Siebel Object Interfaces Reference

Version 7.5.3, Rev. A

12-FRX5OU

July 2003

Siebel Systems, Inc., 2207 Bridgepointe Parkway, San Mateo, CA 94404
Copyright © 2003 Siebel Systems, Inc.
All rights reserved.
Printed in the United States of America

No part of this publication may be stored in a retrieval system, transmitted, or reproduced in any way, including but not limited to photocopy, photographic, magnetic, or other record, without the prior agreement and written permission of Siebel Systems, Inc.

Siebel, the Siebel logo, TrickleSync, TSQ, Universal Agent, and other Siebel product names referenced herein are trademarks of Siebel Systems, Inc., and may be registered in certain jurisdictions.

Other product names, designations, logos, and symbols may be trademarks or registered trademarks of their respective owners.

U.S. GOVERNMENT RESTRICTED RIGHTS. Programs, Ancillary Programs and Documentation, delivered subject to the Department of Defense Federal Acquisition Regulation Supplement, are "commercial computer software" as set forth in DFARS 227.7202, Commercial Computer Software and Commercial Computer Software Documentation, and as such, any use, duplication and disclosure of the Programs, Ancillary Programs and Documentation shall be subject to the restrictions contained in the applicable Siebel license agreement. All other use, duplication and disclosure of the Programs, Ancillary Programs and Documentation by the U.S. Government shall be subject to the applicable Siebel license agreement and the restrictions contained in subsection (c) of FAR 52.227-19, Commercial Computer Software - Restricted Rights (June 1987), or FAR 52.227-14, Rights in Data—General, including Alternate III (June 1987), as applicable. Contractor/licensor is Siebel Systems, Inc., 2207 Bridgepointe Parkway, San Mateo, CA 94404.

Proprietary Information
Siebel Systems, Inc. considers information included in this documentation and in Siebel eBusiness Applications Online Help to be Confidential Information. Your access to and use of this Confidential Information are subject to the terms and conditions of: (1) the applicable Siebel Systems software license agreement, which has been executed and with which you agree to comply; and (2) the proprietary and restricted rights notices included in this documentation.

Contents

Introduction to Object Interfaces

How This Guide Is Organized 18
Typographic Conventions 19
Revision History 20

Chapter 1. Siebel Programming Tools

Components of the Siebel Programming Environment 25
Supported Uses of Siebel Programming Languages 27
- Business Rule Definition 27
- Custom Behavior for User Interface Components 28

Choosing Your Scripting Language 29
Adding New Business Rules to a Business Component 30
The Siebel Script Editor 30
- Using the Siebel Script Editor 32
- Scripted Flag 34
- Script Editing Preferences 34

The Siebel Debugger 35
- Using the Siebel Debugger 36
- Debugging and Run-Time Preferences 37
- Checking Syntax 39
- Breakpoints 40
- Variable Window 40
- Siebel Calls Window 40

Script Tracing 41
Siebel Compiler and Run-Time Engine 43
Getting Started with Siebel VB 43
A Few Notes About Siebel eScript 47

Chapter 2. Programming

Installing Siebel Object Interfaces 53
Siebel Object Interfaces 53
Siebel COM Interfaces 54
Siebel Java Interfaces 58
Siebel CORBA Interfaces 59
Built-in Scripting 61
Usage Evaluation Matrix 61
Exposed Object Types 61
Application Object Type 62
Business Object Object Type 62
Business Component Object Type 63
Business Service Object Type 63
Applet Object Type 64
Property Set Object Type 65
User Interface Control Object Type 65
Application Factory Object Type 65
Summary of Exposed Object Types 66
Siebel Object Interface Method Syntax 66
Getting Started with the Siebel Object Interfaces 69
Accessing Siebel COM Interfaces 70
Accessing the Siebel Web Client Automation Server 71
Accessing the Siebel Mobile/Dedicated Web Client Automation Server 72
Instantiating the Siebel COM Data Server 75
Instantiating the Siebel COM Data Control 78

Java Data Bean 81
Accessing Siebel CORBA Interfaces 88
Initializing ORB and Binding Siebel App Factory (Orbix) 88
Initializing ORB and Binding Siebel App Factory (Visibroker) 90
Siebel Object Interface Methods 92
Locating Objects 93
Accessing Business Components 94
Navigation 99
User Interaction 99
Global State Properties and Functions 100
Creating a Siebel Application Object in CORBA 100
Variable Scoping for Siebel Script Variables 101
Local Variables 101
Module Variables 101
Global Variables 103
Inter-Application Variable Methods 103
Tracing 104
Siebel Object Interface Events and Siebel Extension Events 104
Event Method Syntax 105
How Your Script Affects Program Flow 106
When Events Occur 109
Siebel Business Component Events 109
Applet Events 111
Application Events 112
Connect String 112
Error Handling 115

Chapter 3. Interfaces Reference

Object Interface Methods Tables 120
Applet 120
Application 121
Business Component 123
Business Object 125
Business Service 125
Control 126
Property Set 127
SiebelApp Factory 128
Siebel Exceptions 128
Miscellaneous Methods and Events 128
Object Interface Events 129
Applet 129
Application 130
Business Component Events 130
Business Service Events 131
Siebel Constants 132
Applet Methods 133
ActiveMode 133
BusComp 134
BusObject 134
FindActiveXControl 135
FindControl 136
InvokeMethod 136
Name 138
Applet Events 138
Applet_ChangeFieldValue 139
Applet_ChangeRecord 140
Applet_InvokeMethod 140

Applet_Load 141
Applet_PreInvokeMethod 142
WebApplet_InvokeMethod 142
WebApplet_Load 143
WebApplet_PreCanInvokeMethod 144
WebApplet_PreInvokeMethod 145
WebApplet_ShowControl 145
WebApplet_ShowListColumn 148
Application Methods 151
ActiveApplet 153
ActiveBusComp 153
ActiveBusObject 153
ActiveViewName 154
Attach 155
CurrencyCode 155
Detach 156
EnableExceptions 156
FindApplet 157
GetBusObject 157
GetDataSource 158
GetLastErrCode 159
GetLastErrText 160
GetProfileAttr 160
GetService 161
GetSharedGlobal 162
GotoView 163
InvokeMethod 164
LoadObjects 165
LoadUserAttributes 166
Login 167
LoginId 169

LoginName 170
Logoff 170
LookupMessage 171
LookupValue 172
Name 172
NewPropertySet 173
PositionId 173
PositionName 174
RaiseError 175
RaiseErrorText 176
Release 178
SetPositionId 178
SetPositionName 179
SetProfileAttr 179
SetSharedGlobal 180
SWEAlert 181
Trace 182
TraceOff 184
TraceOn 185
Application Events 189
Application_Close 189
Application_InvokeMethod 190
Application_Navigate 191
Application_PreInvokeMethod 191
Application_PreNavigate 193
Application_Start 194
Business Component Methods 196
ActivateField 198
ActivateMultipleFields 199
Associate 201
BusObject 202

ClearToQuery 203
DeactivateFields 204
DeleteRecord 205
ExecuteQuery 206
ExecuteQuery2 209
FirstRecord 210
FirstSelected 211
GetAssocBusComp 212
GetFieldValue 214
GetFormattedFieldValue 216
GetLastErrCode 219
GetLastErrText 219
GetMultipleFieldValues 220
GetMVGBusComp 220
GetNamedSearch 222
GetPicklistBusComp 222
GetSearchExpr 224
GetSearchSpec 225
GetUserProperty 225
GetViewMode 226
InvokeMethod 228
InvokeMethod Methods 230
LastRecord 233
Name 234
NewRecord 234
NextRecord 236
NextSelected 237
ParentBusComp 237
Pick 238
PreviousRecord 239
RefineQuery 241

Release 241
SearchExpr 242
SetFieldValue 243
SetFormattedFieldValue 245
SetMultipleFieldValues 247
SetNamedSearch 248
SetSearchExpr 249
SetSearchSpec 251
SetSortSpec 255
SetUserProperty 257
SetViewMode 258
UndoRecord 262
ViewMode 262
WriteRecord 263
Business Component Events 266
BusComp_Associate 267
BusComp_ChangeRecord 267
BusComp_CopyRecord 269
BusComp_DeleteRecord 269
BusComp_InvokeMethod 270
BusComp_NewRecord 271
BusComp_PreAssociate 271
BusComp_PreCopyRecord 272
BusComp_PreDeleteRecord 272
BusComp_PreGetFieldValue 274
BusComp_PreInvokeMethod 275
BusComp_PreNewRecord 275
BusComp_PreQuery 276
BusComp_PreSetFieldValue 277
BusComp_PreWriteRecord 279
BusComp_Query 280

BusComp_SetFieldValue 281
BusComp_WriteRecord 282
Business Object Methods 283
GetBusComp 283
GetLastErrCode 284
GetLastErrText 285
Name 285
Release 286
Business Service Methods 287
GetFirstProperty 287
GetLastErrCode 288
GetLastErrText 288
GetNextProperty 289
GetProperty 290
InvokeMethod 290
Name 292
PropertyExists 293
Release 293
RemoveProperty 294
SetProperty 294
Business Service Events 295
Service_InvokeMethod 295
Service_PreCanInvokeMethod 297
Service_PreInvokeMethod 298
Control Methods 301
Applet 302
BusComp 302
GetProperty 303
GetValue 303
Name 304
SetLabelProperty 305

SetProperty 307
SetValue 309
Property Set Methods 309
AddChild 310
Copy 311
GetChild 313
GetChildCount 314
GetFirstProperty 315
GetNextProperty 316
GetProperty 317
GetPropertyCount 318
GetType 319
GetValue 319
InsertChildAt 320
PropertyExists 321
RemoveChild 321
RemoveProperty 322
Reset 323
SetProperty 323
SetType 324
SetValue 325
SiebelAppFactory Methods 325
CreateSiebelAppObject 325
Release 327
Miscellaneous Methods 327
GetErrorCode 328
GetErrorMessage(); 328
TheApplication 329

Chapter 4. Accessing Siebel COM Data Server with C++

Building the Siebel COM Client in C++ 331
Testing Your Program 344

Chapter 5. COM Data Control Quick Reference

Application 345
Business Component 348
Business Object 352
Business Service 352
Property Set 353

Chapter 6. COM Data Server Quick Reference

Application 357
Business Component 360
Business Object 364
Business Service 364
Property Set 365

Chapter 7. Mobile/Dedicated Web Client Automation Server Quick Reference

Application 369
Business Component 372
Business Object 376
Business Service 377
Property Set 378

Chapter 8. Siebel Web Client Automation Server Quick Reference

SiebelHTMLApplication 381
SiebelService 382
PropertySet 382

Chapter 9. Java Data Bean Quick Reference

Data Bean 385
Business Component 387
Business Object 391
Business Service 391
Property Set 392
SiebelException 394

Chapter 10. CORBA Quick Reference

CORBA Interfaces and Unicode 395
Application 396
Business Component 398
Business Object 401
Business Service 402
Property Set 403
SiebelAppFactory 405
SiebelException 405
desc 405
id 406

Chapter 11. Siebel VB Quick Reference

Applet 407
Application 409
Business Component 412
Business Object 418
Business Service 418
Property Set 420
Miscellaneous Interfaces and Events 422

Chapter 12. Browser Scripting

Browser Script Events and Methods 423
Applet 424
Application 425
Business Component 427
Business Object 428
Business Service 429
PropertySet 430
Control 432
Supported DOM Events for High Interactivity Mode 433
Supported DOM Events for Standard Interactivity Mode 434

Chapter 13. Siebel eScript Quick Reference

Applet 437
Application 439
Business Component 441
Business Object 447
Business Service 447
PropertySet 449
Miscellaneous Interfaces and Events 450

Chapter 14. Invoking Custom Methods with MiniButtons

Invoking Custom Methods 451

Index

Introduction to Object Interfaces

This guide provides information about the methods that can be used to access Siebel events and methods from both internal and external programming environments. It also includes syntax information for the events and methods themselves.

