Proxy keys

For more information, see [“Overview of Customizing Authentication” on page 285](#).

Windows Registry Keys You Must Set to Enable CRM Desktop SSO

[Table 22](#) describes command-line parameters that you must use with the msixexec.exe installer. Each command-line parameter modifies a Windows Registry key that CRM Desktop SSO requires. For more information, see [“Using the Windows Command Line to Set Optional Parameters” on page 98](#).

In the Registry Editor (regedit), you can modify these keys in the following path:

HKEY_CURRENT_USER\Software\Oracle\CRM Desktop\SSO

Table 22. Windows Registry Keys You Must Set to Enable CRM Desktop SSO

| Command Line Parameter | Description |
|------------------------|---|
| SSOENABLE | <p>Specifies to enable CRM Desktop SSO. You can use one of the following values:</p> <ul style="list-style-type: none"> ■ 0. Disable CRM Desktop SSO. The default value is 0. ■ 1. Enable CRM Desktop SSO. <p>If disabled, then CRM Desktop SSO it is not active when CRM Desktop communicates with Siebel CRM.</p> <p>CRM Desktop copies the value that the SSOENABLE parameter contains to the SSO\Enable registry key.</p> |
| SSOSCRIPTFILENAME | <p>Specifies the name of the JavaScript file that implements the CRM Desktop SSO logic. This file must define the entry point for SSO scenario handling. The file name must be relative to the directory that the SSOSCRIPTINCLUDEPATH parameter specifies.</p> <p>The default value is sso.js.</p> <p>CRM Desktop copies the value that the SSOSCRIPTFILENAME parameter contains to the SSO\ScriptFileName registry key.</p> <p>For more information about the entry point, see “Request Handler Function” on page 312.</p> |
| SSOSCRIPTINCLUDEPATH | <p>Specifies the directory path where the CRM Desktop SSO script file resides. This directory must also contain any files that this script references. For example:</p> <p style="padding-left: 40px;">C:\users\user1\AppData\Roaming\Oracle\CRM Desktop\bin</p> <p>The default value is an empty string.</p> <p>CRM Desktop copies the value that the SSOSCRIPTINCLUDEPATH parameter contains to the SSO\ScriptIncludePath registry key.</p> <p>For autoupdate, you must use the SSOSCRIPTINCLUDETEMPLATE parameter instead of the SSOSCRIPTINCLUDEPATH parameter. For more information, see “Installing CRM Desktop SSO If You Use Autoupdate” on page 293.</p> |

Table 22. Windows Registry Keys You Must Set to Enable CRM Desktop SSO

| Command Line Parameter | Description |
|------------------------|--|
| SSOUPDATEDISABLE | <p>Specifies to enable autoupdate. You can use one of the following values:</p> <ul style="list-style-type: none">■ 0. Enable autoupdate. The default value is 0.■ 1. Disable autoupdate. <p>CRM Desktop copies the value that the SSOUPDATEDISABLE parameter contains to the SS0\UpdateDisable registry key. For more information, see "Installing CRM Desktop SSO If You Use Autoupdate" on page 293.</p> |
| SSOURL | <p>Specifies the URL or UNC path that CRM Desktop uses to download autoupdate information.</p> <p>The default value is an empty string.</p> <p>CRM Desktop copies the value that the SSOURL parameter contains to the SS0\UpdateZIPURL registry key.</p> <p>You must make sure you set this parameter during deployment. If you do not, then autoupdate will not work.</p> <p>If you use external provisioning, then the SSOURL parameter is not required. For more information, see "Installing CRM Desktop SSO If You Use Autoupdate" on page 293.</p> |

Table 22. Windows Registry Keys You Must Set to Enable CRM Desktop SSO

| Command Line Parameter | Description |
|--------------------------|--|
| SSOCHECKINTERVAL | <p>Specifies the timestamp value that CRM Desktop uses as the minimum time interval between update attempts. It measures this value in 100 nanosecond units. If this value is smaller than 36000000000 (1 hour), then CRM Desktop ignores this smaller value and sets the interval to 36000000000.</p> <p>The default value is 864000000000 (24 hours).</p> <p>CRM Desktop copies the value that the SSOCHECKINTERVAL parameter contains to the SS0\UpdateCheckInterval registry key.</p> |
| SSOScriptINCLUDETEMPLATE | <p>Specifies the template that CRM Desktop uses to create a unique directory name. It uses this directory to store the scripts that it downloads from the URL that the SSURL parameter identifies. You must use the following format:</p> <p><i>%path%</i></p> <p>For example:</p> <p><i>%appdata%</i></p> <p>The default value is %appdata%\Invisibl eSS0\Script.</p> <p>CRM Desktop copies the value that the SSOScriptINCLUDETEMPLATE parameter contains to the SS0\ScriptIncludeTemplate registry key.</p> |

Registry Keys That Control SSO for Siebel CRM Desktop

Table 23 describes the Windows Registry keys that you can modify to control SSO for Siebel CRM Desktop. For more information, see the topic about Regular Expression Syntax for JavaScript at the Microsoft Developer Network web site. You can modify these keys in the following path in the Registry Editor (regedit):

HKEY_CURRENT_USER\Software\Oracle\CRM Desktop

Table 23. Windows Registry Keys That Control SSO for Siebel CRM Desktop

| Windows Registry Key | Description |
|----------------------|---|
| AuthType | <p>Determines the authentication type that the CRM Desktop-Login dialog box displays, by default. You can set it to one of the following values:</p> <ul style="list-style-type: none"> ■ DIRECT ■ NTLM ■ SSO <p>You can set this default value externally before the first time CRM Desktop runs. If you do this, then you must make sure the external value you provide is the same value that the following key contains:</p> <p style="padding-left: 40px;">HKEY_CURRENT_USER\Software\Oracle\CRM Desktop\SSO\Enable</p> <p>You must set this Enable key to one of the following values:</p> <ul style="list-style-type: none"> ■ 1 for NTLM or SSO ■ 0 for DIRECT |
| EndpointRegExp | <p>Sets the regular expression that CRM Desktop uses to evaluate the rule that stops the interactive authentication. It must match the URL of the last web page that the interactive authentication displays.</p> <p>If you do not set EndpointRegExp, then CRM Desktop compares the URL that displays the login dialog box to the URL that displays the last web page.</p> |
| SuccessLoginRegExp | <p>Sets the regular expression that CRM Desktop uses to evaluate the rule that stops the interactive authentication. It must match the body of the last web page that the interactive authentication displays.</p> <p>If you do not set SuccessLoginRegExp, then CRM Desktop compares the page contents to the following regular expression:</p> <p style="padding-left: 40px;">FAULTSTRING.*?10944629</p> <p>This expression matches the response that the Siebel EAI server returns in reply to an empty GET request. It is a SOAP FAULT response that includes a message that indicates that it passed the empty request body. This is the default value.</p> |

Registry Keys That Control SSO for Credentials

Table 24 describes the Windows Registry keys that you can modify to control SSO for credentials. You can modify these keys in the following path in the Registry Editor (regedit):

HKEY_CURRENT_USER\Software\Oracle\CRM Desktop\Credentials

Table 24. Windows Registry Keys That Control SSO for Credentials

| Windows Registry Key | Description |
|----------------------|---|
| Siebel:SSOUser | <p>Specifies the value for the user name that Siebel CRM SSO enforces for installation.</p> <p>Applicable only if the authorization type is SSO.</p> <p>If you:</p> <ul style="list-style-type: none"> ■ Set Siebel:TrustToken. The User name field is read-only and Siebel CRM SSO automatically populates a value in this field. ■ Do not set Siebel:TrustToken. The user can enter a value in the User name field. |

Parameters You Can Use with Log Files

This topic describes parameters that you can use with log files. It includes the following topics:

- [Parameters You Can Use with the General Log on page 336](#)
- [Parameters You Can Use with the Exception Log on page 339](#)
- [Parameters You Can Use with the Crash Log on page 339](#)
- [Parameter You Can Use with the SOAP Log on page 342](#)
- [Parameters You Can Use with the Synchronization Log on page 343](#)

An integer that you can use with a parameter represents a DWORD value in the registry in Windows XP or a 32-bit DWORD value in the registry in Windows Vista. For more information, see [“Using the Windows Registry to Control Siebel CRM Desktop” on page 103](#).

Parameters You Can Use with the General Log

Table 25 describes the parameters that you can use with the General Log with Siebel CRM Desktop.

Table 25. Parameters You Can Use with the General Log

| Parameter | Description |
|-------------------|---|
| enable | <p>You can use one of the following values:</p> <ul style="list-style-type: none"> ■ 1. General Log is enabled. ■ 0. General Log is disabled. <p>To disable the general log, you must use the Windows Registry. You cannot use the Logging Configuration dialog box. For more information, see “Assigning Logging Profiles for Siebel CRM Desktop” on page 117.</p> |
| log_level | <p>Defines logging verbosity. You can use one of the following integers:</p> <ul style="list-style-type: none"> ■ 4294967295. Disable all. ■ 0. Enable all. ■ 1000. Debug. ■ 2000. Information messages. ■ 3000. Warnings. ■ 4000. Errors. ■ 5000. Fatal errors. <p>For more information, see “Setting Logging Verbosity” on page 119.</p> |
| out_file | <p>You can include a string that identifies the file name for the general log. For example, assume you use the following value:</p> <p style="text-align: center;">log.txt</p> <p>In this example, the first file name is log.0000.txt, the second file name is log.0001.txt, and so on. If you do not include the out_file parameter, then CRM Desktop uses log.txt as the default value.</p> |
| enable_dbg_window | <p>You can use one of the following values:</p> <ul style="list-style-type: none"> ■ 1. Enables output through the OutputDebugString that CRM Desktop displays in the VS Debug Output window during a debugging session. The OutputDebugString is a string that CRM Desktop sends to the debugger for display. The VS Debug Output window is a window in Microsoft Visual Studio that displays debugging results. For more information about Visual Studio, see the documentation at the Microsoft TechNet Web site. ■ 0. Disables the General Log. |