The audience for this guide consists of:

Siebel Application Developers	Persons who plan, implement, and configure Siebel applications, possibly adding new functionality.

Programmers with experience in other languages can use these volumes to become proficient in the languages embedded in Siebel Tools. Those with no programming experience may need to use other sources for basic information about programming.

Product Modules and Options

This Siebel Bookshelf contains descriptions of modules that are optional and for which you may not have purchased a license. Siebel's Sample Database also includes data related to these optional modules. As a result, your software implementation may differ from descriptions in this Bookshelf. To find out more about the modules your organization has purchased, see your corporate purchasing agent or your Siebel sales representative.

How This Guide Is Organized

This book comprises three chapters and seven parallel appendices. The first chapter describes the programming environment embedded in Siebel Tools. The second details the interfaces available for accessing Siebel objects. The third explains each of the Siebel events and methods, grouped by the type of object with which they can be used. The appendices, one for each interface, provide quick references to the syntax for Siebel events and methods.

Typographic Conventions

This guide uses the following typographic conventions:

To Represent	Syntax Is
Events, objects, and methods	Initial letter uppercase: BusComp.SetFieldValue(*FieldName, FieldValue*)
Arguments to statements or functions	Lowercase, italics; an internal capital may be used to indicate multiple English words: *variable, rate, prompt$*
Optional arguments and/or characters	Italicized arguments and/or characters in brackets: [*caption*], [*type*], [$]
Required choice for an argument from a list of choices	A list inside braces, with OR operator (|) separating choices: {Goto *label* | Resume Next | Goto 0}

Revision History

Siebel Object Interfaces Reference

Version 7.5.3, Rev. A

Table 1. Changes Made in Version 7.5.3, Rev. A

Topic	Revision
"COM Data Server" on page 56	Added argument information.
"ExecuteQuery" on page 206	Modified description and usage information.
"GotoView" on page 163	Modified usage information.
"InvokeMethod" on page 290	Modified syntax information.
"NewRecord" on page 234	Removed usage note.
"Script Tracing" on page 41	Revised logging instructions.
"Siebel Object Interface Method Syntax" on page 66	Added information on causes of Type Mismatch.

Additional Changes:

- Improved consistency in text.

Version 7.5.3

Table 2. Changes Made in Version 7.5.3

Topic	Revision
"Accessing the Siebel Web Client Automation Server" on page 71	Revised for 7.5.3: Added configuration information.
"Application_PreNavigate" on page 193	Revised for 7.5.3: Added example.
"DeactivateFields" on page 204	Revised for 7.5.3: Modified information on what fields are affected.
"GetDataSource" on page 158	New for 7.5.3: Added information on supported method.

Table 2. Changes Made in Version 7.5.3

Topic	Revision
"GetLastErrCode" on page 159	Revised for 7.5.3: Added information on execution status.
"GetNextProperty" on page 316	Revised for 7.5.3: Updated example.
"GetService" on page 161	Revised for 7.5.3: Updated usage information.
"GetSharedGlobal" on page 162 and "SetSharedGlobal" on page 180	Revised for 7.5.3: Revised method description.
"Initializing ORB and Binding Siebel App Factory (Orbix)" on page 88	Revised for 7.5.3: Revised entire topic, including code sample.
"InvokeMethod" on page 228	New for 7.5.3: Added information on supported methods.
"Login Errors" on page 87	New for 7.5.3: Added new topic.
"LookupValue" on page 172	New for 7.5.3: Added information on supported method.
"RaiseError" on page 175	Revised for 7.5.3: Added information on user-defined errors.
"Script Tracing" on page 41	New for 7.5.3: Added information on new feature.
"SetFieldValue" on page 243	Revised for 7.5.3: Modified usage information.
"SetLabelProperty" on page 305	New for 7.5.3: Added new command.
"SetViewMode" on page 258	Revised for 7.5.3: Modified syntax information.
"SWEAlert" on page 181	New for 7.5.3: Added new command to replace Alert.
"WebApplet_InvokeMethod" on page 142	Revised for 7.5.3: Modified information on triggers.
"WebApplet_Load" on page 143	Revised for 7.5.3: Added usage information.

Additional Changes:

- Modified examples to correct syntax errors.

- Modified text to improve clarity and readability.

Version 7.5, Rev. C

Table 3. Changes Made in Version 7.5, Rev. C

Topic	Revision
"Accessing the Siebel Web Client Automation Server" on page 71	Added information on usage with web clients.
"BusComp_WriteRecord" on page 282	Added information on SetFieldValue usage.
"Property Set" on page 392	Revised syntax of the Copy method.
"SetViewMode" on page 258	Expanded information on SiebelViewMode constants.
"Supported DOM Events for High Interactivity Mode" on page 433	Added note about DOM events. Added usage information about OnClick.
"Using the Siebel Script Editor" on page 32	Revised instructions.

Additional Changes:

- Modified examples to correct syntax errors.
- Modified text to improve clarity and readability.

Version 7.5, Rev. B

Table 4. Changes Made in Version 7.5, Rev. B

Topic	Revision
"Applet Events" on page 111	Added information on High Interactivity mode.
"Connect String" on page 112	Added a connect string for PowerBuilder.
"ExecuteQuery" on page 206	Revised usage information.
"Exposed Object Types" on page 61	Added a paragraph about passing objects.

Table 4. Changes Made in Version 7.5, Rev. B

Topic	Revision
"Getting Started with Siebel VB" on page 43	Revised information on destroying objects.
"Instantiating the Siebel COM Data Server" on page 75	Added a note about the debug environment.
"InvokeMethod" on page 228	Revised VB Syntax.
"NewRecord" on page 234	Revised usage information.
"SetFieldValue" on page 243	Revised usage information.
"SetSearchExpr" on page 249 and "SetSearchSpec" on page 251	Revised usage information.
"SetViewMode" on page 258	Revised description.

Additional Changes:

- Modified examples to correct syntax errors.
- Modified text to improve clarity and readability.

Version 7.5, Rev. A

Table 5. Changes Made in Version 7.5, Rev. A

Topic	Revision
"Application_InvokeMethod" on page 190	Added information on Browser script implementation
"Application_Start" on page 194	Added information on when the event is called.
"Choosing Your Scripting Language" on page 29	Added information on changing script languages.
"ExecuteQuery" on page 206	Added information on querying child business components.
"NewRecord" on page 234	Added values for JavaDataBean.

Table 5. Changes Made in Version 7.5, Rev. A

Topic	Revision
"SetFieldValue" on page 243	Added information on Standard Interactivity mode.
"TheApplication" on page 329	Added information on determining data sources.

Additional Changes:

- Modified eScript examples to correct syntax errors.
- Modified examples to remove deprecated commands.

Siebel Programming Tools 1

The Siebel applications include two programming languages. Siebel VB is a Visual Basic-like programming environment that includes an editor, debugger, interpreter and compiler. Siebel VB runs on the Windows operating system only. Siebel eScript is, similarly, a JavaScript-like programming environment, which uses the same tools that Siebel VB uses. Siebel eScript runs on both Windows and Unix operating systems. With these built-in languages, you can extend and configure your Siebel application beyond the capabilities provided by declarative object property definition. The languages are integrated with other Siebel tools, such as the Applet Designer, Siebel CTI, and Siebel SmartScript. Using this integration you can define object properties both with the designer and by attaching scripts.

You should regard coding as a last resort. Siebel Tools provides many ways to configure your Siebel application without coding, and these methods should be exhausted before you attempt to write your own code, for three reasons:

- Using Siebel Tools is easier than writing code.
- More important, your code may not survive an upgrade. Customizations created directly in Siebel Tools are upgraded automatically when you upgrade your Siebel application, but code is not touched, and it may need to be reviewed following an upgrade.
- Finally, declarative configuration through Siebel Tools results in better performance than implementing the same functionality through code.

Components of the Siebel Programming Environment

The individual components of the Siebel programming environment include:

- **Server Script:**

- **Siebel VB language.** A programming language that is syntactically and semantically compatible with Microsoft Visual Basic™. Because the language uses most of the same commands and standards as Microsoft Visual Basic, you can extend your Siebel application and reduce training costs.
- **Siebel eScript language.** A programming language that is syntactically and semantically compatible with Netscape JavaScript™. In parallel with Siebel VB, the language uses most of the same commands and standards as JavaScript, giving you the same advantages in an alternative language. Moreover, you can use Siebel eScript on all Siebel-supported operating systems. Siebel VB is supported on Windows only.

- **Browser Script:** A new type of script in Siebel 7 that executes in and is interpreted by the Browser. Browser Scripts are written in JavaScript and interact with the Document Object Model (DOM) as well as with the Siebel Object Model available in the Browser through the Browser Interaction Manager. A developer can script the behavior of Siebel events as well as the Browser events that are exposed through the DOM. Be aware that the DOMs for Internet Explorer and Netscape Navigator are different. Browser Script may only be used with applications which run in High Interactivity mode, except when scripting Control events supported by the Browser Document Object Model.
- **Siebel Script Editor.** An integrated editor used to create, view, edit, and save custom program routines. The Siebel Editor is similar to the code editor that accompanies Microsoft's Visual Basic program. The Siebel Editor is described in more detail in "The Siebel Script Editor" on page 30.
- **Siebel Debugger.** Assists you in detecting errors contained within Siebel programming language routines. It does not assist in detecting errors outside of the context of custom program routines. The Siebel Debugger can be invoked automatically from Siebel applications when a run-time error occurs if the Siebel application was invoked with the debug option, /H, on the command start-up line. The Debugger can also be invoked from the Debug toolbar and Debug menu. The Debugger is described in more detail in "The Siebel Debugger" on page 35.
- **Compiler/Interpreter.** A nonvisual component of the Siebel programming languages that compiles and executes Siebel custom program routines. It is similar to Microsoft's Visual Basic Language Interpreter. Siebel language routines are compiled into p-code and stored with the other object definitions in the SRF file.

- **Object Interfaces.** A collection of selected objects that expose their data and functionality to custom routines. The interface provides access to Siebel business objects with defined methods, events, and associated data. The object interfaces are the subject of this book.

Supported Uses of Siebel Programming Languages

The Siebel programming languages provide the ability to extend the behavior of the Siebel application in specific ways. Supported extensions can be grouped into the following:

- "Business Rule Definition"
- "Custom Behavior for User Interface Components" on page 28

Business Rule Definition

The Siebel programming languages let you extend data validation beyond what is already provided for in the standard Siebel application. The unique validation requirements of a business can be satisfied by custom extension routines that implement the specific business rules prior to performing record manipulation operations, such as record write or record delete.

Data validation routines may incorporate validations based on data from sources within or outside the Siebel application. For example, a validation routine may verify that an opportunity revenue amount is greater than zero if the probability of the opportunity is more than 20 percent using internal Siebel data. Alternatively, an extension routine could verify the availability of a conference room prior to inserting a new activity record by reading the information from another application's database table.

The Siebel programming languages provide data manipulation capabilities that can be used to modify data, such as updating, inserting, and deleting records. For example, a custom routine can be used to set the value of one field based on the value of another before a new record is created. A custom routine could thus be used to set the value of opportunity probability based on a stage in the sales cycle, simplifying data entry.

The methods used to support data manipulation provide error notification. The Siebel programming language is notified of the error and has access to information so you can handle the error and take appropriate action.

Data manipulation methods in the Siebel programming languages conform to the same visibility rules as the standard Siebel applications user interface. For example, if a business object is readable but not editable because of visibility rules in the Siebel applications user interface, the same is true when you are accessing the object through the Siebel languages. These languages cannot circumvent the visibility rules or the security constraints enforced by the standard Siebel applications.

Custom Behavior for User Interface Components

With Siebel Applet Designer, you can add selected user interface objects to applets. With the Siebel programming languages, you can associate behavior to the objects. An example of this feature is placing a button on an applet which, when clicked, launches another program such as Excel.

With the Siebel programming languages, you can update a particular field based on the values of other fields. An extension routine could enforce the business rule that states, “If the sales cycle is at or past the Quote Submitted stage, do not allow the Revenue field to be modified.” The feature can also be used to support the user-specific data maintenance rule by restricting updates to certain fields based on the position of the current user.

Choosing Your Scripting Language

Browser and Server scripts are created in the Script Editor embedded in Siebel Tools. To choose your default Server scripting language, navigate to View > Options > Scripting tab > Scripting Language as shown in Figure 1. Otherwise you are prompted to pick a scripting language each time you edit a script, unless you check the “Do not prompt again, use the default scripting language” check box.

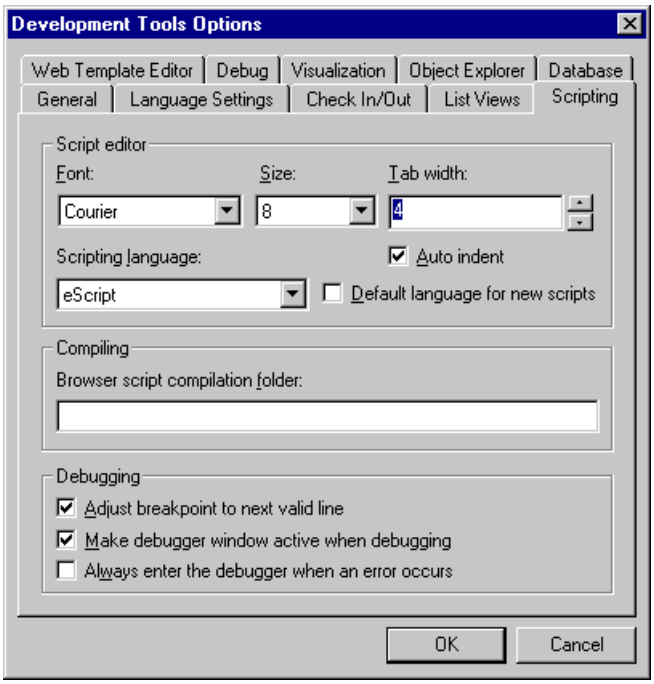


Figure 1. Script Editing Preferences

To specify your scripting language

- Choose View > Options > Scripting tab > Scripting Language.

If you have added a script to an object in the Tools Script Editor and later wish to change that object's scripting language (such as from eScript to VB), you must erase your entire script before changing the scripting language for that object.

Adding New Business Rules to a Business Component

The following procedure describes the steps required to add new business rules to a business component.

To add business rules to a business component

1. Start Siebel Tools.
2. Choose Repository > Check Out to lock the project from the server repository.
3. Select the business component using the Object Explorer and Object List Editor.
4. Right-click to bring up the menu, and choose Browser or Server Script.
5. Select the event from the Event List Tree applet and add your Server scripts in the script editor applet.
6. Validate the Siebel script syntax by selecting Debug: Check Syntax.
7. Choose File > Save to save the changes.
8. Compile the modified business component by pressing F7.
9. Press F5 to run the modified application and test the new control.
10. Choose Repository > Check In to check the modified project into the server repository.

The Siebel Script Editor

The Siebel Script Editor is a simple window-based editor designed to create and maintain Siebel VB, Siebel eScript, and Browser Script programs.

When creating Siebel custom programs, note the following:

- Check out or lock the project containing the object definitions being modified. If the project is not locked, you are unable to add any text in the Editor window.
- Choose Debug > Check Syntax to verify the syntax of your Basic or eScript program. The Siebel Compiler reports any syntax errors and indicates the lines where they occur.

- Choose File > Save when you have finished entering and editing the custom statements to save your work. Closing the Siebel Script Editor without saving your work discards the changes.
- Before you run the application, you must compile the projects that you have modified and generate a new SRF file. For information on the Object Compiler, read *Siebel Developer's Reference*.
- Run the application with the new application extensions by choosing Debug > Start or clicking the Start button in the Debug toolbar. The Siebel application executes with the new modifications incorporated.
- You may inadvertently create programming errors that, when encountered, halts the execution of the extension routine. If you started Siebel applications in debug mode (/H option on the command start-up line), a message box opens indicating the nature of the error. You can then return to the Script Editor and choose Debug > Check Syntax. For further details, read "Checking Syntax" on page 39.
- When a script error is encountered by an end user, or when the Siebel application is not running in Debug mode, the application displays an appropriate error message and returns control back to the point in the standard Siebel code just prior to the error.

See Also

"Using the Siebel Script Editor" on page 32
"Scripted Flag" on page 34
"Script Editing Preferences" on page 34

Using the Siebel Script Editor

To access the Siebel Script Editor (shown in Figure 2), select an object definition in the Object List Editor and click the right mouse button. If the editor is available from that object type, you can select Edit Scripts from the dialog box. Siebel scripts can be attached to the object types application, applet, and business component.

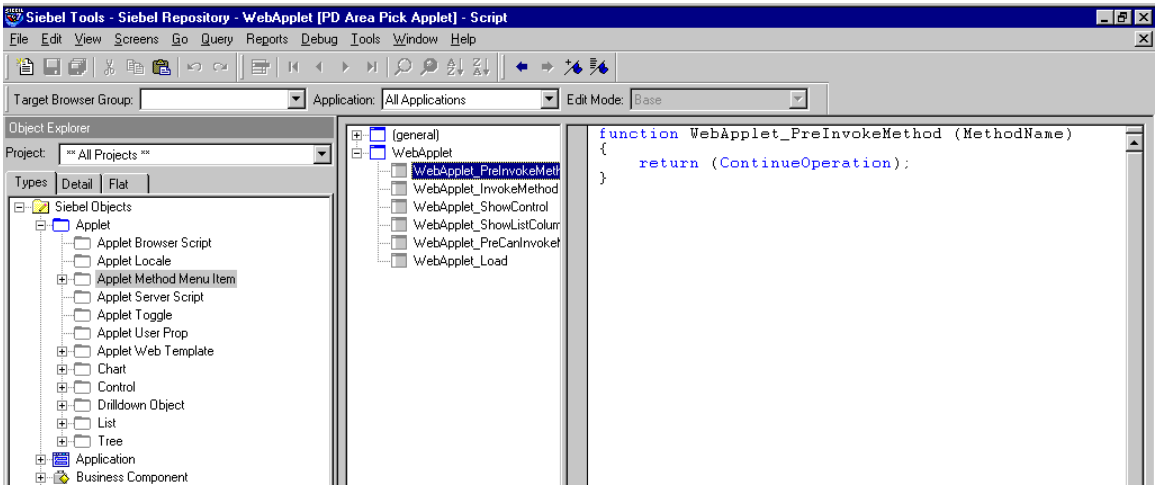


Figure 2. Siebel Script Editor

The Siebel Script Editor is a window-based editor similar to the Windows Notepad editor. The Editor's interface consists of a title bar, a drop-down list for specifying an object, a drop-down list for specifying an event, and a text entry window. There are vertical and horizontal scroll bars for scrolling within the entry region.

When using the Siebel Script Editor, you can do the following:

- Cut, copy, and paste the text from one location to another location within or from outside the Editor. When pasting into the Editor, avoid having two code blocks with the same name by placing the code between the function <Name> {} (eScript) or Sub <Name> / End Sub block (VB).
- Import and export Siebel scripts.

- Associate a given Siebel script with a predefined object event, such as a PreSetFieldValue event for a Business Component.
- Debug a custom routine by invoking the Siebel Debugger.
- Compile a custom routine by invoking the Siebel Compiler from the Siebel Script Editor.

The editor functions can be accessed from the title bar menus, keyboard shortcuts, and the Edit toolbar. The following are File menu options pertaining to Siebel VB and Siebel eScript:

- **Import.** Imports Siebel scripts.
- **Export.** Exports Siebel scripts.
- **Save.** Saves a Siebel script. Be sure to save your scripts before exiting the editor.
- **Exit.** Closes the Siebel Script Editor window.

The following are Edit menu options pertaining to the Siebel Editor:

- **Cut.** Deletes selection and saves it to the Clipboard.
- **Copy.** Copies selection to the Clipboard.
- **Paste.** Copies what is on the Clipboard to the selected area.
- **Delete.** Deletes selection.
- **Select All.** Selects the entire script.
- **Find.** Displays the Find in Script dialog box. You can search for text or white space.
- **Replace.** Displays the Replace in Script dialog box. You can search and replace text or white space.

Some editing functions are available from the Edit toolbar. The toolbar buttons perform the same functions as the comparably named menu options described previously.

- Remove extraneous comments.
- Indent code using tabs instead of spaces.

- Use short variable and method names.
- Create subroutines and functions.

Scripted Flag

For object types that can have a Siebel script attached to them (applet, application, and business component), there is a property in the Object List Editor called Scripted. This property indicates whether Siebel scripts are attached to the object definition. A check mark indicates the presence of scripts; no check mark indicates that the object definition has no scripts.

Script Editing Preferences

To access the script editing preferences, choose View > Options, and then click the Scripting tab (see Figure 3).