Table 25. Parameters You Can Use with the General Log

| Parameter | Description |
|----------------------|--|
| enable_cout | <p>You can use one of the following values:</p> <ul style="list-style-type: none"> ■ 1. Enables logging in the console. ■ 0. Disables logging in the console. |
| max_size_bytes | <p>You can use an integer that sets the maximum number of bytes that a single log file can contain. For example, if you set max_size_bytes to 10485760 bytes, and if the current log reaches 10485760 in size, then CRM Desktop creates a new log file. This behavior is similar to setting the rotate_on_start_only parameter to 0.</p> |
| reuse_not_exceeded | <p>Determines the file that CRM Desktop uses when a new logging session starts. You can use one of the following values:</p> <ul style="list-style-type: none"> ■ 1. If the size of the most current log file is less than the value that the max_size_bytes parameter sets, then CRM Desktop reuses this file. 1 is the default value. ■ 0. CRM Desktop does not reuse the most current log file. |
| rotate_on_start_only | <p><i>Log file rotation</i> is a configuration that CRM Desktop uses to prevent log files from growing indefinitely. You can use one of the following values:</p> <ul style="list-style-type: none"> ■ 1. Examines the log file size that exists when IBM Notes starts. If this log file size is larger than the value that the max_size_bytes parameter specifies, then CRM Desktop creates a new log file and then logs all subsequent entries to this new file. The log file size can exceed the value that the max_size_bytes parameter specifies. CRM Desktop stores all log entries for one IBM Notes session to one file. 1 is the default value. ■ 0. Examines the log file size every time CRM Desktop writes to the log. If this log file size is larger than the value that the max_size_bytes parameter specifies, then CRM Desktop creates a new log file. Multiple log files can exist for a single IBM Notes session. |
| file_count | <p>An integer that specifies the maximum number of rotated log files.</p> |
| initial_erase | <p>You can use one of the following values:</p> <ul style="list-style-type: none"> ■ 1. Deletes the contents of all general log files when CRM Desktop starts. ■ 0. Does not delete the contents of any general log file when CRM Desktop starts. |

Table 25. Parameters You Can Use with the General Log

| Parameter | Description |
|-------------|---|
| time_format | <p>A string that specifies the date and time format in the log files. You can use one of the following values:</p> <ul style="list-style-type: none"> ■ \$nano. Nanoseconds. ■ \$micro. Microseconds. ■ \$milli. Milliseconds. ■ \$ss. A 2 digit second. ■ \$mm. A 2 digit minute. ■ \$hh. A 2 digit hour. ■ \$dd. A 2 digit day. ■ \$MM. A 2 digit month. ■ \$yy. A 2 digit year. ■ \$yyyy. A four digit year. <p>Nanoseconds or microseconds are available only if the system clock allows nanoseconds or microseconds. If the system clock does not allow nanoseconds or microseconds, and if you specify nanoseconds or microseconds, then CRM Desktop pads the value that it creates for the log entry with zeros.</p> |
| log_format | <p>A string that specifies the logging message format. You can use one of the following values:</p> <ul style="list-style-type: none"> ■ %index%. Sequential message index. ■ %time%. Time in the format that the time_format parameter sets. ■ %thread%. Thread ID that CRM Desktop uses to log the message. ■ %level%. Message logging level in human readable format. ■ %level_num%. Message logging level in numeric format. ■ %log_src%. Source of the logging message. This source can be a CRM Desktop internal component. For example, a connector or application manager. |

Table 25. Parameters You Can Use with the General Log

| Parameter | Description |
|-----------|---|
| starter | A string that specifies the first message that CRM Desktop adds to the log file when it starts logging. For example: --- logging is initialized --- |
| finisher | A string that specifies the last message that CRM Desktop adds to the log file when it stops logging. For example: --- logging is finished --- If a log does not include the finisher message, then it indicates that CRM Desktop stopped logging abnormally. |

Parameters You Can Use with the Exception Log

Table 26 describes the parameters that you can use with the Exception Log.

Table 26. Parameters You Can Use with the Exception Log

| Parameter | Description |
|----------------|---|
| enable | You can use one of the following values: <ul style="list-style-type: none"> ■ 1. Exception Log is enabled. ■ 0. Exception Log is disabled. |
| file_count | An integer that specifies the maximum number of rotated log files. |
| initial_erase | You can use one of the following values: <ul style="list-style-type: none"> ■ 1. Deletes the contents of all exception log files when CRM Desktop starts. ■ 0. Does not delete the contents of any exception log file when CRM Desktop starts. |
| max_size_bytes | An integer that sets the maximum number of bytes that a single log file can contain. For more information, see the description for the max_size_bytes parameter in “Parameters You Can Use with the General Log” on page 336 . |
| out_file | A string that specifies the base file name that CRM Desktop uses for logging. |

Parameters You Can Use with the Crash Log

Table 27 describes the parameter that you can use with the crash log. You can use the following parameters:

- any_in_trace
- all_in_trace

■ ex_white_list

■ ex_black_list

To separate values in these parameters, you can use the following symbols:

■ , (comma)

■ ; (semicolon)

■ | (vertical bar)

■ / (forward slash)

■ \t (backward slash with a t)

■ (single space)

Table 27. Parameters You Can Use with the Crash Log

| Parameter | Description |
|-----------|--|
| enable | <p>You can use one of the following values:</p> <ul style="list-style-type: none"> ■ 1. Crash log is enabled. ■ 0. Crash log is disabled. |
| count | An integer that specifies the maximum of old log files that CRM Desktop preserves in the output logging directory. |
| on_top | <p>A string that specifies the filtering condition for the crash log. CRM Desktop applies this filtering only if the item you specify resides at the start of the stack trace when the failure occurs. For example, you can use the following value:</p> <p style="text-align: center;">CRMDesktop3D.dl I</p> <p>If you do not specify a value, then CRM Desktop ignores this parameter.</p> |
| on_top_op | <p>You can use one of the following values:</p> <ul style="list-style-type: none"> ■ 0. Combine the item that the on_top parameter filters with the item that the any_in_trace parameter specifies or with the item that the all_in_trace parameter specifies. Use an OR logic operation. ■ 1. Combine the item that the on_top parameter filters with the item that the any_in_trace parameter specifies and with the item that the all_in_trace parameter specifies. Use an AND logic operation. ■ 2. Combine the item that the on_top parameter does not filter with the item that the any_in_trace parameter specifies or with the item that the all_in_trace parameter specifies. Use an OR logic operation. ■ 3. Combine the item that the on_top parameter does not filter with the item that the any_in_trace parameter specifies and with the item that the all_in_trace parameter specifies. Use an AND logic operation. |

Table 27. Parameters You Can Use with the Crash Log

| Parameter | Description |
|---------------|---|
| any_in_trace | <p>A string that specifies a list of items. If the stack trace includes an item you specify, then CRM Desktop saves a crash log for this item. For example, you can use the following value:</p> <p style="text-align: center;">Ntdll.dll, Kernel32.dll</p> <p>In this example, if the stack trace includes an entry for Ntdll.dll or Kernel32.dll, then CRM Desktop saves a crash log for this item.</p> |
| all_in_trace | <p>A string that specifies a list of items. If the stack trace includes all the items you specify, then CRM Desktop saves a crash log for these items. For example, you can use the following value:</p> <p style="text-align: center;">Ntdll.dll, Kernel32.dll</p> <p>In this example, if the stack trace includes an entry for Ntdll.dll and an entry for Kernel32.dll, then CRM Desktop saves a crash log for these items.</p> |
| delay_ms | <p>An integer that specifies a delay in milliseconds. If the time that elapses after a previous exception exceeds the value you specify, then CRM Desktop saves a crash log.</p> <p>The default value is 0. This default makes sure that CRM Desktop always saves a crash log.</p> |
| ex_white_list | A string that specifies a list of C++ exception names. If an exception occurs for an exception name you specify, then CRM Desktop creates a minilog for this exception. |
| ex_black_list | A string that specifies a list of C++ exception names that CRM Desktop uses to not create a minilog. If the white_list parameter is empty, then CRM Desktop creates a log for all exceptions except the exceptions that the black_list parameter lists. |
| sigabrt | <p><i>SIGABRT</i> is a signal that CRM Desktop sends to a process to tell it to end. You can use one of the following values:</p> <ul style="list-style-type: none"> ■ 1. Intercept the <i>SIGABRT</i> signal to save a crash log. You must use this parameter only for troubleshooting. Do not enable it permanently. ■ 0. Do not intercept the <i>SIGABRT</i> signal to save a crash log. |
| minidump_type | <p>An integer that specifies the minilog type. The default value is 1 (MiniDumpNormal MiniDumpWithDataSegs).</p> <p>Use this parameter with caution because other values could increase log size significantly. For more information, see documentation about the DumpType parameter of the MiniDumpWriteDump function at the Microsoft MSDN web site.</p> |

Parameter You Can Use with the SOAP Log

Table 28 describes the parameters that you can use with the SOAP log.

Table 28. Parameters You Can Use with the SOAP Log

| Parameter | Description |
|-------------------|--|
| enable | You can use one of the following values: <ul style="list-style-type: none"> ■ 1. SOAP log is enabled. ■ 0. SOAP log is disabled. |
| enable_dbg_window | You can use one of the following values: <ul style="list-style-type: none"> ■ 1. Enables output through the OutputDebugString. The SOAP log displays in the VS Debug Output window during a debugging session. ■ 0. Disables output through the OutputDebugString. The SOAP log does not display in the VS Debug Output window during a debugging session. |
| enable_cout | You can use one of the following values: <ul style="list-style-type: none"> ■ 1. Enables logging to the console. ■ 0. Disables logging to the console. |
| file_count | An integer that specifies the maximum number of rotated log files. |
| root_tag_name | A string that specifies the root XML tags that CRM Desktop uses in generated XML files. The default value is soap_comm_log. If you use the following value, then it does not write the opening or closing root tags: <pre>""</pre> |
| file_extension | A string that specifies the file extension that CRM Desktop uses for generated log files. The default value is xml. |
| bin_mode | You can use one of the following values: <ul style="list-style-type: none"> ■ 1. CRM Desktop uses the \n format for new line termination. ■ 0. CRM Desktop uses the \r\n format for new line termination. |

Parameters You Can Use with the Synchronization Log

Table 29 describes the parameters that you can use with the synchronization log.

Table 29. Parameters You Can Use with the Synchronization Log

| Parameter | Description |
|------------|---|
| enable | You can use one of the following values: <ul style="list-style-type: none">■ 1. Synchronization log is enabled.■ 0. Synchronization log is disabled. |
| file_count | An integer that specifies the maximum number of rotated log files. |
| out_file | A string that specifies the base file name that CRM Desktop uses for logging. |

Filters in the CRM Desktop Filter - Edit Criterion Dialog Box

Table 30 describes the filters that the user can specify in the CRM Desktop Filter - Edit Criterion dialog box. The user chooses values in the Condition field and the Value field to specify a filter. It is recommended that the user use the = (equal) operator or the <> (not equal) operator only for the Exact Date filter. For more information, see [“Controlling the Date Range in the Filter Records Tab” on page 129](#).

Table 30. Filters in the CRM Desktop Filter - Edit Criterion Dialog Box

| Filter | Description |
|-----------------|--|
| Days From Today | <p>Sets a date range that includes a specific number of days before or after today. The user can enter one of the following values:</p> <ul style="list-style-type: none"> ■ Positive value. Indicates the number of days to include in the filter starting with tomorrow and continuing into the future. ■ Negative value. Indicates the number of days to include in the filter starting with yesterday and continuing into the past. <p>For example, the user can specify one of the following filters:</p> <ul style="list-style-type: none"> ■ Start < 2 Days From Today. If today is January 22, 2012, then CRM Desktop synchronizes activities that start before January 24, 2012. ■ Start > 2 Days From Today. If today is January 22, 2012, then CRM Desktop synchronizes activities that start after than January 24, 2012. ■ Start <= 2 Days From Today. If today is January 22, 2012, then CRM Desktop synchronizes activities that start on or before January 24, 2012. ■ Start >= 2 Days From Today. If today is January 22, 2012, then CRM Desktop synchronizes activities that start on or after January 24, 2012. ■ Start < -2 Days From Today. If today is January 22, 2012, then CRM Desktop synchronizes activities that start before January 20, 2012. ■ Start > -2 Days From Today. If today is January 22, 2012, then CRM Desktop synchronizes activities that start after January 20, 2012. ■ Start <= -2 Days From Today. If today is January 22, 2012, then CRM Desktop synchronizes activities that start on or before January 20, 2012. ■ Start >= -2 Days From Today. If today is January 22, 2012, then CRM Desktop synchronizes activities that start on or after January 20, 2012. |

Table 30. Filters in the CRM Desktop Filter - Edit Criterion Dialog Box

| Filter | Description |
|-------------|---|
| Exact Date | <p>The user can choose an exact date from the calendar in the CRM Desktop Filter - Edit Criterion dialog box. CRM Desktop only synchronizes records for the date that the user chooses. For example, the following condition synchronizes activities that are due only for January 1, 2012:</p> <ul style="list-style-type: none"> ■ Field is Due ■ Condition is = ■ Value is January 1, 2012 |
| Month Ago | <p>Sets a date range that is related to 30 days prior to today. For example, the user can specify one of the following filters:</p> <ul style="list-style-type: none"> ■ Start < Month Ago. If today is January 31, 2012, then CRM Desktop synchronizes activities that start before January 1, 2012. ■ Start > Month Ago. If today is January 31, 2012, then CRM Desktop synchronizes activities that start after January 1, 2012. ■ Start <= Month Ago. If today is January 1, 2012, then CRM Desktop synchronizes activities that start on or before January 1, 2012. ■ Start >= Month Ago. If today is January 1, 2012, then CRM Desktop synchronizes activities that start on or after January 1, 2011. |
| Month Ahead | <p>Sets a date range that is related to 30 days after today. For example, the user can specify one of the following filters:</p> <ul style="list-style-type: none"> ■ Start < Month Ahead. If today is January 1, 2012, then CRM Desktop synchronizes activities that start before January 31, 2012. ■ Start > Month Ahead. If today is January 1, 2012, then CRM Desktop synchronizes activities that start after January 31, 2012. ■ Start <= Month Ahead. If today is January 1, 2012, then CRM Desktop synchronizes activities that start on or before January 31, 2012. ■ Start >= Month Ahead. If today is January 1, 2012, then CRM Desktop synchronizes activities that start on or after January 31, 2012. |
| Today | <p>Sets a date range that is related to today. For example, the user can specify one of the following filters:</p> <ul style="list-style-type: none"> ■ Start < Today. If today is January 15, 2012, then CRM Desktop synchronizes activities that start on or before January 14, 2012. ■ Start > Today. If today is January 15, 2012, then CRM Desktop synchronizes activities that start on or after January 16, 2012. ■ Start <= Today. If today is January 15, 2012, then CRM Desktop synchronizes activities that start on or before January 15, 2012. ■ Start >= Today. If today is January 15, 2012, then CRM Desktop synchronizes activities that start on or after January 15, 2012. |

Table 30. Filters in the CRM Desktop Filter - Edit Criterion Dialog Box

| Filter | Description |
|-----------|--|
| Tomorrow | <p>Sets a date range that is related to tomorrow. For example, the user can specify one of the following filters:</p> <ul style="list-style-type: none">■ Start < Tomorrow. If today is January 15, 2012, then CRM Desktop synchronizes activities that start on or before January 15, 2012.■ Start > Tomorrow. If today is January 15, 2012, then CRM Desktop synchronizes activities that start on or after January 17, 2012.■ Start <= Tomorrow. If if today is January 15, 2012, then CRM Desktop synchronizes activities that start on or before January 16, 2012.■ Start >= Tomorrow. If if today is January 15, 2012, then CRM Desktop synchronizes activities that start on or after January 16, 2012. |
| Yesterday | <p>Sets a date range that is related to yesterday. For example, the user can specify one of the following filters:</p> <ul style="list-style-type: none">■ Start < Yesterday. If today is January 15, 2012, then CRM Desktop synchronizes activities that start on or before January 13, 2012.■ Start > Yesterday. If today is January 15, 2012, then CRM Desktop synchronizes activities that start on or after January 15, 2012.■ Start <= Yesterday. If if today is January 15, 2012, then CRM Desktop synchronizes activities that start on or before January 14, 2012.■ Start >= Yesterday. If if today is January 15, 2012, then CRM Desktop synchronizes activities that start on or after January 14, 2012. |

Threshold That Siebel CRM Desktop Uses to Display the Confirm Synchronization Tab

Table 31 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 147.](#)

Table 31. Threshold That CRM Desktop Uses to Display the Confirm Synchronization Tab

| Object | Number of Deleted Records |
|--------------------------------------|---------------------------|
| Account | 3 |
| Account.Account_Note | 10 |
| Account.Assignment_Group.Association | 5 |
| Account.Business_Address (HOR) | 10 |

Table 31. Threshold That CRM Desktop Uses to Display the Confirm Synchronization Tab

| Object | Number of Deleted Records |
|--|---------------------------|
| 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 |
| 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 |
| Opportunity.Position.Association | 20 |

Table 31. Threshold That CRM Desktop Uses to Display the Confirm Synchronization Tab

| Object | Number of Deleted Records |
|------------------------------|---------------------------|
| Position | Not applicable |
| Sales_Method.Sales_Cycle_Def | Not applicable |

Files That the Metadata and Customization Package Contains

This topic describes Files in the Metadata and Customization Package. It includes the following topics:

- [Files in the Metadata on page 348](#)
- [Files in the Customization Package on page 355](#)

Files in the Metadata

This topic describes the files in the metadata.

Metadata File Types

[Table 32](#) lists each of the metadata file types that you must add in the Metadata Files view of the Administration - CRM Desktop screen. You must add only one type in the Metadata File Types view of the Administration - CRM Desktop screen. For example, Metadata Type = LN3.2_package, Metadata File Name = package. If you must support a language other than English, then see [“Metadata File Types That Support Languages” on page 354](#).

For more information, see [“Files in the Customization Package” on page 356](#).

Table 32. Metadata File Types

| Metadata File Name | Metadata File Type | Customizable |
|------------------------|--------------------|--------------|
| action_support.js | JavaScript | No |
| application_script.js | JavaScript | Yes |
| autoresolve_helpers.js | JavaScript | No |
| autoresolver.js | JavaScript | Yes |
| business_logic.js | JavaScript | Yes |
| code_pages.xml | XML | No |
| data_model.js | JavaScript | No |
| data_sources.xml | JavaScript | Yes |

Table 32. Metadata File Types

| Metadata File Name | Metadata File Type | Customizable |
|--------------------------------|--------------------|--------------|
| dialogs.xml | JavaScript | Yes |
| dxl_config.xml | JavaScript | Yes |
| form_helpers.js | JavaScript | No |
| info.xml | XML | Yes |
| In_connector_configuration.xml | XML | Yes |
| In_package_res.xml | XML | Yes |
| In_siebel_basic_mapping.xml | XML | Yes |
| mvg_dialogs.js | JavaScript | No |
| online_lookup.js | JavaScript | No |
| online_lookup_sbl.js | JavaScript | Yes |
| platform_configuration.xml | XML | Yes |
| recurrence_processing.js | JavaScript | NO |
| security_manager.js | JavaScript | No |
| siebel_meta_info.xml | XML | Yes |
| views.xml | XML | Yes |
| SD3.Lib.Interfaces.dxl | Script library | No |
| SD3.Lib.Utils.dxl | Script library | No |
| SD3.Lib.Errors.dxl | Script library | No |
| SD3.Lib.Strings.dxl | Script library | No |
| SD3.Lib.RegistryHelpers.dxl | Script library | No |
| SD3.Lib.Constants.dxl | Script library | No |
| SD3.Lib.MQ.dxl | Script library | No |
| SD3.Lib.BMProvider.dxl | Script library | No |
| SD3.Lib.Tools.2.dxl | Script library | No |
| SD3.Lib.HandlerHelpers.dxl | Script library | No |
| SD3.Lib.DataModel.dxl | Script library | No |
| SD3.Lib.Validator.dxl | Script library | No |
| SD3.Lib.Translator.dxl | Script library | No |
| SD3.Lib.FormHelpers.dxl | Script library | No |
| SD3.Lib.Actions.dxl | Script library | No |
| SD3.Lib.ProgressBar.dxl | Script library | No |

Table 32. Metadata File Types

| Metadata File Name | Metadata File Type | Customizable |
|--|--------------------|--------------|
| SD3.Lib.iCalendar.dxl | Script library | No |
| SD3.Lib.NativePIM.dxl | Script library | No |
| SD3.Lib.iCalendarUtil.dxl | Script library | No |
| SD3.Lib.Handlers.dxl | Script library | No |
| SD3.Lib.PickListDialogs.dxl | Script library | No |
| SBL.Lib.Constants.dxl | Script library | Yes |
| SBL.Lib.Strings.dxl | Script library | Yes |
| SBL.Lib.Helpers.dxl | Script library | Yes |
| SBL.Lib.SecurityRules.dxl | Script library | Yes |
| SBL.Lib.Recurrence.dxl | Script library | Yes |
| SBL.Lib.ActivityHandlers.dxl | Script library | Yes |
| SBL.Lib.BusinessLogic.dxl | Script library | Yes |
| SBL.Lib.Actions.dxl | Script library | Yes |
| SBL.Lib.Handlers.dxl | Script library | Yes |
| SBL.Lib.PickListDialogs.dxl | Script library | Yes |
| SBL.Lib.Forms.dxl | Script library | Yes |
| SD3.Lib.ProductAdapter.dxl | Script library | Yes |
| SD3.View.AssociationsByLeftId.dxl | View | No |
| SD3.View.AssociationsByRightId.dxl | View | No |
| SD3.View.AssociationSearch.dxl | View | No |
| SD3.View.CRMDocuments.dxl | View | No |
| SD3.View.MirrorDirectSearch.dxl | View | No |
| SD3.View.SettingsDefaults.dxl | View | No |
| SD3.View.ActivityHandlingSrc.dxl | View | No |
| SBL.View.Accounts.dxl | View | Yes |
| SBL.View.Activities.dxl | View | Yes |
| SBL.View.ActivitySearchByParams.dxl | View | Yes |
| SBL.View.ActivitySearchByPIMObjectId.dxl | View | Yes |
| SBL.View.ContactsByEmail.dxl | View | Yes |