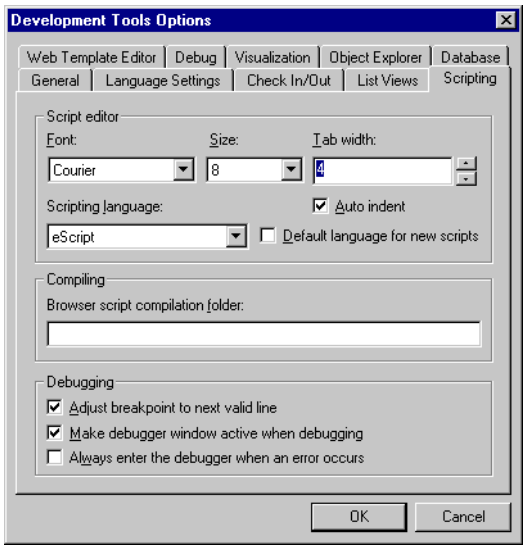


Figure 3. Script Editing Preferences

The following window features in this tabbed form apply to script editing:

- **Font field.** Used to select the font for display of scripts.
- **Tab Width field.** Defines the number of spaces for a tab character. The default is four spaces.
- **Auto Indent field.** When checked, each succeeding line is indented to the position set by the current line.
- **Scripting Language.** Used to specify the default scripting language for server side components.
- **Compiling.** Used to specify the folder where `bscripts\all` resides. This is where Browser Scripts are generated. For example, if the value is set to `D:\sea750\client\PUBLIC\enu`, then the Browser Script files are generated to `D:\sea750\client\PUBLIC\enu\<genbscript time stamped folder>\bscripts\all`.

The Siebel Debugger

The Siebel Debugger assists in editing and removing errors from scripts written in Siebel VB and Siebel eScript.

The Siebel Debugger uses the Siebel Script Editor window plus a diagnostic window to display program variables and their values. The Debugger helps you locate and correct execution errors in custom program routines. You can use it to slow or suspend execution of the program routines so that the program flow and variable contents can be examined.

With the Siebel Debugger you can do the following:

- Set and clear breakpoints in your Siebel script. A breakpoint is a marker on a line of Basic code that tells Basic to suspend execution at that line so that the state of the program can be examined using the Debugger.
- Step over a line of code. If the current line is a call to a subroutine or function, the Debugger stops at the next line in the current procedure (skipping the subroutine).

- Step into a subroutine of custom routine code. Step Into is used to execute one line of code in the Debugger. If the current line is a call to a subroutine or function, the Debugger stops at the first line of that function. Otherwise, the Debugger stops at the next line of the current procedure.
- View the value of custom routine variables. The Siebel Debugger includes a window in which variables and their values are displayed. This window can be used to monitor the values of specific variables as the custom routine executes.

See Also

"Using the Siebel Debugger"
"Debugging and Run-Time Preferences" on page 37
"Checking Syntax" on page 39
"Breakpoints" on page 40
"Variable Window" on page 40
"Siebel Calls Window" on page 40

Using the Siebel Debugger

You can access the Debugger in several ways:

- You can set breakpoints in the current routine and begin execution by clicking the Start button. Execution is suspended when one of the lines that contains a breakpoint is about to be executed. The Debugger is activated and it highlights the line containing the breakpoint.
- If an executing program encounters a run-time error, such as an unhandled Siebel VB or eScript error, execution is suspended, the Debugger is activated, and it highlights the line containing the error.

Debug options are available from the Debug title bar menu and the Debug toolbar. See the Siebel Toolbars and Menus topics for details.

Debugging and Run-Time Preferences

To access the debugging preferences, select View > Options and then click the Scripting tab (see Figure 4).

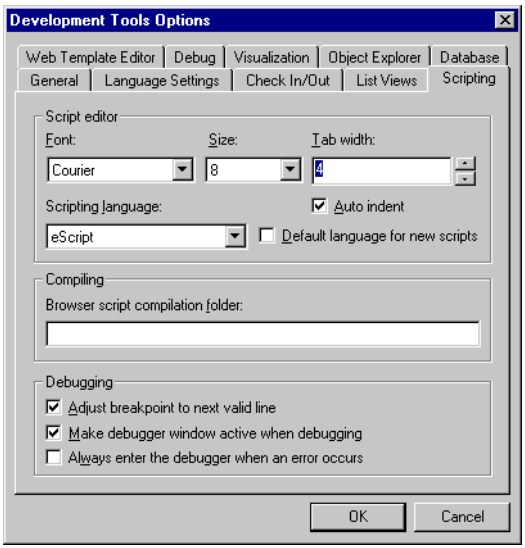


Figure 4. Debugging Preferences

This dialog box has the following debugger settings:

- **Adjust breakpoint to next valid line.** When breakpoints are deleted on invalid lines, this option creates a breakpoint at the next valid line.
- **Make debugger window active when debugging.** The Siebel Debugger window appears whenever you are in debug mode.
- **Always enter the debugger when an error occurs.** The Siebel Debugger window appears whenever a script error occurs.

To access the run-time preferences, select Views > Options and then click the Debug tab (see Figure 5).

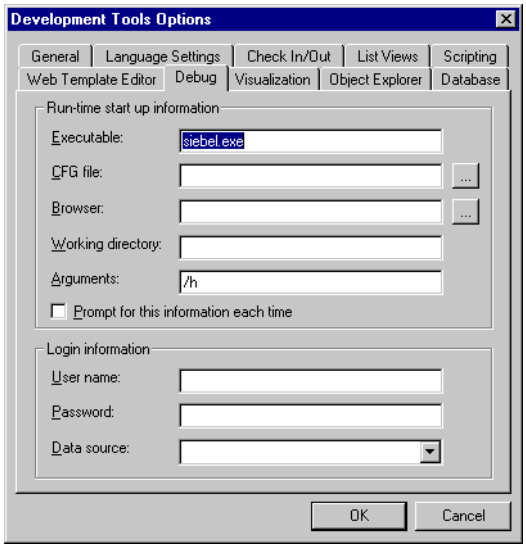


Figure 5. Run-Time Preferences

This dialog box has the following preference settings:

- **Executable.** The name of the Siebel Web Client executable (Siebel.exe).
- **CFG file.** Configuration file to be used by the client.
- **Browser.** The path to the browser executable.
- **Working directory.** Siebel root directory (location of dlls).
- **Arguments.** Additional line options for starting the application. Common arguments are:
  - /h - to enable local debugging of Server scripts
  - /s <filename> - to enable SQL spooling
- **User name.** Login name of the user.

- **Password.** Password of the user name.
- **Data source.** Default data source. Values listed depend upon the configuration file specified in the "cfg File" parameter.

Checking Syntax

The debugger includes a syntax checker to make sure that your script compiles properly. (It is your responsibility to see that the script does what you want it to.)

To check the syntax of your script

1. Click the Check Syntax button, or choose Debug > Check Syntax.

   Siebel Tools does a test compile. If you have made no errors, you get no response. If there are errors in your script, a message box appears describing the error. The message box has two buttons: Next Error and Go to Line. If there is more than one error, it is best to handle them one at a time.

2. Click Go to Line.

   The cursor falls on the line of the script containing the error, with the line highlighted.

3. Correct the code and check the syntax again.

   If the syntax of the line you changed is now correct, the message box displays the next error, if any.

4. Repeat Step 2 and Step 3 until you see no more messages.
5. Choose File > Save to save your file, and close the Siebel application.
6. Press F7 to compile the SRF file.
7. When the compilation finishes, click Run or press F5 to restart the application.

CAUTION: The Check Syntax function checks only for syntax errors and errors that stem from failure to properly initialize objects or variables. It does not check other types of errors, and cannot trap errors in logic that may cause run-time errors.

At this point, your script should run. Test it to see if it gives you the desired results. The following sections describe debugging tools to help you accomplish that end.

CAUTION: The Check Syntax command checks only the script in the active object definition. If there are errors in other scripts, you are not able to compile the SRF file.

Breakpoints

A breakpoint is a marker on a line of Siebel code that tells the interpreter to suspend execution at that line so that the state of the program can be examined using the Debugger. There are two ways to set breakpoints on lines of Siebel code when editing, and there is an additional way to set a breakpoint when debugging.

- When editing, place the cursor on the line of code on which you wish to set a breakpoint by clicking on that line with the mouse or using the arrow keys. To toggle the breakpoint, press F9 or click the toolbar button. If the line already has a breakpoint in it, pressing F9 or the toolbar button clears the breakpoint.
- When debugging, clicking on a line of Siebel code toggles a breakpoint on that line.

Variable Window

The variable window is used to examine the contents of the variables associated with a Siebel script when debugging.

Siebel Calls Window

The Calls window contains a list of subroutine and function calls that were executed prior to the current line. To access the Calls window, click the Calls button in the Debugger toolbar when you are running the Debugger. A typical Calls window may contain several lines, one for each subroutine entered into and not yet completed.

Selecting an entry in this list box causes the interpreter to shift to that entry. The Debugger window displays the line of code that made the call, and the Variable window displays the variables that are associated with the procedure that made the call.

Script Tracing

As part of debugging scripts you can run a trace on allocations, events, and SQL commands. The tracing can be activated for specified user accounts, such as your development team. The Siebel Server sends the tracing information to a log file.

To enable logging

1. Navigate to Server Administration > Components.
2. Select a component to log. Not all components support logging, but the majority do.
3. Click the Component Event Configuration tab.
4. Select the Object Manager Extension Language Log record. If this record does not exist, then the selected component does not support logging.
5. Set the Log Level to 1. To disable logging when you are done, set the Log Level to 0 (zero).
6. Click the Component Parameters tab.
7. (Optional) To display only the script tracing parameters, query for:
   Parameter Alias = Trace*
   Subsystem = Object Manager

   Changes to the script tracing parameters can take effect immediately. If you want changes to take effect now, then make changes to the values in the Current Value column. If you want the changes to take effect only after a restart, then make changes to the values in the Value on Restart column.
8. Set one or more tracing parameters from the following table.

Information to Trace	Parameter Alias	Settings for Current Value and Value on Restart
Allocations	TraceAlloc	0 (zero) to disable logging, 1 to enable logging
Events	TraceEvents	0 (zero) to disable logging, 1 to enable logging

Information to Trace	Parameter Alias	Settings for Current Value and Value on Restart
SQL Commands	TraceSql	0 (zero) to disable logging, 1 to enable logging
Users	TraceUser	Comma-separated list of user names. Do not use spaces. Example: sadmin,mmasters,hkim,cconnors

The following is a sample trace.

```
2021 2003-04-09 15:37:20 2003-04-09 16:40:52 -0700 00000022 001 001f 0001 09
SCCObjMgr_enu 47126 1680 1584 C:\sea752\siebsrvr\log\SCCObjMgr_enu_47126.log 7.5.3
[16122] ENU

ObjMgrSessionInfoObjMgrLogin32003-04-09 15:37:20Login name : SADMIN

ObjMgrSessionInfoObjMgrAuth32003-04-09 15:37:20Authentication name : SADMIN

ObjMgrSessionInfoObjMgrLogin32003-04-09 15:37:20Session Type: Regular Session

GenericLogGenericError12003-04-09 15:37:20Invocation of Applet Menu New
Service::NewExpense is not allowed.

GenericLogGenericError12003-04-09 15:37:20Invocation of Applet Menu New
Service::NewTimeSheet is not allowed.

ObjMgrExtLangLogObjMgrExtLangLog02003-04-09 15:38:27[User: SADMIN] EVENT, BEGIN,
BusComp [Account], BusComp_Query.

ObjMgrExtLangLogObjMgrExtLangLog02003-04-09 15:38:27[User: SADMIN] EVENT, END,
BusComp [Account], BusComp_Query.

ObjMgrExtLangLogObjMgrExtLangLog02003-04-09 15:38:58[User: SADMIN] EVENT, BEGIN,
BusComp [Account], BusComp_NewRecord.

ObjMgrExtLangLogObjMgrExtLangLog02003-04-09 15:38:58[User: SADMIN] EVENT, END,
BusComp [Account], BusComp_NewRecord.

ObjMgrExtLangLogObjMgrExtLangLog02003-04-09 15:39:08[User: SADMIN] EVENT, BEGIN,
BusComp [Account], BusComp_PreSetFieldValue.

ObjMgrExtLangLogObjMgrExtLangLog02003-04-09 15:39:08[User: SADMIN] EVENT, END,
BusComp [Account], BusComp_PreSetFieldValue.

ObjMgrSessionInfoObjMgrLogin32003-04-09 16:40:52Username: SADMIN, Login Status:
Attempt,  Session Id: !1.690.b816.3e94a0a0, IP Address: 172.20.94.66
```

Script tracing is not the same as file-based tracing. For more information on file-based tracing, read "Trace" on page 182.