Table 32. Metadata File Types

| Metadata File Name | Metadata File Type | Customizable |
|--|--------------------|--------------|
| SBL.View.ElementByID.dxl | View | Yes |
| SBL.View.ActionByAccount.dxl | View | Yes |
| SBL.View.ActionByOpportunity.dxl | View | Yes |
| SBL.View.AttachmentsByParent.dxl | View | Yes |
| SBL.View.NotesByParent.dxl | View | Yes |
| SBL.View.OpportunityByAccount.dxl | View | Yes |
| SBL.View.ProductsByParent.dxl | View | Yes |
| SBL.View.EmployeesByEmail.dxl | View | Yes |
| SBL.View.PicklistCurrency.dxl | View | Yes |
| SBL.View.PicklistOpportunitySalesStage.dxl | View | Yes |
| SBL.View.PicklistsAccount.dxl | View | Yes |
| SBL.View.PicklistsActivity.dxl | View | Yes |
| SBL.View.PicklistsAddress.dxl | View | Yes |
| SBL.View.PicklistsNote.dxl | View | Yes |
| SBL.View.PicklistsOpportunity.dxl | View | Yes |
| SBL.View.SalesbookAccounts.dxl | View | Yes |
| SBL.View.SalesbookAddress.dxl | View | Yes |
| SBL.View.SalesbookAssignmentGroup.dxl | View | Yes |
| SBL.View.SalesbookContacts.dxl | View | Yes |
| SBL.View.SalesbookEmployees.dxl | View | Yes |
| SBL.View.SalesbookIndustry.dxl | View | Yes |
| SBL.View.SalesbookInternalDivision.dxl | View | Yes |
| SBL.View.SalesbookOpportunities.dxl | View | Yes |
| SBL.View.SalesbookPositions.dxl | View | Yes |
| SBL.View.SalesbookProducts.dxl | View | Yes |
| SBL.View.SaleSearch.dxl | View | Yes |
| SBL.View.SettingsSyncOwner.dxl | View | Yes |
| SBL.View.Opportunities.dxl | View | Yes |

Table 32. Metadata File Types

| Metadata File Name | Metadata File Type | Customizable |
|--|--------------------|--------------|
| SBL.View.ActivityProcessingSrc.dxl | View | Yes |
| SBL.Folder.Account.dxl | Folder | Yes |
| SBL.Folder.AccountAddresses.dxl | Folder | Yes |
| SBL.Folder.AccountContactWithNew.dxl | Folder | Yes |
| SBL.Folder.Action.dxl | Folder | Yes |
| SBL.Folder.Address.dxl | Folder | Yes |
| SBL.Folder.Contact.dxl | Folder | Yes |
| SBL.Folder.Employee.dxl | Folder | Yes |
| SBL.Folder.Industry.dxl | Folder | Yes |
| SBL.Folder.Opportunity.dxl | Folder | Yes |
| SBL.Folder.OpportunityContactWithNew.dxl | Folder | Yes |
| SBL.Folder.Organization.dxl | Folder | Yes |
| SBL.Folder.Position.dxl | Folder | Yes |
| SBL.Folder.Territory.dxl | Folder | Yes |
| SBL.Folder.AccountsFiltered.dxl | Folder | Yes |
| SBL.Folder.AccountsSelected.dxl | Folder | Yes |
| SBL.Folder.AddressesFiltered.dxl | Folder | Yes |
| SBL.Folder.AddressesSelected.dxl | Folder | Yes |
| SBL.Folder.ContactsFiltered.dxl | Folder | Yes |
| SBL.Folder.ContactsSelected.dxl | Folder | Yes |
| SBL.Folder.EmployeesFiltered.dxl | Folder | Yes |
| SBL.Folder.EmployeesSelected.dxl | Folder | Yes |
| SBL.Folder.IndustriesFiltered.dxl | Folder | Yes |
| SBL.Folder.IndustriesSelected.dxl | Folder | Yes |
| SBL.Folder.OpportunitiesFiltered.dxl | Folder | Yes |
| SBL.Folder.OpportunitiesSelected.dxl | Folder | Yes |
| SBL.Folder.OrganizationsFiltered.dxl | Folder | Yes |
| SBL.Folder.OrganizationsSelected.dxl | Folder | Yes |

Table 32. Metadata File Types

| Metadata File Name | Metadata File Type | Customizable |
|--|--------------------|--------------|
| SBL.Folder.PositionsFiltered.dxl | Folder | Yes |
| SBL.Folder.PositionsSelected.dxl | Folder | Yes |
| SBL.Folder.ProductsFiltered.dxl | Folder | Yes |
| SBL.Folder.ProductsSelected.dxl | Folder | Yes |
| SBL.Folder.TerritoriesSelected.dxl | Folder | Yes |
| SD3.Subform.ForwardInvocation.dxl | Subform | No |
| SBL.Subform.Contact.dxl | Subform | Yes |
| SBL.Subform.CRMContact.version7.dxl | Subform | Yes |
| SBL.Subform.CRMContact.version8.dxl | Subform | Yes |
| SBL.Subform.ActivitySubform.dxl | Subform | Yes |
| SD3.Form.PickListDialog.dxl | Form | No |
| SD3.Form.PickListDialogEx.dxl | Form | No |
| SD3.Form.PickListDialogNarrow.dxl | Form | No |
| SD3.Form.StartSync.dxl | Form | No |
| SD3.Form.MVG.dxl | Form | No |
| SBL.Form.Account.dxl | Form | Yes |
| SBL.Form.Activity.dxl | Form | Yes |
| SBL.Form.Address.dxl | Form | Yes |
| SBL.Form.Note.dxl | Form | Yes |
| SBL.Form.Opportunity.dxl | Form | Yes |
| SBL.Form.Product.dxl | Form | Yes |
| SBL.Form.Attachment.dxl | Form | Yes |
| SBL.Form.ContactDialog.dxl | Form | Yes |
| SD3.Agent.NoteCreatedHandler.dxl | agent | No |
| SD3.Agent.NoteDeletingHandler.dxl | agent | No |
| SD3.Agent.NoteUpdatedHandler.dxl | agent | No |
| SD3.Agent.PreSynchronizeProcessor.dxl | agent | No |
| SD3.Agent.PostSynchronizeProcessor.dxl | agent | No |

Table 32. Metadata File Types

| Metadata File Name | Metadata File Type | Customizable |
|----------------------------|--------------------|--------------|
| SD3.Agent.CleanUp.dxl | agent | No |
| SD3.Agent.SyncNow.dxl | agent | No |
| SBL.Agent.ShowHelp.dxl | agent | Yes |
| SBL.Agent.ShareContact.dxl | agent | Yes |
| SBL.Agent.ShareObject.dxl | agent | Yes |
| SBL.Outline.ShareBar.dxl | outline | Yes |
| SD3.Resources.dxl | image resources | Yes |
| resources.dxl | image resources | Yes |
| toolbar_images.dxl | image resources | Yes |

Metadata File Types That Support Languages

If Siebel CRM Desktop must support a language other than English, then you must add the required language files.

Table 33 lists each of the metadata file types that you must add to the customization package. 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 Ln_package_res.de_DE file. For example, if you must support Japanese, then you must add the Ln_package_res.ja_JPfile.

Siebel CRM Desktop includes support for English, by default. You do not need to include a metadata file type for English.

Table 33. Metadata File Types That Support Languages

| Metadata Type | Metadata File Name |
|------------------------|--------------------------|
| LN Resource Bundle ENG | Ln_package_res.xml |
| LN Resource Bundle SA | Ln_package_res.ar_SA.xml |
| LN Resource Bundle CZ | Ln_package_res.cs_CZ.xml |
| LN Resource Bundle DK | Ln_package_res.da_DK.xml |
| LN Resource Bundle DE | Ln_package_res.de_DE.xml |
| LN Resource Bundle ES | Ln_package_res.es_ES.xml |
| LN Resource Bundle FI | Ln_package_res.fi_FI.xml |
| LN Resource Bundle FR | Ln_package_res.fr_FR.xml |
| LN Resource Bundle IL | Ln_package_res.he_IL.xml |
| LN Resource Bundle IT | Ln_package_res.it_IT.xml |
| LN Resource Bundle JP | Ln_package_res.ja_JP.xml |

Table 33. Metadata File Types That Support Languages

| Metadata Type | Metadata File Name |
|-----------------------|--------------------------|
| LN Resource Bundle KR | Ln_package_res.ko_KR.xml |
| LN Resource Bundle NL | Ln_package_res.nl_NL.xml |
| LN Resource Bundle PL | Ln_package_res.pl_PL.xml |
| LN Resource Bundle BR | Ln_package_res.pt_BR.xml |
| LN Resource Bundle PT | Ln_package_res.pt_PT.xml |
| LN Resource Bundle RU | Ln_package_res.ru_RU.xml |
| LN Resource Bundle SE | Ln_package_res.sv_SE.xml |
| LN Resource Bundle TH | Ln_package_res.th_TH.xml |
| LN Resource Bundle TR | Ln_package_res.tr_TR.xml |
| LN Resource Bundle CN | Ln_package_res.zh_CN.xml |
| LN Resource Bundle TW | Ln_package_res.zh_TW.xml |

Files in the Customization Package

To customize some Siebel CRM Desktop features, you can modify XML files and JavaScript files in the customization package. 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 82](#).

Files in the Customization Package

Table 34 describes the XML files that Siebel CRM Desktop includes in the customization package. For more information, see [“About the Customization Package” on page 33](#).