Siebel Compiler and Run-Time Engine

To invoke the Siebel Compiler and Run-time Engine, click the Compile button on the Debugger toolbar, or press F7. You can also invoke it when compiling a project containing object definitions with associated Siebel scripts. The Siebel Compiler and Run-time Engine has no user interface of its own. When the compiler is invoked, it compiles the custom routines and returns a message when completed that indicates success or failure.

Compilation Order Considerations

The Siebel Compiler compiles Siebel VB functions and procedures in alphabetical order within an object definition. If a function or procedure calls another function or procedure that has not been defined, the compiler generates an error message in the form:

function_name Is An Unknown Function

To avoid this error, use the Declare statement to declare the function or procedure in the (general) (declarations) section. For more information, read the VB Language Reference topics within *Siebel Tools Online Help*.

Siebel eScript does not require forward declaration of functions.

Getting Started with Siebel VB

If you have never programmed in Visual Basic before, you may want to start by reading the VB Language Reference topics within *Siebel Tools Online Help*. They include a reference to the internal VB program constructs, statements, and functions. You need to understand how these objects behave before you can program using the Siebel object methods and events.

Declare your variables. As a general rule, using the Option Explicit statement is helpful as it forces you to declare your variables (using the Dim statement) before you use them. Doing so makes it easier for others to understand your code, and for you to debug the code. You can declare a variable without giving it a data type, but if you do not specify a data type, Siebel VB assumes the type Variant, which requires 16 bytes—twice as much memory as the next smallest data type. If you can avoid Variant variables, you reduce the amount of memory required by your code, which may make execution faster. In Siebel VB, you place Option commands in the (general) (declarations) window.

Use standardized naming conventions. Another way to improve the readability of your code is to follow a set of standardized naming conventions. It does not really matter what conventions you follow as long as everyone in the programming group follows the same conventions. One very common convention is to prefix each variable with a letter denoting its type, as shown here.

Data Type	Symbol	Example
String	s	`sName`
Integer	i	`iReturn`
Long integer	l	`lBigCount`
Single-precision number	si	`siAllowance`
Double-precision number	d	`dBudget`
Object	o	`oBusComp`
Currency	c	`cAmtOwed`

You can also use suffix characters on your variable names.

Use the Me object reference. The special object reference *Me* is a VB shorthand for "the current object." You should use it in place of references to active business objects. For example, in a business component event handler, you should use Me in place of *ActiveBusComp,* as shown in the following example. You can see other examples of Me in "ParentBusComp" on page 237, "SetViewMode" on page 258, "BusComp_PreQuery" on page 276, "BusComp_PreWriteRecord" on page 279, and "ActiveMode" on page 133.

```
Function BusComp_PreSetFieldValue(FieldName As String, FieldValue
As String) As Integer

   If Val(Me.GetFieldValue("Rep %")) >75 Then
      ....
   End If

   BusComp_PreSetFieldValue = ContinueOperation

End Function
```

Trap run-time errors. The standard VB methods return numeric error codes, which are documented in the VB Language Reference topics within *Siebel Tools Online Help*. Siebel VB methods also may return error codes; however, they must be handled differently from those returned by the standard VB methods. For standard methods, you can use some combination of Err, ErrText, and Error, which are documented in *Siebel Tools Online Help*. Siebel methods use numeric error codes in the range from 4000 to 4999. When you access Siebel object interfaces through COM or ActiveX, use a construct of this form to see the text of the error message:

```
If errCode <> 0 Then
   ErrText = GetLastErrText
   TheApplication.RaiseErrorText ErrText
   Exit Sub
End If
```

NOTE: The GetLastErrText method is only available using interfaces external to Siebel Tools. Therefore, you can use it in Microsoft VB, but not in Siebel VB.

If you are working within the Siebel applications, especially in a LAN environment, where you cannot be sure that a record has not been changed or deleted by another user, create routines that keep the program from failing when it meets an unexpected condition. For information about error-handling routines, read the Language Overview topics in the Siebel VB Language Reference.

Make effective use of the Select Case construct. The Select Case construct chooses among any number of alternatives you require, based on the value of a single variable. This is greatly preferable to a series of nested If statements, because it simplifies code maintenance and also improves performance because the variable must be evaluated only once.

Use the With shortcut. Use the With statement to apply several methods to a single object. It makes the code easier to read, reduces typing, and improves performance. Instead of a series of statements such as:

```
Set oBusComp = objBusObject.GetBusComp("Opportunity")
oBusComp.ClearToQuery
oBusComp.SetSearchSpec . . .
oBusComp.ExecuteQuery ForwardBackward
oBusComp.FirstRecord
oBusComp.NewRecord NewAfter
oBusComp.SetFieldValue "QuoteNumber", sQuoteId
oBusComp.SetFieldValue "Account", sAccount
. . .
sSolutionId(cSolution) = oBusComp.GetFieldValue( "Id" )
. . .
```

use the following:

```
Set oBusComp = objBusObject.GetBusComp("Opportunity")
With oBusComp
   .ClearToQuery
   .SetSearchSpec . . .
   .ExecuteQuery ForwardBackward
   .FirstRecord
   .NewRecord NewAfter
   .SetFieldValue "QuoteNumber", sQuoteId
   .SetFieldValue "Account", sAccount
   . . .
   sSolutionId(cSolution) =.GetFieldValue( "Id" )
   . . .
End With
```

Use extreme care when working with date variables. When working with date variables extreme care has to be taken regarding the date format. GetFieldValue always returns the date in dd/mm/yyyy format (eventually followed by the time). As a result, applying the CVDate() function, which expects the regional setting, to the return value may cause an error. The GetFormattedFieldValue method uses the regional settings of the user's operating system. The regional setting specifies the year with two digits in most cases, thereby creating the possibility of Y2K non-compliance. For these reasons, you should use the following approach for performing date arithmetic.

To perform date arithmetic

1. Retrieve the value of date fields with the GetFieldValue method. For more information, read "GetFieldValue" on page 214.
2. Convert it into a date variable using the DateSerial() function.
3. Perform the required date arithmetic.

Here is an example:

```
Dim strDate as String, varDate as Variant
strDate = oBC.GetFieldValue("Date Field")
varDate =DateSerial(Val(Mid(strDate,7,4)),Val(Left(strDate,2)),_
   Val(Mid(strDate,4,2)))
[any date arithmetic]
```

Destroy any objects you have created when you no longer need them. While the interpreter theoretically takes care of object cleanup, complex code involving many layers of object instantiation may in some cases cause the interpreter to not release objects in a timely manner. This issue becomes more critical when accessing the application using the Siebel Object Manager. Therefore, explicit destruction of Siebel objects should occur in the procedure in which they are created.

To destroy an object in Siebel VB, set it to Nothing. The best practice is to destroy objects in the reverse order of creation. Destroy child objects before parent objects. For example:

```
Set oBusObj = TheApplication.GetBusObject("contact")
Set oBusComp= oBusObj.GetBusComp("contact")

[ Your code here ]

Set oBusComp = Nothing
Set oBusObj = Nothing
```

A Few Notes About Siebel eScript

There are some important differences between Siebel eScript and Siebel VB:

- Siebel eScript is case-sensitive; theApplication is different from TheApplication. Siebel VB is not case-sensitive.

- Siebel eScript does not distinguish between subroutines (which take no arguments) and functions (which take arguments). In Siebel eScript, every method is a function, whether or not it accepts arguments; therefore, it should be followed by a pair of parentheses.

Keep these differences in mind when you read the syntax diagrams. In many instances, the only difference between the VB syntax and the eScript syntax is that the eScript syntax requires the pair of parentheses at the end. In these instances, only the VB syntax is shown; you can derive the eScript syntax by adding the parentheses.

There are also some important differences between Siebel eScript and standard ECMAscript. Most important, Siebel eScript has no user interface functions. It cannot, therefore, be used to animate or control Web pages. Second, it contains two objects that are not part of standard ECMAscript: SELib and Clib. These objects implement a variety of C-like functions for interacting with the operating and file systems, and for file I/O. For details on these and other eScript functions not covered here, read the eScript Language Reference topics within *Siebel Tools Online Help*.

Declare your variables. Standard ECMAscript does not require that you declare variables. Variables are declared implicitly as soon as they are used. However, Siebel eScript requires you to declare variables with the var keyword. As a general rule, declare the variables used in a module before you use them. Doing so makes it easier for others to understand your code, and for you to debug the code. There is one notable exception to this standard. Declaring a variable inside a loop controller restricts the scope of that reference to within the loop. This prevents the variable from persisting; it can therefore be declared again inside another loop.

Use the this object reference. The special object reference *this* is eScript shorthand for “the current object.” You should use it in place of references to active business objects and components. For example, in a business component event handler, you should use *this* in place of *ActiveBusComp,* as shown in the following example.

```
function BusComp_PreQuery ()
{
   this.ActivateField("Account");
   this.ActivateField("Account Location");
   this.ClearToQuery();
   this.SetSortSpec( "Account(DESCENDING)," +
      " Account Location(DESCENDING)");
   this.ExecuteQuery();
```

```
   return (ContinueOperation);
}
```

Use the with shortcut. The with shortcut applies several methods to a single object. It makes the code easier to read, reduces typing, and improves performance. Instead of a series of statements such as:

```
var oBusComp = oBusObject.GetBusComp("Opportunity");
oBusComp.ClearToQuery();
oBusComp.SetSearchSpec( . . .);
oBusComp.ExecuteQuery(ForwardBackward)
oBusComp.FirstRecord();
oBusComp.NewRecord(NewAfter);
oBusComp.SetFieldValue("QuoteNumber", sQuoteId);
oBusComp.SetFieldValue("Account", sAccount)
. . .
sSolutionId(cSolution) = oBusComp.GetFieldValue( "Id" );
. . .
```

use the following:

```
var oBusComp = oBusObject.GetBusComp("Opportunity");
with oBusComp
{
   ClearToQuery();
   SetSearchSpec( . . .);
   ExecuteQuery(ForwardBackward)
   FirstRecord();
   NewRecord(NewAfter);
   SetFieldValue("QuoteNumber", sQuoteId);
   SetFieldValue("Account", sAccount)
}
. . .
sSolutionId(cSolution) = oBusComp.GetFieldValue( "Id" );
. . .
```

Make effective use of the Switch construct. Use the Switch construct to choose among any number of alternatives you require, based on the value of a single variable. This is greatly preferable to a series of nested If statements because it simplifies code maintenance. It also improves performance because the variable must be evaluated only once.