Table 34. Files in the Customization Package

| File Name | Description |
|--------------------------------|--|
| Ln_connector_configuration.xml | <p>This XML file provides the following capabilities:</p> <ul style="list-style-type: none">■ Defines objects that are synchronized■ Defines the criteria that CRM Desktop uses to detect duplicate objects in the Siebel database■ Defines the preset filters for a custom synchronization■ Defines the object types that CRM Desktop displays in the Filter Records tab of the Synchronization control panel.■ Defines presets for sliding date ranges. <p>For more information, see “Customizing Synchronization” on page 154.</p> |
| info.xml | <p>This XML file provides the following capabilities:</p> <ul style="list-style-type: none">■ Identifies the product name■ Identifies the package version and the CRM Desktop version■ Identifies the product versions that are compatible with the package■ Identifies the Siebel Server versions that are compatible with the package■ Identifies the general comments for the package <p>You can use it to track the package version. You can use the product name and version to check compatibility.</p> <p>For more information, see “How Siebel CRM Desktop Determines Compatibility” on page 75.</p> |
| Ln_package_res.xml | <p>Defines various resources for the customization package.</p> |

Table 34. Files in the Customization Package

| File Name | Description |
|-----------------------------|--|
| platform_configuration.xml | <p>Defines the configuration for the platform. For example, the platform_configuration.xml file defines how to do the following:</p> <ul style="list-style-type: none"> ■ Connect to the Internet ■ Share native IBM Notes items ■ Convert contacts ■ Control the synchronization intervals that display in the Synchronization tab ■ Control the data that CRM Desktop removes if the user removes CRM Desktop |
| Ln_siebel_basic_mapping.xml | <p>This XML file provides the following capabilities:</p> <ul style="list-style-type: none"> ■ Defines field mapping between CRM Desktop and IBM Notes ■ Defines field mapping between CRM Desktop and Siebel CRM ■ Describes objects to add to IBM Notes ■ Defines the form that CRM Desktop uses to display an object in IBM Notes ■ Defines a set of custom IBM Notes views that CRM Desktop applies for an object <p>For more information, see “Customizing Field Mapping” on page 154.</p> |
| siebel_meta_info.xml | <p>This XML file provides the following capabilities:</p> <ul style="list-style-type: none"> ■ Defines the object types that CRM Desktop supports ■ Defines fields and their types ■ Defines the XML element names that CRM Desktop uses to create a Siebel message <p>For more information, see “Customizing Meta Information” on page 157.</p> |

JavaScript Files in the Customization Package

[Table 35](#) describes the JavaScript files that Siebel CRM Desktop includes in the customization package.

Table 35. JavaScript Files in the Customization Package

| JavaScript File Name | Description |
|--------------------------|---|
| autoresolver.js | Defines functions to resolve conflicts. |
| autoresolve_helpers.js | An internal CRM Desktop file. You must not modify this file. |
| application_script.js | Defines entry points for scripts and to call scripts from other files. |
| business_logic.js | Defines logic for the following items: <ul style="list-style-type: none">■ Activities■ Mail processing■ The data model |
| data_model.js | Defines the data model and functions for objects. |
| recurrence_processing.js | Defines patterns for recurrence processing. For more information, see “How Siebel CRM Desktop Transforms Objects Between Siebel CRM Data and IBM Notes Data” on page 372. |

JavaScript Files That Siebel CRM Desktop Uses Internally

Siebel CRM Desktop uses the following files internally. You must not modify these files:

- autoresolve_helpers.js
- data_model.js

IBM Notes Field Types and Equivalent Convertor Classes

This topic includes example IBM Notes field types and their equivalent convertor classes. Siebel CRM Desktop defines IBM Notes field types in the Ln_siebel_basic_mapping.xml file.

Example of a String Field

The following code is an example of a string field:

```
<field id="First Name" ver="1">
  <reader>
    <lotus_std>
      <lotus_field id="FirstName"/>
      <convertor>
        <string/>
      </convertor>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
```

```

    <lotus_field id="FirstName"/>
    <convertor>
      <string/>
    </convertor>
  </lotus_std>
</writer>
</field>

```

Example of a Double Field

The following code is an example of a double field:

```

<field id="Primary Revenue Amount" ver="1">
  <reader>
    <lotus_std>
      <lotus_field id="Revenue"/>
      <convertor>
        <double/>
      </convertor>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="Revenue"/>
      <convertor>
        <double/>
      </convertor>
    </lotus_std>
  </writer>
</field>

```

Example of a Binary Hex String Field

The following code is an example of a Binary hex string field:

```

<field id="CurrentUser" ver="1">
  <reader>
    <lotus_std>
      <lotus_field id="CRMDSibel CurrentUser"/>
      <convertor>
        <binary_hexstring/>
      </convertor>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="CRMDSibel CurrentUser"/>
      <convertor>
        <binary_hexstring/>
      </convertor>
    </lotus_std>
  </writer>
</field>

```

Example of a Boolean Field

The following code is an example of a Boolean field:

```
<field id="PersonAccountEnabled" ver="1">
  <reader>
    <lotus_std>
      <lotus_field id="CRMDSiebel_PA_Enabled"/>
      <convertor>
        <string_boolean/>
      </convertor>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="CRMDSiebel_PA_Enabled"/>
      <convertor>
        <string_boolean/>
      </convertor>
    </lotus_std>
  </writer>
</field>
```

Example of a Datetime Field

The following code is an example of the datetime field:

```
<field id="Planned" ver="1">
  <reader>
    <lotus_std>
      <lotus_field id="StartDateTime"/>
      <convertor>
        <datetime/>
      </convertor>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="StartDateTime"/>
      <convertor>
        <datetime/>
      </convertor>
    </lotus_std>
  </writer>
</field>
```

Example of a Date Field

The following code is an example of a date field:

```
<field id="Primary Revenue Close Date" ver="1">
  <reader>
    <lotus_std>
      <lotus_field id="CloseDate"/>
```



```

        <converter>
        <date/>
    </converter>
</lotus_std>
</reader>
<writer>
    <lotus_std>
        <lotus_field id="CloseDate"/>
        <converter>
        <date/>
    </converter>
    </lotus_std>
</writer>
</field>

```

Example of a Number Field

The following code is an example of a number field:

```

<field id="SortOrder" ver="1">
    <reader>
        <lotus_std>
            <lotus_field id="SortOrder"/>
            <converter>
            <number/>
        </converter>
        </lotus_std>
    </reader>
    <writer>
        <lotus_std>
            <lotus_field id="SortOrder"/>
            <converter>
            <number/>
        </converter>
        </lotus_std>
    </writer>
</field>

```

Example of a Double String Field

The following code is an example of a double string field:

```

<field id="Primary Revenue Win Probability" ver="1">
    <reader>
        <lotus_std>
            <lotus_field id="Probability"/>
            <converter>
            <double_string/>
        </converter>
        </lotus_std>
    </reader>
    <writer>
        <lotus_std>

```

```
<lotus_field id="Probability"/>
<converter>
  <double_string/>
</converter>
</lotus_std>
</writer>
</field>
```

Example of an Integer Field

The following code is an example of an integer field:

```
<field id="ObjectState" ver="1">
  <reader>
    <lotus_std>
      <lotus_field id="ObjectState"/>
      <converter>
        <int_string/>
      </converter>
    </lotus_std>
  </reader>
  <writer>
    <lotus_std>
      <lotus_field id="ObjectState"/>
      <converter>
        <int_string/>
      </converter>
    </lotus_std>
  </writer>
  <writer>
    <lotus_std>
      <lotus_field id="ObjectStateString"/>
      <converter>
        <flags_mask="-I S-P-----V"/>
      </converter>
    </lotus_std>
  </writer>
</field>
```

B

How Siebel CRM Desktop Maps Fields Between Siebel CRM Data and IBM Notes Data

This appendix describes how Siebel CRM Desktop maps fields between Siebel CRM data and IBM Notes data. It includes the following topics:

- [How Siebel CRM Desktop Maps Fields Between Siebel Activities and IBM Notes Calendar on page 363](#)
- [How Siebel CRM Desktop Maps Fields Between Siebel Activities and IBM Notes To Do Items on page 366](#)
- [How Siebel CRM Desktop Maps Fields Between Siebel CRM Activities and IBM Notes Emails on page 371](#)
- [How Siebel CRM Desktop Transforms Objects Between Siebel CRM Data and IBM Notes Data on page 372](#)

How Siebel CRM Desktop Maps Fields Between Siebel Activities and IBM Notes Calendar

[Table 36](#) describes how Siebel CRM Desktop maps objects between a Siebel CRM activity and IBM Notes Calendar.

Table 36. How Siebel CRM Desktop Maps Fields Between Siebel Activities and IBM Notes Calendar

| Siebel Field | IBM Notes Field | Required | Description |
|--------------|---|----------|---|
| Account | Account association that the user specifies when the user links a CRM record. | No | Not applicable. |
| Comments | Description A private calendar entry does not include a description. | No | Not applicable. |
| Contacts | List of contacts that is received from the list of participants that remain after employees are removed from this list when CRM Desktop resolves the primary email of the contacts. | No | The logic that applies for the Employees field also applies for the Contacts field. |
| Created by | User ID of the employee who creates the activity. | Yes | Not applicable. |

Table 36. How Siebel CRM Desktop Maps Fields Between Siebel Activities and IBM Notes Calendar

| Siebel Field | IBM Notes Field | Required | Description |
|-----------------------------|--|----------|---|
| Created Date | Timestamp of when the activity is created. | Yes | Not applicable. |
| Description | For more information, see “How Siebel CRM Desktop Handles Private Activities” on page 365. | No | The value that is set for a private To Do item is defined in the customization package and is included in the localization resources. |
| Duration | Duration | Yes | Not applicable. |
| Employees | List of employees that is received from the list of participants when CRM Desktop resolves the primary email of the employees. | No | CRM Desktop does the preliminary resolution when the user saves a calendar entry. To improve the quality of this list, the Siebel Server analyzes the processing that occurs on the server. It does this work after the next synchronization. |
| ExceptionsList | No direct mapping to the IBM Notes field exists. | Yes | For more information, see “How Siebel CRM Desktop Handles a Repeating Calendar Entry” on page 55. |
| Meeting Location | Location | No | Not applicable. |
| Opportunity | Opportunity association that the user specifies when the user links a CRM record. | No | Not applicable. |
| Owner | For more information, see “How Siebel CRM Assigns Meeting Organizers” on page 44. | Yes | Not applicable. |
| PIM Meeting Participants | List of participants for the calendar entry, delimited by a semicolon (;). | No | Not applicable. |
| Planned | Start Date/Time | Yes | Not applicable. |
| Planned Completion/End time | End Date/Time | Yes | Not applicable. |

How Siebel CRM Desktop Maps Fields Between Siebel CRM Data and IBM Notes Data

■ How Siebel CRM Desktop Maps Fields Between Siebel Activities and IBM Notes Calendar

Table 36. How Siebel CRM Desktop Maps Fields Between Siebel Activities and IBM Notes Calendar

| Siebel Field | IBM Notes Field | Required | Description |
|--|--|----------|---|
| Priority Values are: ■ 1-ASAP ■ 2-High ■ 3-Medium ■ 4-Low | IBM Notes 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 366. |
| Private | Private | No | Not applicable. |
| Repeat | Recurring flag | No | Not applicable. |
| Repeat Frequency | CRM Desktop uses a binary field that combines the data from a number of IBM Notes fields. The frequency is daily, weekly, monthly, or yearly. | No | Not applicable. |
| Repeat Until | Recurrence Range End | No | Not applicable. |
| Type | Type that the user specifies when linking the CRM record. The default value is Calendar Entry. | Yes | The user can choose the activity type in the form of the IBM Notes calendar entry only for an activity that contains the following <i>Display in</i> value: Calendar and Activities The customization package defines the default value. You can change this value. |

How Siebel CRM Desktop Handles Private Activities

Siebel CRM Desktop handles a private activity in the following ways:

- If an activity in Siebel CRM is marked private, then it does not synchronize this activity from the Siebel Server to IBM Notes. The master filters in the customization package restrict it from synchronizing a private activity.
- If the user creates a private activity in IBM Notes and then attempts to share it with Siebel CRM, then validation disallows this attempt. For example, assume a user creates a shared activity in a Siebel CRM calendar entry. If the user attempts to mark this calendar entry as private, then CRM Desktop displays a warning message that it cannot share a private IBM Notes item with Siebel CRM.