Destroy any objects you have created when you no longer need them. While the interpreter theoretically takes care of object cleanup, complex code involving many layers of object instantiation may in some cases cause the interpreter to not release objects in a timely manner. This issue becomes more critical when accessing the application using the Siebel Object Manager. Therefore, explicit destruction of Siebel objects should occur in the procedure in which they are created.

To destroy an object in Siebel eScript, set it to null, or set the variable containing it to another value. The best practice is to destroy objects in the reverse order of creation. Destroy child objects before parent objects. For example:

```
var oBusObject = TheApplication().GetBusObject("contact")
var oBusComp = oBusObject.GetBusComp("contact")

[ Your code here ]

oBusComp = null;
oBusObject = null;
```

Programming 2

Siebel object interfaces provide open interfaces into the Siebel applications, supporting integration between Siebel applications and external applications.

Siebel object interface definitions are based on Siebel business objects and declarative object definitions that can be configured and automatically upgraded to successive releases using Siebel Tools.

Siebel object interfaces are available to developers through the following technologies:

- Built-in scripting of Siebel objects using Siebel VB, Siebel eScript, and Browser Script
- Component Object Model (COM) using the Siebel Web Client Automation Server, Siebel COM Data Control, Siebel COM Data Server, and Siebel Mobile/ Dedicated Web Client Automation Server
- CORBA using Siebel CORBA Object Manager
- Java using Siebel Java Data Bean

Siebel developers can integrate client and server applications from a variety of vendors. Application integration typically requires that cooperative software application programs interactively pass data back and forth. In addition, application integration sometimes requires that one application “controls” or “automates” another application.

The Siebel object interfaces are a collection of methods on Siebel objects that expose their data and functions to custom routines written in Server Script, and also to other languages external to the Siebel application. The interfaces provide access to Siebel business objects with defined methods, events, and data.

CAUTION: Your Siebel application is a Web-based or client/server application designed to meet the sales and marketing information requirements of large multinational corporations. Use caution when extending the Siebel applications or accessing them through the interface described here, as this should be done only by trained technical professionals. Improper application configuration or use of these interfaces can cause your configured Siebel application to be less reliable, or to perform poorly. Always test your configured application thoroughly before production rollout.

Siebel Systems does not support the following:

- Functions developed through custom programming
- Custom-developed applications
- Specific performance characteristics of other vendors' software

In addition, Siebel business objects, the Siebel object interfaces, and their associated behavior and properties are defined at the sole discretion of Siebel Systems, Inc. Siebel Systems reserves the right to change the behavior, properties, and events at any time without notice.

This chapter describes the interface environments and object types. Chapter 3, "Interfaces Reference," describes the supported methods of the Siebel object interfaces and provides examples of how you can use them.

Installing Siebel Object Interfaces

Table 6 lists the installation procedure for each object interface.

Table 6. Interface Installation

Interface	Installation
CORBA Object Manager	Installed by running the CORBA Object Manager Installer.
Java Data Bean	Installed by the Siebel Enterprise Server Installer under a Typical installation, with the "EAI Siebel Connectors" option. For details, refer to *Siebel Server Installation Guide for Microsoft Windows* or *Siebel Server Installation Guide for UNIX*.
COM Data Control	Installed by the Siebel Enterprise Server Installer under a Typical installation, with the "EAI Siebel Connectors" option. COM Data Control is also installed with the OLE DB Provider and BizTalk Connector. For details, refer to *Siebel Server Installation Guide for Microsoft Windows* or *Siebel Server Installation Guide for UNIX*.
COM Data Server	Installed by default with the Mobile/Dedicated Web Client.
Siebel Mobile/Dedicated Web Client Automation Server	Installed by default with the Siebel Mobile/Dedicated Web Client.
Siebel Web Client Automation Server	Installed by default with the Siebel Mobile/Dedicated Web Client. Also installed by default with the Siebel Enterprise Server Installer.

Siebel Object Interfaces

Siebel object interfaces provide:

- "Siebel COM Interfaces" on page 54
- "Siebel CORBA Interfaces" on page 59
- "Siebel Java Interfaces" on page 58

- Built-in scripting of Siebel objects using Siebel VB, Siebel eScript, and Browser Script. For more information, read "Built-in Scripting" on page 61.

See Also "Usage Evaluation Matrix" on page 61

Siebel COM Interfaces

Siebel COM object interfaces can be accessed in four ways: COM Data Control, COM Data Server, Siebel Web Client Automation Server, and Siebel Mobile/Dedicated Web Client Automation Server.

NOTE: The programming environment you use may impose limitations on the functionality of COM servers. For example, code using the Data Server written in VB should not be implemented as a Windows NT service.

COM Data Control

The Siebel COM Data Control interfaces allow external applications to access Siebel business objects remotely.

To develop an application using the Siebel COM Data Control, you must have a Siebel Application Object Manager set up and running on a Siebel Server. Refer to *Siebel Server Administration Guide* for information about installing and configuring the Siebel Object Manager.

Any external applications or components that use Siebel COM Data Control connects and communicates with Siebel Application Object Manager. The Siebel Application Object Manager, which could be running on a remote Siebel Server, is a multi-threaded, multiprocess application server that hosts Siebel business objects and supports session-based connections by clients. Figure 6 shows how external applications use *Siebel COM Data Control* to communicate with the Siebel application.

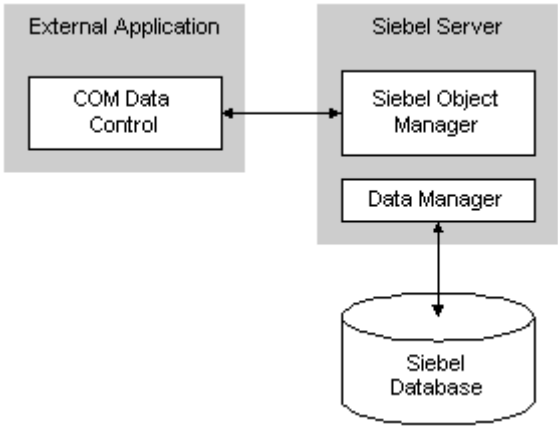


Figure 6. Siebel COM Data Control

COM Data Server

Figure 7 shows how external applications use Siebel COM Data Server without having to access the user interface objects.

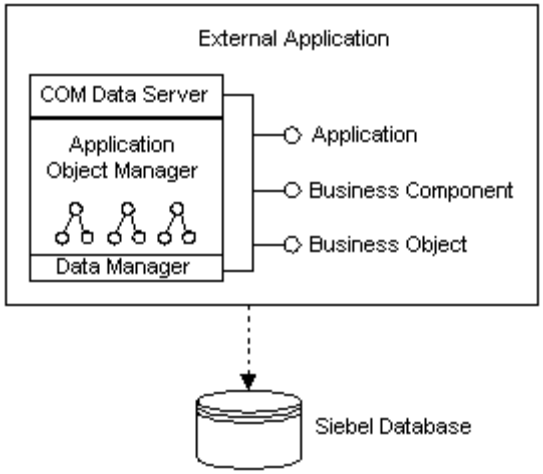


Figure 7. Siebel COM Data Server

You can expect differences in performance between Siebel COM Data Server and Siebel Mobile/Dedicated Web Client Automation Server. This is due in part to the fact that COM Data Server is a DLL running in the same address space as the calling program, while Automation Server is an executable that runs in its own address space. DLLs that are accessed by a server task must be thread safe.

Calls to Siebel COM Data Server methods require arguments passed by reference and not by value.

Siebel Web Client Automation Server

The Web Client Automation Server is implemented as a small COM object resident within the Web browser (IE 5.0 or greater). The Web Client Automation Server is supported with the High Interactive client only. When accessing the Web Client Automation Server, Siebel Web Client must be running.

To enable the Web Client Automation Server, make sure that the `EnableWebClientAutomation` parameter is set to TRUE in the [SWE] section of the application's configuration file. With this parameter set to TRUE, a small ActiveX Control downloads to the desktop and the SiebelHTMLApplication process starts. This process terminates when the Siebel Web Client is gracefully terminated. You may need to modify the ActiveX controls and plug-ins security settings in the Browser to use the Web Client Automation Server.

Figure 8 shows how external applications can invoke business services and manipulate property sets in the Siebel Web Client Automation Server.

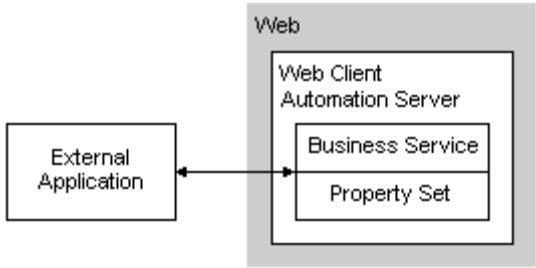


Figure 8. Siebel Web Client Automation Server

Siebel Mobile/Dedicated Web Client Automation Server

When accessing the Mobile/Dedicated Web Client Automation Server, Siebel Mobile Web Client must be running. Figure 9 shows how the Siebel Mobile/Dedicated Web Client Automation Server is used by external applications to control the Siebel application.

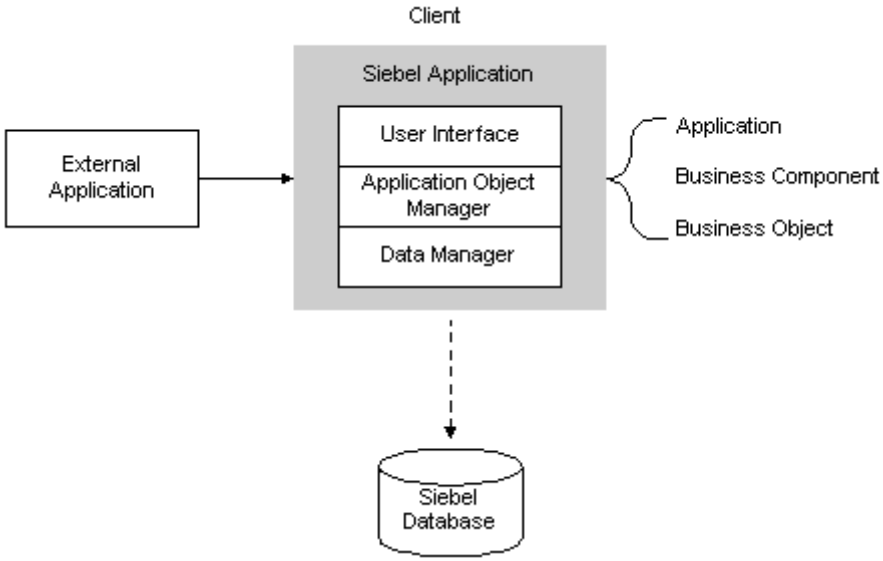


Figure 9. Siebel Mobile/Dedicated Web Client Automation Server

Siebel Java Interfaces

The Siebel Java Data Bean allows external applications to access Siebel objects without having to display the Siebel user interface. These objects are made available through the Siebel Java Data Bean, which can be used by an external application, component, or Java applet. The Java Data Bean provides functional access to the Siebel applications for both reading and writing data. The set of interfaces exposed through this interface is similar to that exposed by the Siebel COM Data Control.

Any external application that uses the Siebel Java Data Bean connects and communicates with a Siebel Application Object Manager. The Siebel Application Object Manager, which could be running on a remote Siebel Server, is a multithreaded, multiprocess application server that hosts Siebel objects and supports session-based connections by clients. The Siebel Application Object Manager specified in the connect string must be running on the specified Siebel Server.

Using the Siebel Java Data Bean with Multiple Threads

Multiple threads of a single process should not access a common instance of the Java Data Bean. If a process with multiple threads wants to use the Data Bean, each thread must create its own instance of it.

For the same reasons, you should not reuse instances of any other objects exposed by the Java Data Bean (SiebelBusObject, SiebelBusComp, SiebelService, and SiebelPropertySet) across multiple threads of the same process.

CAUTION: You should create one instance of the Siebel Java Data Bean for each thread that wishes to use it. Data Bean Objects obtained by one thread should not be shared among multiple threads.

Siebel CORBA Interfaces

Siebel CORBA interfaces allow external applications to access Siebel business objects through CORBA using Inprise's Visibroker or Iona's Orbix ORB. (See the system requirements and supported platforms documentation for information on the correct version to use.) The set of interfaces exposed through CORBA is similar to the Siebel COM Data Control and Siebel COM Data Server interfaces. External applications access the Siebel business objects through CORBA. This interface can be used by any external application that accesses Siebel applications for both reading and writing data, without having to display Siebel’s user interface.

For external applications to communicate with Siebel applications, you must install the Siebel CORBA Object Manager. A typical setup for a CORBA client accessing the Siebel CORBA Interfaces is shown in Figure 10.

NOTE: Siebel CORBA Object Manager (SCOM) is a stand-alone product and cannot be administered with Siebel Server Manager.

The name of the Siebel Interface Definition Language (IDL) file is scorba.idl. It can be found in the IDL subdirectory in the `CORBAOM` directory.

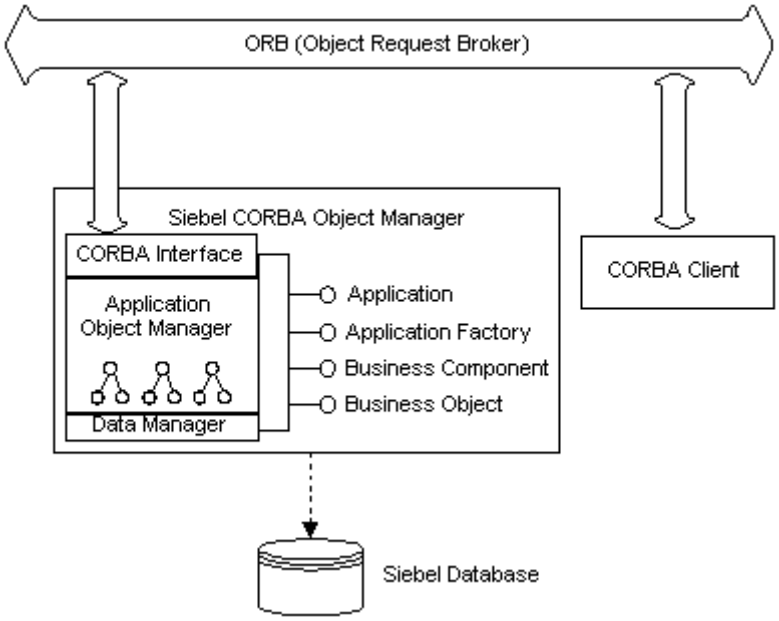


Figure 10. The CORBA Client Accessing the Siebel CORBA Object Manager

Built-in Scripting

You can access Siebel methods and events from within the Siebel application through Siebel VB or Siebel eScript. Both languages are procedural programming languages for writing custom extensions that access and control Siebel objects through the Siebel object interfaces.

Usage Evaluation Matrix

Use Table 7 to determine which types of Siebel object interface to use.

Table 7. Usage Evaluation

Usage	Web Client Automation Server	Mobile/ Dedicated Web Client Automation Server	Siebel COM Data Control	Siebel COM Data Server	Siebel Java Data Bean	Siebel CORBA Interfaces
Control Siebel user interface from your external application	X	X				
Access Siebel business objects without Siebel user interface			X	X	X	X
Objects execute on a Siebel Server	X		X		X	
Execute on the client side in mobile environments		X		X		
Execute on the Siebel CORBA Object Manager						X

Exposed Object Types

Siebel object interfaces provide access to Siebel business objects. See the following sections for a discussion of each exposed object type:

- "Application Object Type" on page 62

- "Business Object Object Type" on page 62
- "Business Component Object Type" on page 63
- "Business Service Object Type" on page 63
- "Applet Object Type" on page 64
- "Property Set Object Type" on page 65
- "User Interface Control Object Type" on page 65
- "Application Factory Object Type" on page 65

There are additional object types used in Siebel eBusiness applications, including specialized types derived from the base object types. However, object types not specifically discussed here are not exposed in the Siebel object interfaces and references to them may not be passed to external DLLs, such as a Microsoft Visual Basic COM DLL.

NOTE: Interfaces may be subject to change.

Application Object Type

The application object represents the Siebel application that is currently active and is an instance of the Application object type. An application object is created when a user session starts. This object contains the properties and events that interact with Siebel software as a whole. An instance of a Siebel application always has exactly one application object.

Business Object Object Type

Business objects are customizable, object-oriented building blocks of Siebel applications. Business objects define the relationships between different business component objects (BusComps) and contain semantic information about, for example, sales, marketing, and service-related entities.

A Siebel business object groups one or more business components into a logical unit of information. Examples of Siebel business objects include Opportunity, Quote, Campaign, and Forecast. An opportunity business object may consist of opportunity, contact, and product business components. The opportunity business component dictates the information of the other business components in a parent-child relationship.

Business Component Object Type

A business component defines the structure, the behavior, and the information displayed by a particular subject such as a product, contact, or account. Siebel business components are logical abstractions of one or more database tables. The information stored in a business component is usually specific to a particular subject and is typically not dependent on other business components. Business components can be used in one or more business objects.

Business component objects have associated data structured as records, they have properties, and they contain data units called *fields.* In the object interfaces, fields are accessed through business components. The business component object supports getting and setting field values, moving backward and forward through data in a business component object, and filtering changes to data it manages. This object type is available to every interface.

Business Service Object Type

Business service objects are objects that can be used to implement reusable business logic within the Object Manager. They include:

- Built-in business services, which are defined in Siebel Tools and stored in the repository.
- Run-time business services, which are defined in the run-time client and stored in the application database.

There are two types of built-in business services:

- Standard, which are based on the class CSSService and can be scripted or modified.
- Specialized, which are based on a specialized C++ class. Those specialized services whose behavior has been documented can be scripted.

Using business services, you can configure stand-alone “objects” or “modules” with both properties and scripts (written in VB or eScript). Business Services may be used for generic code libraries that can be called from any other scripts.

Built-in services cannot be modified at run time, and they cannot be overridden by run-time scripts.

User-created services can be created by adding a new record to the Business Service list applet in Siebel Tools. They can also be defined by administrators at run time by using views in the Siebel client. They can have whatever properties are needed to accomplish a particular task. They can be called either from scripts or from object interfaces.

Because they are reusable and can be set to persist throughout a session, business service objects can be used to simulate global procedures.

Applet Object Type

Because applet objects are part of the user interface, they are not accessible when using the Siebel object interfaces through the Siebel COM Data Server, Siebel COM Data Control, Siebel Web Client Automation Server, Siebel Mobile/Dedicated Web Client Automation Server, Siebel Java Data Bean, and the Siebel CORBA Object Manager.

An applet object represents an applet that is rendered by the Siebel Web Engine. It exists only as a scriptable object, and is accessed by using the Edit Server Scripts or Edit Browser Scripts command on the selected applet. Applet objects are accessible through Siebel VB and Siebel eScript in Server Scripts, and through Browser JavaScript in Browser Scripts. Some Applet Events, such as WebApplet_ShowControl and WebApplet_ShowListColumn, do not execute if the client is running in High Interactivity mode.

To add a Browser or Server script to an applet in Siebel Tools

1. In the Explorer window, choose the Applet object type.
2. In the right pane, locate the object to which you want to add a script.
3. Make sure that the project containing the applet is locked.
4. Right-click the item and select Edit Server Scripts or Edit Browser Scripts.

Property Set Object Type

Property set objects are collections of properties, which can be used for storing data. They may have child property sets assigned to them to form a hierarchical data structure. Property sets are used primarily for inputs and outputs to business services.

User Interface Control Object Type

A user interface control object, or a *control*, is a visual object with which the user can directly interact, such as a button or text box. Control objects have properties that can be accessed by Siebel Browser Script. Because control objects are part of the user interface, they are not accessible through the Siebel COM Data Server, Siebel COM Data Control, Mobile/Dedicated Web Client Automation Server, Web Client Automation Server, Siebel Java Data Bean, and the Siebel CORBA Object Manager.

Controls are the visible building blocks of applets. Each control is unique and exists only in a single applet. Only controls on the active (currently visible) applet are available to Siebel Browser Script. Each control has a unique name within its containing applet, but control names need not be unique across applets.

The control object supports getting and setting values and customized behavior when used in conjunction with Siebel Browser Script.

Application Factory Object Type

The Siebel Application Factory object is the externally visible object. It is supported only in Siebel CORBA Object Manager Interfaces. To access the other objects, a bind to the Siebel Application Factory object has to be made.

Summary of Exposed Object Types

Table 8 summarizes the names and types of objects exposed.

Table 8. Exposed Object Types for Each Siebel Object Interface

Object Type	Server Script	Browser Script	Siebel Web Client Automation Server	Siebel Mobile/ Dedicated Web Client Automation Server	Siebel COM Data Control	Siebel COM Data Server	Siebel Java Data Bean	Siebel CORBA Object Manager
Applet	X	X						
Application	X	X	X	X	X	X	X	X
Application Factory								X
Business Component	X	X		X	X	X	X	X
Business Object	X	X		X	X	X	X	X
Business Service	X	X	X	X	X	X	X	X
Property Set	X	X	X	X	X	X	X	X
Control		X						

Siebel Object Interface Method Syntax

The following conventions are used in this guide to describe methods, arguments, and return values:

Syntax *ObjectType.MethodName*(*arg1*[, *arg2, …, argn*])

Argument	Description
arg1	Description of *arg1*
arg2	Description of *arg2*
.	.
.	.
argn	Description of *argn*

Returns Description of the value returned by the method, if any.