How Siebel CRM Desktop Maps the Priority Field

Siebel CRM data includes an ASAP priority value but IBM Notes data does not. It is not possible to save the ASAP value during synchronization, so Siebel CRM Desktop does not include it in the mapping. If a Siebel task is marked 1-ASAP, then CRM Desktop creates a mapping error.

The IBM Notes activity includes another field to store the Priority value that CRM Desktop derives from the Siebel CRM data. Note the following behavior:

- If the IBM Notes item is updated, and if the Priority of the IBM Notes item is High, then CRM Desktop validates the stored value.
- If the value that is stored for the activity is ASAP, then CRM Desktop uses the ASAP value from the activity.
- If the value that is stored for the activity is not ASAP, then CRM Desktop uses the value from the IBM Notes record.

How Siebel CRM Desktop Maps Fields Between Siebel Activities and IBM Notes To Do Items

Table 37 describes how CRM Desktop maps fields between a Siebel CRM activity and the IBM Notes To Do item.

Table 37. How Siebel CRM Desktop Maps Fields Between Siebel Activities and IBM Notes To Do Items

| Siebel Field | IBM Notes Field | Required | Description |
|--------------|---|----------|---|
| Owner | Owner | Yes | For more information, see “How Siebel CRM Desktop Maps the Owner Field Between Siebel CRM Activities and IBM Notes To Do Items” on page 368 . |
| 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 To Do item, then CRM Desktop creates an activity in IBM Notes with a <i>Display in</i> value that includes the following: To Do and Activities The customization package defines the default value. You can change this value. |
| Description | For more information, see “How Siebel CRM Desktop Handles Private Activities” on page 365 . | No | The customization package defines the value that is set for a private To Do item. Localization resources include this value. |

Table 37. How Siebel CRM Desktop Maps Fields Between Siebel Activities and IBM Notes To Do Items

| Siebel Field | IBM Notes Field | Required | Description |
|--|--|----------|--|
| Priority Values are: ■ 1-ASAP ■ 2-High ■ 3-Medium ■ 4-Low | IBM Notes 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 366. |
| Comments | Description A private To Do item does not include a description. | No | If the Private check box is set to allow a shared To Do item, then 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 To Do item. 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 369. |
| Status | Status | No | For more information, see “How Siebel CRM Desktop Maps the Status Field of an Activity” on page 369. |
| Account | Account association that the user specifies when the user links a CRM record. | No | Not applicable. |

Table 37. How Siebel CRM Desktop Maps Fields Between Siebel Activities and IBM Notes To Do Items

| Siebel Field | IBM Notes Field | Required | Description |
|--------------|---|----------|-----------------|
| 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 Siebel CRM Activities and IBM Notes To Do Items

This topic describes how Siebel CRM Desktop maps the owner field between a Siebel CRM activity and the IBM Notes To Do item. CRM Desktop sets the owner differently for each of the following situations:

- A shared To Do item is assigned to another employee. It sets the assignee as the owner of the activity.
- A shared To Do item is assigned to a number of employees. It sets each assignee as the owner of their own activity.
- A shared To Do item is assigned to a shared contact or to an external person. It sets the employee who created the To Do item as the owner of the activity.
- A To Do item is created by an external person and assigned to an employee and this employee shares the To Do item. It sets the employee as the owner of the activity.
- An external person who is not a Siebel employee sends a To Do item to a number of employees who are CRM Desktop users. It sets each employee as the owner of the activity and adds the remaining employees to the employee team. It does this work after each employee accepts and shares the To Do item.

How Siebel CRM Desktop Maps the Status Field of an Activity

[Table 38](#) describes how Siebel CRM Desktop maps the status of a Siebel CRM activity to the status of the IBM Notes To Do item when the user shares a To Do item.

Table 38. How Siebel CRM Desktop Maps the Status Field of an Activity

| Siebel Status | IBM Notes Status |
|---------------|-------------------------|
| Not Started | Not Started |
| In Progress | In Progress |
| Done | Completed |
| On Hold | Waiting on someone else |
| Canceled | Deferred |

[Table 39](#) describes how 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 value:

To Do and Activities

CRM Desktop does this work during synchronization.

Table 39. How Siebel CRM Desktop Maps the Status of a Siebel CRM Activity When To Do Contains *To Do and Activities*

| Siebel Status | IBM Notes 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 40 on page 370 determines the value for the status. |

Table 40 describes how CRM Desktop uses the Percent Complete value in the IBM Notes To Do item to determine the value of the Status field.

Table 40. 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 IBM Notes To Do Item |
|---------------|--|
| 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. The following sequence occurs:

- 1 A user creates a shared To Do item in IBM Notes that includes a status that is In Progress and a Percent Complete that is 0.
- 2 CRM Desktop creates an activity on the Siebel Server that includes a status of In Progress.
- 3 The user synchronizes with the Siebel Server.
- 4 On the Siebel Server, CRM Desktop changes the status to Requested.
- 5 The user synchronizes with the Siebel Server again.
- 6 In IBM Notes, CRM Desktop updates the activity status to Requested.
- 7 At this point, the Percent Complete field of the To Do item is 0. CRM Desktop updates the To Do item status to Not Started.

How Siebel CRM Desktop Maps Fields Between Siebel CRM Activities and IBM Notes Emails

Table 41 describes how Siebel CRM Desktop maps fields between a Siebel CRM activity and the IBM Notes email.

Table 41. How Siebel CRM Desktop Maps Fields Between a Siebel CRM Activity and the IBM Notes Email

| Siebel CRM Field | Required | IBM Notes 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 |
| Display | Yes | Changes to Communication and Activities. This value is the default value. |
| Email Attachment Flag | No | Changes to Y. This value is the default value. |
| Email BCC Line | No | Email BCC Line |
| Email Body | No | Email Body |
| Email CC Line | No | Email CC Line |
| Email Sender Address | No | 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@example.com> In this example, CRM Desktop resolves the Email Sender Address to the following address: jane.smith@example.com |
| Email Sender Name | No | 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. |

Table 41. How Siebel CRM Desktop Maps Fields Between a Siebel CRM Activity and the IBM Notes Email

| Siebel CRM Field | Required | IBM Notes Field |
|------------------|----------|--|
| Priority | No | Importance. CRM Desktop applies the following mapping: <ul style="list-style-type: none">■ 1-ASAP in Siebel CRM data is ASAP in IBM Notes■ 2-High in Siebel CRM data is High in IBM Notes■ 3-Medium in Siebel CRM data is Normal in IBM Notes■ 4-Low in Siebel CRM data is Low in IBM Notes For more information, see “How Siebel CRM Desktop Maps the Priority Field” on page 366 . |
| 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 CRM Desktop saves as an attachment to the activity. |

How Siebel CRM Desktop Transforms Objects Between Siebel CRM Data and IBM Notes Data

This topic describes how Siebel CRM Desktop transforms objects between Siebel CRM Data and IBM Notes data. It includes the following topics:

- [How Siebel CRM Desktop Transforms a Calendar Entry That Does Not Repeat on page 373](#)
- [How Siebel CRM Desktop Transforms a Repeating Calendar Entry That Matches a Siebel Repeating Pattern on page 374](#)
- [How Siebel CRM Desktop Transforms a Repeating Calendar Entry That Does Not Match Siebel Repeating Patterns on page 375](#)
- [How Siebel CRM Desktop Transforms Siebel CRM Activities That Do Not Repeat on page 376](#)
- [How Siebel CRM Desktop Transforms Siebel CRM Activities That Repeat on page 377](#)
- [How Siebel CRM Desktop Maps Fields Between a Siebel Calendar Entry and a IBM Notes Calendar Entry on page 379](#)

If the user synchronizes a repeating activity between the Siebel Server and **IBM Notes**, then **CRM Desktop** uses the following logic:

- 1 Maps changes between Siebel activities in IBM Notes and native IBM Notes items.
CRM Desktop updates changes in the IBM Notes Calendar entry in the activity that is linked to the Calendar entry.
- 2 Synchronizes changes between Siebel activities in IBM Notes and Siebel CRM activities.
CRM Desktop updates this change in Siebel CRM data after the synchronization finishes.

How Siebel CRM Desktop Transforms a Calendar Entry That Does Not Repeat

Table 42 describes how Siebel CRM Desktop transforms a Calendar entry in IBM Notes that does not repeating.

Table 42. How Siebel CRM Desktop Transforms a Calendar Entry That Does Not Recur

| Action in IBM Notes | Work That Siebel CRM Desktop Performs |
|--|--|
| The user creates an activity that does not repeat in IBM Notes. | Creates a corresponding Siebel CRM activity. |
| The user changes an activity that does not repeat in IBM Notes. | Changes the corresponding Siebel CRM activity. |
| The user deletes an activity that does not repeat in IBM Notes. | Deletes the corresponding Siebel CRM activity. |
| The user changes an activity that does not repeat in IBM Notes to a repeating Calendar entry that matches the Siebel repeating pattern. | Changes the corresponding Siebel CRM activity. |
| The user changes an activity that does not repeat in IBM Notes to a repeating Calendar entry that does not match the Siebel repeating pattern. | CRM Desktop does the following work: <ul style="list-style-type: none">■ Changes the corresponding activity. It uses the Siebel repeating pattern that contains the longest interval between occurrences and that can incorporate all occurrences of the chosen IBM Notes pattern.■ To match the calendars in Siebel CRM and in IBM Notes, it identifies the list of exceptions for the activity. |

How Siebel CRM Desktop Transforms a Repeating Calendar Entry That Matches a Siebel Repeating Pattern

Table 43 describes how Siebel CRM Desktop transforms a repeating Calendar entry that matches a Siebel CRM repeating pattern.

Table 43. How Siebel CRM Desktop Transforms a Repeating Calendar Entry That Matches a Siebel CRM Repeating Pattern

| Action in IBM Notes | Work That Siebel CRM Desktop Performs |
|--|---|
| The user creates a repeating Calendar entry that matches a Siebel repeating pattern. | Creates a corresponding Siebel CRM repeating activity. |
| The user changes a single occurrence for a repeating Calendar entry that matches a Siebel repeating pattern. | CRM Desktop does the following work in IBM Notes: <ul style="list-style-type: none">■ Adds the date of the exception that changed to the exceptions list for the target activity.■ Adds the new, nonrepeating activity to the newly created calendar entry in IBM Notes. |
| The user changes a repeating Calendar entry that matches a Siebel repeating pattern. | Changes the corresponding Siebel CRM repeating activity. |
| The user deletes a single occurrence for a repeating Calendar entry that matches a Siebel repeating pattern. | CRM Desktop does the following work in IBM Notes: <ul style="list-style-type: none">■ 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 repeating Calendar entry that matches a Siebel repeating pattern into the Calendar entry that does not repeat. | Changes the corresponding Siebel CRM activity to an activity that does not repeat. |
| The user changes a repeating Calendar entry that matches a Siebel repeating pattern into a repeating Calendar entry that does not match any Siebel repeating patterns. | CRM Desktop does the following work in IBM Notes: <ul style="list-style-type: none">■ Changes the corresponding Siebel CRM activity. It uses the Siebel repeating pattern that contains the longest interval between occurrences and that can incorporate all occurrences of the chosen IBM Notes pattern.■ To match IBM Notes and Siebel calendars, it identifies the list of exceptions for this activity. |

How Siebel CRM Desktop Transforms a Repeating Calendar Entry That Does Not Match Siebel Repeating Patterns

Table 44 describes how Siebel CRM Desktop transforms a repeating Calendar entry that does not match a Siebel repeating pattern.

Table 44. How Siebel CRM Desktop Transforms a Repeating Calendar Entry That Does Not Match a Siebel Repeating Pattern

| Action in IBM Notes | Work That Siebel CRM Desktop Performs |
|---|---|
| The user creates a repeating Calendar entry that does not match a Siebel repeating pattern. | <p>CRM Desktop does the following work in IBM Notes:</p> <ul style="list-style-type: none">■ Creates the corresponding activity. To identify this activity, it uses the Siebel repeating pattern that contains the longest interval between occurrences and that can incorporate all occurrences of the chosen IBM Notes pattern.■ To match the calendars in IBM Notes and CRM Desktop, it creates the list of exceptions for this activity. |
| The user changes a single occurrence for a repeating Calendar entry that does not match any Siebel repeating pattern. | <p>CRM Desktop does the following work in the IBM Notes Calendar:</p> <ul style="list-style-type: none">■ Creates a new calendar entry that is nonrepeating.■ Deletes the occurrence that changed for the IBM Notes calendar entry.■ Creates an exception in IBM Notes for the occurrence date that changed. <p>CRM Desktop does the following work in IBM Notes:</p> <ul style="list-style-type: none">■ Adds an exception to the exceptions dates list of the activity.■ Adds a new activity that does not repeat. |

Table 44. How Siebel CRM Desktop Transforms a Repeating Calendar Entry That Does Not Match a Siebel Repeating Pattern

| Action in IBM Notes | Work That Siebel CRM Desktop Performs |
|---|---|
| The user deletes a single occurrence for a repeating Calendar entry that does not match a Siebel repeating pattern. | CRM Desktop does the following work in IBM Notes: <ul style="list-style-type: none">■ Adds an exception to the exceptions dates list of the activity.■ Deletes this single occurrence of the activity. |
| The user deletes a repeating Calendar entry that does not match a Siebel repeating pattern. | Deletes the corresponding repeating activity. |

How Siebel CRM Desktop Transforms Siebel CRM Activities That Do Not Repeat

Table 45 describes how Siebel CRM Desktop transforms a Siebel CRM activity that is not repeated.

Table 45. How Siebel CRM Desktop Transforms a Siebel CRM Activity That Is Not Repeated

| Work That Occurs in Siebel CRM | Work That Siebel CRM Desktop Performs |
|--|--|
| The user creates a Siebel CRM activity that is not repeated. | Creates the corresponding IBM Notes activity that does not repeat. |
| The user changes a Siebel CRM activity that is not repeated. | Changes the corresponding IBM Notes activity that does not repeat. |
| 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 IBM Notes activity that does not repeat. |

How Siebel CRM Desktop Transforms Siebel CRM Activities That Repeat

Table 46 describes how Siebel CRM Desktop transforms an activity in Siebel CRM that is repeated.

Table 46. How Siebel CRM Desktop Transforms a Repeated Activity in Siebel CRM

| Work That Occurs in Siebel CRM | Work That Siebel CRM Desktop Performs |
|--|--|
| The user creates a repeating activity in Siebel CRM. | Creates a corresponding repeating Calendar entry in IBM Notes. |

Table 46. How Siebel CRM Desktop Transforms a Repeated Activity in Siebel CRM

| Work That Occurs in Siebel CRM | Work That Siebel CRM Desktop Performs |
|---|--|
| <p>The user changes a single occurrence of the repeating activity in Siebel CRM:</p> <ul style="list-style-type: none"> ■ Creates a delete activity. ■ Creates a new activity that does not repeat. | <p>Deletes the occurrence of the IBM Notes Calendar entry for the date of the occurrence that is deleted in Siebel CRM.</p> |
| <p>The user can modify a repeating activity in Siebel CRM. The activity was created in IBM Notes from a repeating Calendar entry, so the activity can originate in Siebel CRM or CRM Desktop can synchronize it to Siebel CRM from IBM Notes.</p> <p>If the user changes all instances of the repeating activity in Siebel CRM, then CRM Desktop deletes every instance of the repeating activity from the current day forward. Any instances that exist before the current day remain on the calendar.</p> <p>If the user deletes only one instance, then Siebel CRM still schedules every other instance.</p> <p>If the user changes a repeating Siebel CRM activity, then CRM Desktop does the following work:</p> <ul style="list-style-type: none"> ■ Changes the following date of the initial activity to the date minus one occurrence: repeat until ■ Creates a new repeated Calendar entry 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 Calendar entry: repeat until ■ Relinks the delete records to the corresponding repeating activities, depending on the exception date. | <p>CRM Desktop does the following work:</p> <ul style="list-style-type: none"> ■ Changes the end date of the initial repeating activity ■ Creates a new repeating activity. The Siebel CRM activity determines this new repeating activity. ■ Changes the links for the delete activities depending on the changes that CRM Desktop synchronizes from the Siebel Server |

How Siebel CRM Desktop Maps Fields Between a Siebel Calendar Entry and a IBM Notes Calendar Entry

Table 47 describes how Siebel CRM Desktop maps some fields between a repeating Siebel calendar entry and a repeating IBM Notes calendar entry.

Table 47. How Siebel CRM Desktop Maps Fields Between a Siebel Calendar Entry and a IBM Notes Calendar Entry

| Siebel CRM | | IBM Notes | | |
|------------|----------------------------|-----------|---|---|
| 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 |

This appendix describes the code in the XML files that Siebel CRM Desktop includes in the customization package. It includes the following topics:

- [Getting Information About Tags of the Metadata Files on page 381](#)
- [XML Code That Maps a Field on page 381](#)
- [XML Code That Customizes Platform Configuration on page 384](#)
- [XML Code That Customizes Synchronization on page 384](#)

Getting Information About Tags of the Metadata Files

The metadata files that Siebel CRM Desktop uses includes a number of tags that you can modify. XSD files include documentation for many of these tags. To get a copy of these files and to view the documentation that they contain, see Article ID 1502099.1 on My Oracle Support.

XML Code That Maps a Field

This topic describes the code of the Ln_siebel_basic_mapping.xml file. It includes the following topics:

- [Example Code of the Siebel Basic Mapping File on page 382](#)
- [Type Tag of the Siebel Basic Mapping File on page 383](#)
- [Field Tag of the Siebel Basic Mapping File on page 383](#)
- [Writer Tag of the Siebel Basic Mapping File on page 383](#)

For more information, see [“Customizing Field Mapping” on page 154](#).

Example Code of the Siebel Basic Mapping File

The following code is an example of the Ln_siebel_basic_mapping.xml file:

```
<root>
  <types>
    <type id="Contact" form="SBLPerson">
      <post_sync_processor>
        <contact_post_processor/>
      </post_sync_processor>
      <field id="First Name" ver="1">
        <reader>
          <lotus_std>
            <lotus_field id="FirstName"/>
            <convertor>
              <string/>
            </convertor>
          </lotus_std>
        </reader>
        <writer>
          <lotus_std>
            <lotus_field id="FirstName"/>
            <convertor>
              <string/>
            </convertor>
          </lotus_std>
        </writer>
      </field>
      <field id="Last Name" ver="1">
        <reader>
          <lotus_std>
            <lotus_field id="LastName"/>
            <convertor>
              <string/>
            </convertor>
          </lotus_std>
        </reader>
        <writer>
          <lotus_std>
            <lotus_field id="LastName"/>
            <convertor>
              <string/>
            </convertor>
          </lotus_std>
        </writer>
      </field>
    </type>
  </types>
</root>
```

Type Tag of the Siebel Basic Mapping File

The type tag defines the Siebel CRM object that Siebel CRM Desktop maps to IBM Notes.

CRM Desktop includes the following attributes in the type tag:

- **id.** Defines the ID or name of the Siebel CRM object that CRM Desktop maps to IBM Notes.
- **form.** Defines the ID of the form that CRM Desktop uses to display the object that the type tag defines.

Field Tag of the Siebel Basic Mapping File

The field tag describes the field mapping. It describes one field, so the number of field tags must be the same as the number of fields that are mapped.

The field tag includes the following attributes:

- **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.dxl file.
- **ver.** Used during development. CRM Desktop does not apply any change to the field description until the value of the ver attribute increases.

Writer Tag of the Siebel Basic Mapping File

The writer tag defines write access to the field that Siebel CRM Desktop maps to IBM Notes. It includes only the class attribute. The following values are available for the class attribute:

- lotus_std
- rich_string
- attached_file_writer
- attached_filename_writer
- multiwriter

The writer tag can include the following tags:

- lotus_field
- convertor
- linked_fields

XML Code That Customizes Platform Configuration

This topic describes the XML code that you can use to customize the platform configuration. The following code is an example of the platform_configuration.xml file. For more information, see [“Files in the Customization Package” on page 356](#):

```
<platform>
  <initialization_script>
    <![CDATA[
      application.settings.set("ProgressFormAutoshow", 0);
    ]]>
  </initialization_script>
</platform>
```

XML Code That Customizes Synchronization

This topic describes the code of the Ln_connector_configuration.xml file. It includes the following topics:

- [Example Code of the Connector Configuration File on page 385](#)
- [Types Tag of the Connector Configuration File on page 385](#)
- [Type Tag of the Connector Configuration File on page 385](#)
- [View Tag of the Connector Configuration File on page 386](#)
- [Synchronizer Tag of the Connector Configuration File on page 386](#)
- [Links Tag of the Connector Configuration File on page 386](#)
- [Natural Key Tag of the Connector Configuration File on page 387](#)
- [Filter Presets Tag of the Connector Configuration File on page 388](#)

For more information, see [“Customizing Synchronization” on page 154](#).

Example Code of the Connector Configuration File

The following code is an example of the Ln_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 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 Ln_siebel_basic_mapping.xml file.

Type Tag of the Connector Configuration File

The type tag describes the type to synchronize. You must describe it in the Ln_siebel_basic_mapping.xml file. It includes the id attribute that defines the ID of the object.

The type tag must contain the following tags:

- view
- synchronizer

View Tag of the Connector Configuration File

The view tag defines the type in the user interface. It defines the Filter Records tab on the Synchronization Control Panel dialog box.

The view tag includes the following attributes:

- **label**. Label that displays this object in the Synchronization Control Panel dialog box if CRM Desktop cannot resolve the name of this object.
- **label_plural**. Label that displays 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 CRM Desktop uses for this object on the Synchronization Control Panel dialog box.
- **normal_icon**. Defines the icon that displays next to the object type in the Filter Records tab of the Synchronization Control Panel dialog box.
- **large_icon**. Defines the icon that displays in the CRM Desktop Synchronization dialog box while CRM Desktop synchronizes this type of object.
- **suppress_sync_ui**. If `suppress_sync_ui` is true, then this attribute hides objects of this type from the Filter Records tab of the Synchronization Control Panel dialog box. If `suppress_sync_ui` is not defined, then 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 127](#).

Synchronizer Tag of the Connector Configuration File

The synchronizer tag describes attributes that the Synchronization Engine requires. For example, it describes relationships between objects or criteria to identify duplicate objects.

The synchronizer tag includes the `name_format` attribute that 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 CRM Desktop add-in.

The synchronizer tag can contain the following tags:

- `links`
- `natural_keys`

Links Tag of the Connector Configuration File

The links tag describes the references between types. You must specify 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 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 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 includes the following items:

- A set of natural_key tags that describe the criteria. Siebel CRM Desktop uses OR logic for all criteria that the natural_key tag describes.
- A set of field tags. Each of these tags includes a field name that CRM Desktop examines to identify duplicates. 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, if one of the following situations is true, then CRM Desktop detects two objects as duplicates:

- First Name AND Last Name contain the same values
- Email Address fields contain the same values

Filter Presets Tag of the Connector Configuration File

The filter_presets tag contains predefined filter criteria. The preset tag describes this criteria. The preset tag includes the name attribute. It defines the name for this criteria. The preset tag contains a set of type tags that specify filter criteria for each type.

The type tag defines the object type. Siebel CRM Desktop applies the filter criteria to this object type. You must specify 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 Ln_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 That Sets 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 [“Controlling the Size and Type of Synchronized Records” on page 135](#). The following code is an example usage of the group tag of the Ln_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>
  </group>
</group>
```

```

<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>

```


Glossary

access control

The set of Siebel CRM mechanisms that control the records that the user can access and the operations that 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

Work that a user must track. Examples include a to-do, email sent to a contact, or a calendar entry 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 calendar entry 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.

calendar entry (IBM Notes)

A record in the IBM Notes calendar or Siebel Web application calendar that reserves time to do something, such as a calendar entry to schedule a meeting with a customer or to reserve time to complete work in a given time frame.

attendee (IBM Notes)

A person included in the calendar entry, 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 on.

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 on.

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 where you install the CRM Desktop add-in.

consumer

In Siebel CRM, a consumer is a person with a party of usage type Customer. In IBM Notes, 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 IBM Notes, 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 email, telephone, and fax.

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.

CRM Desktop add-in

The technology for Siebel CRM Desktop that resides on the client computer that is provided in the form of a IBM Notes add-in. The IBM Notes add-in performs important work, including storing and displaying Siebel CRM data in native IBM Notes, and synchronizing PIM and nonPIM data with the Siebel Server.

See also ["IBM Notes add-in" on page 396](#).

current view

The IBM Notes view that displays content from the IBM Notes 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 IBM Notes, 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 modifying Siebel CRM Desktop to meet the specific requirements of your organization.

customization package

A logical collection of metadata files that is 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.

direct link

A type of link that possesses a one-to-one relationship between one object type and another object type. A link between one account and one opportunity is an example of a direct link.

Dynamic HTML (DHTML)

See [DHTML](#).

encryption

The method of encoding data for security purposes.

form

A generic concept that IBM Notes 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 IBM Notes that displays a collection of information from IBM Notes 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 that this communication uses and the business objects that they reference. 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 IBM Notes. 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 IBM Notes that allows the user to view records in a list, and then choose one or more records to associate with the current item. To identify the subset of data that the user can choose, a lookup control typically includes the capability to specify a search condition.

meeting

A calendar entry in IBM Notes that includes at least one participant.

metadata files

XML files that hold information on how the user experience must be shaped. The CRM Desktop add-in uses metadata files to do 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 that the user can use with Siebel CRM Desktop but where the user cannot access the Siebel Server. When in offline mode, CRM Desktop uses data in the local data to perform operations. Synchronization is delayed until the user is online.

online

A mode that the user can use with Siebel CRM Desktop while connected to the Siebel Server. The user can synchronize data with the Siebel Server at regular intervals or when the user performs an update. Similar to offline mode, when in online mode 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 particular size of deal or with a specific sales stage, and partner organizations that can help close the deal.

organizer

In IBM Notes, the person who created the calendar entry.

IBM Notes data

Data that is created in the native IBM Notes application.

IBM Notes folder

A folder in IBM Notes 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, the IBM Notes folder might also contain Siebel CRM data.

IBM Notes object

An entity that is native to IBM Notes. Examples of IBM Notes objects include an email, calendar entry, contact, and so on.

IBM Notes add-in

A program that performs important work, including storing and displaying Siebel CRM data in native IBM Notes and synchronizing PIM and nonPIM data with the Siebel Server.

See also [PIM](#); [Siebel Server](#).

IBM Notes portlet

A portlet that uses data in a IBM Notes folder that includes a custom view filter. The IBM Notes portlet includes ActiveX characteristics.

IBM Notes standard view

A default IBM Notes view that exists without Siebel CRM Desktop. The IBM Notes view provides different ways of viewing the same information in a folder by placing the information in different arrangements and formats.

mvg link

A type of link that possesses a one-to-many relationship between one object type and another object type. A link between one opportunity and many contacts is an example of an MVG link.

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 IBM Notes, 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 on. 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 IBM Notes.

PIM

Personal Information Manager. An application that typically helps a user to manage a list of contacts, calendar entries, email, and so on. IBM Notes, 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 IBM Notes that relates to a contact, calendar entry, and so on.

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 user position determines the records that the user can view and the operations that the user can perform on the records in a given IBM Notes 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 IBM Notes, also known as the IBM Notes record.

responsibility

An entity in the Siebel data model that determines the views that the user can access in IBM Notes. For example, the responsibility of the sales representative allows the user to access the My Opportunities view, whereas the responsibility of the Siebel CRM developer allows the user to access administration views. A Siebel CRM 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 IBM Notes 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 that the user can choose and that are appropriate for the context that the user uses to access data.

Siebel Business Application

An application that is part of Siebel CRM, such as Siebel Call Center.

Siebel CRM data

Business data that is created in the CRM Desktop add-in, data that is created in the client of a Siebel Business 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 IBM Notes 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 IBM Notes.

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 IBM Notes

The native IBM Notes application without the CRM Desktop add-in.

synchronization

A process that exchanges transactions between Oracle's Siebel CRM Desktop for IBM Notes and IBM Notes. This synchronization makes sure that CRM data is the same on the Siebel Server and in IBM Notes.

synchronization filter

Criteria that are considered during data synchronization so that some records are included and other records are excluded from processing during synchronization.

To Do item (IBM Notes)

A part of a set of actions that accomplish a job, solve a problem or completes an assignment. In the native IBM Notes application, a To Do item is a collection of simple business objects on the user level. A To Do item 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 IBM Notes user, a task is the view representation of a logical unit of work that the user must perform. The task is presented in Siebel CRM 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.

UniversalId

An attribute on a calendar entry record in IBM Notes that the user can use to correlate a shared calendar entry between meeting attendees. Meeting attendees in IBM Notes include their own copy of the calendar entry, but all copies include the same value for the UniversalId.

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 use components, Internet standards, and protocols, such as HTTP, XML code, and SOAP.

See also [SOAP](#).

Glossary ■ Self-contained, modular applications that can be described, published, located, and called over a network. Web services perform encapsulated business functions, ranging

Index

A

activities

- how it is handled when it is recurrent 58
- how origin affects handling 46

administration

- customization package 78
- metadata 103
- server variables 80

appointments

- field mapping 379
- handling of 50
- how handled when CRM Desktop is removed 114
- invitee list handling for 57
- repeated 55

attachments

- field mapping 372
- handling of 45
- usage with a customization package 35

B

back up

- manual export 67, 113

C

calendar items

- creation of 41
- field mapping 363
- how handled when saved or changed 49

Client

- installation file 81
- role in architecture 21

components

- architecture 27
- CRM Desktop 22, 210
- synchronization 27

connector_configuration.xml

- about 154
- example code 384

CRM Desktop

- architecture diagram, basic 22, 71
- architecture diagram, synchronization 27
- role in architecture 22, 71

CRM Desktop add

- removing for multiple users 114

CRM Desktop add-in

- installing 84

- installing for a single user 87

- item handling when removed 62

- network and infrastructure requirements 85

- removing for a single user 114

customization package

- about 28, 33, 355
- administering 35, 78
- affect of changing 69
- creating and publishing 78
- error handling 76
- items included during initial downloaded 64
- registering and obtaining 90
- relation with responsibilities 34
- relations with other customization objects 32
- role in the architecture 24
- unpublishing 80
- XML code 381
- XML files included in 356

E

Email

- how an email is handled 59

email

- address processing 57
- field mapping 371

error handling

- during synchronization 76
- tracing and logging 121

F

field mapping

- appointments 379
- attachments 372
- recurrent activity 372
- Siebel CRM Activity and client email 371
- Siebel CRM activity and the client calendar 363

First Run Assistant

- about 64

forms.xml

- about 155

I

installation

- CRM Desktop add-in 87

installation file

usage with client 81

M**metadata files**

administering 78
 connector_configuration.xml 154, 356, 384
 customization of 153, 209, 269
 definition of 35
 downloading of 68
 forms_xx.xml 155
 list of metadata files 355
 relations with customization package 34
 role in customization package 28
 siebel_basic_mapping.xml 154, 196, 204, 357, 381
 siebel_meta_info.xml 157, 357
 uploading 78

P**parameters**

options for setting client installation 98

R

repeated appointments 55

responsibilities

relations with other customization objects 32

S**Siebel Server**

administering 80

siebel_basic_mapping.xml

about 154
 code description 381
 using to create a user interface 204
 using to define a custom object 196

siebel_meta_info.xml

about 157

synchronization

appointment handling 50
 attachment handling 45
 customizing 154
 defining for a custom object 202
 engine in logical architecture 24
 error handling during 76
 of metadata during initial download 64
 of metadata during subsequent
 downloads 68
 XML code for 384