The following conventions are used in the syntax diagram:

- *ObjectType* is the object type, for example BusComp (business component), for which the method is defined.
- *MethodName* is the name of the method that is being invoked. A method can be a subroutine that does not return a value, such as SetViewMode, or a function that returns a value, such as GetFieldValue.
- *arg1, arg2* can be a string, constant, integer, or object. If a method returns a value, the arguments must be enclosed in parentheses in Siebel VB. In Siebel eScript, enclose arguments in parentheses, even if they do not return a value.
- Brackets [ ] indicate an optional parameter. In the description of the argument, the default value for the optional parameter is indicated.

If a method does not return a value or if you are using it in a manner that does not return a value, then the arguments should not be enclosed in parentheses in Siebel VB.

When the using the COM Data Server, an additional argument, errCode, is always required as the last argument.

Usage Syntax The usage syntax for a method may differ between Server Script and COM, as described in the text that follows. The description uses the following terms in addition to the ones defined previously:

- `ObjectReference` is a variable name of type ObjectType that points to the object on which the method is invoked.

  NOTE: You do not need to explicitly specify an ObjectReference when you invoke a method on an object inside its event handler.

- `returnValue` is the value, if any, that is returned by the method. Some methods, such as GetBusComp, return an object of the type business component. Other methods return strings or integers.

Siebel VB

If there is a return value,

```
returnValue = ObjectReference.MethodName(arg1, arg2, ..., argn)
```

If there are no arguments,

```
returnValue = ObjectReference.MethodName
```

If there is no return value,

```
ObjectReference.MethodName arg1, arg2, ..., argn
```

Examples

```
acctName = acctBC.GetFieldValue("Name")
```

```
acctBC.SetViewMode AllView
```

Siebel eScript

If there is a return value,

```
returnValue = ObjectReference.MethodName(arg1, arg2, ..., argn);
```

If there are no arguments,

```
returnValue = ObjectReference.MethodName();
```

If there is no return value,

```
ObjectReference.MethodName(arg1, arg2, ..., argn);
```

Examples

```
acctName = acctBC.GetFieldValue("Name");
```

```
acctBC.SetViewMode(AllView);
```

Using parenthesis ( ) when none are required, or failing to use them when they are required, generates a Type Mismatch (error code 13) message. Another cause of this error code is using an incorrect quantity of arguments.

COM

The usage depends on the language being used to call the COM Interfaces. For Microsoft Visual Basic and equivalent languages, the usage is similar to that of Siebel VB, except that an error code is passed as the final argument in the case of the COM Data Control.

Getting Started with the Siebel Object Interfaces

The following sections contain directions for connecting to the COM Servers, COM Controls, or ORBs:

- "Accessing Siebel COM Interfaces" on page 70
- "Accessing the Siebel Web Client Automation Server" on page 71
- "Accessing the Siebel Mobile/Dedicated Web Client Automation Server" on page 72
- "Instantiating the Siebel COM Data Server" on page 75
- "Instantiating the Siebel COM Data Control" on page 78
- "Java Data Bean" on page 81
- "Accessing Siebel CORBA Interfaces" on page 88
- "Initializing ORB and Binding Siebel App Factory (Orbix)" on page 88
- "Initializing ORB and Binding Siebel App Factory (Visibroker)" on page 90

Accessing Siebel COM Interfaces

To use the Siebel COM interfaces, you must set the EnableOLEAutomation flag in the CFG file to TRUE. For Siebel Interface methods through COM, use the object browser of your COM programming tool to determine the correct method syntax. Figure 11 displays an example of an object browser in Microsoft Visual Basic 5.0.

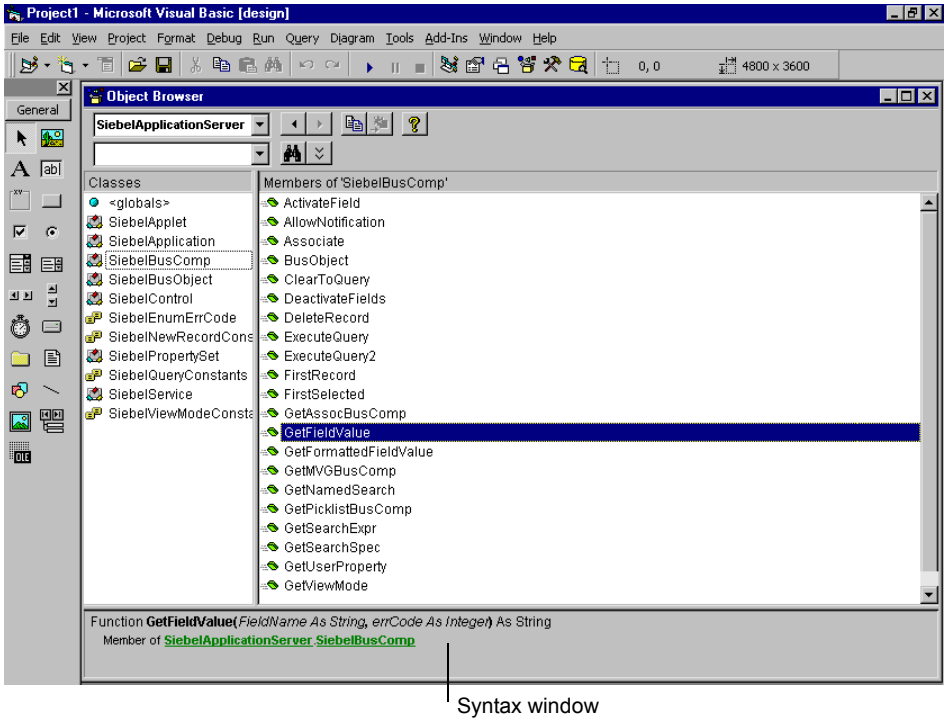


Figure 11. Determining Correct COM Syntax in Microsoft Visual Basic

Accessing the Siebel Web Client Automation Server

The Web Client Automation Server allows external applications to invoke business services and manipulate property sets. The Web Client Automation Server is implemented as a small COM object resident within the Web browser (IE 5.0 or greater). The Web Client Automation Server can be used with the Web client and the Mobile/Dedicated Web client. The Web Client Automation Server is supported with the High Interactivity mode only. If you use Windows 2000 servers, make sure to configure the security settings as described in the *Security Settings* topic in the *Other Requirements for Employee Applications* section in *Siebel System Requirements and Supported Platforms* (available on SupportWeb).

To set up Microsoft Visual Basic to access the Siebel Web Client Automation Server

1. Start Microsoft Visual Basic.
2. Select Standard EXE.
3. Choose Project > References.
4. In the list box, highlight and check the SiebelHTML 1.0 Type Library.

The following example shows how to use Microsoft Visual Basic 6.0 with the Siebel Web Client Automation Server.

```
Private Sub Command1_Click()
'Siebel Application Object
Dim siebApp As SiebelHTMLApplication
Dim siebSvcs As SiebelService
Dim siebPropSet As SiebelPropertySet
Dim bool As Boolean
Dim errCode As Integer
Dim errText As String
Dim connStr As String
Dim lng As String
'Create The SiebelHTML Object
Set siebApp = CreateObject("SiebelHTML.SiebelHTMLApplication.1")


If Not siebApp Is Nothing Then
```

```
   'Create A New Property Set
   Set siebPropSet = siebApp.NewPropertySet
   If Not siebPropSet Is Nothing Then
      Set siebPropSet = Nothing
   Else
      errCode = siebApp.GetLastErrCode
      errText = siebApp.GetLastErrText
      TheApplication().RaiseErrorText("Property Set Creation
failed: " & errCode & "::" & errText);
   End If


   'Get A Siebel Service
   Set siebSvcs = siebApp.GetService("Workflow Process Manager")
   If Not siebSvcs Is Nothing Then
      Set siebSvcs = Nothing
   Else
      errCode = siebApp.GetLastErrCode
      errText = siebApp.GetLastErrText
      TheApplication().RaiseErrorText("Could not Get Siebel
Service: " & errCode & "::" & errText);
   End If

Set siebApp = Nothing
End If
End Sub
```

Accessing the Siebel Mobile/Dedicated Web Client Automation Server

The Siebel Mobile/Dedicated Web Client Automation Server accesses the server object instantiated by the Siebel eBusiness Application. Once you have this object, you can obtain other Siebel objects and execute Siebel object interface methods through those objects. Calls made to the Siebel Mobile/Dedicated Web Client Automation Server are out of process. If you create a DLL that is run in process with the Siebel application, the calls made from the DLL to the Siebel Mobile/Dedicated Web Client Automation Server are still out of process.

The mechanism for instantiating COM servers depends on the programming tool or language being used.

If you use Microsoft Visual Basic 5.0 or later, the required support file must be in the same directory as the CFG file you are using for your Siebel application, or the Mobile/Dedicated Web Client Automation Server does not work. Take the following steps to make sure that you are referring to the correct library.

To set up Microsoft Visual Basic to access the Siebel Mobile/Dedicated Web Client Automation Server

1. Start Microsoft Visual Basic.
2. Select Standard EXE.
3. Choose Project > References.
4. In the list box, highlight (check) Siebel Mobile Web Client Automation Server. Near the bottom of the dialog box, note the directory in which the file Siebel.exe resides.

The following examples show how to use Microsoft Visual Basic 6.0 to interface with Siebel Mobile/Dedicated Web Client Automation Server.

The following is sample code for the Siebel Mobile/Dedicated Web Client Automation Server:

```
Private Sub Command1_Click()
'Siebel Application Object
Dim siebApp As SiebelWebApplication
Dim siebBusObj As SiebelBusObject
Dim siebBusComp As SiebelBusComp
Dim siebSvcs As SiebelService
Dim siebPropSet As SiebelPropertySet
Dim bool As Boolean
Dim errCode As Integer
Dim errText As String
Dim connStr As String
Dim lng As String
'Create The Siebel WebApplication Object
Set siebWebApp = CreateObject("TWSiebel.SiebelWebApplication.1")

If Not siebWebApp Is Nothing Then


'Create A Business Object
Set siebBusObj = siebWebApp.GetBusObject("Contact")
If Not siebBusObj Is Nothing Then
   'Create a Business Component
   Set siebBusComp = siebBusObj.GetBusComp("Contact")
```

```
Else
   errCode = siebWebApp.GetLastErrCode
   errText = siebWebApp.GetLastErrText
   TheApplication().RaiseErrorText("Business Object Creation
failed: " & errCode & "::" & errText);

End If



'Create A New Property Set
Set siebPropSet = siebWebApp.NewPropertySet
If Not siebPropSet Is Nothing Then
   Set siebPropSet = Nothing

Else

      errCode = siebWebApp.GetLastErrCode
      errText = siebWebApp.GetLastErrText
      TheApplication().RaiseErrorText("Property Set Creation
failed: " & errCode & "::" & errText);
End If

'Get A Siebel Service
Set siebSvcs = siebWebApp.GetService("Workflow Process Manager")
If Not siebSvcs Is Nothing Then
   Set siebSvcs = Nothing
Else
   errCode = siebWebApp.GetLastErrCode
   errText = siebWebApp.GetLastErrText
   TheApplication().RaiseErrorText("Could not Get Siebel Service:
" & errCode & "::" & errText);
End If

If Not siebBusComp Is Nothing Then
   Set siebBusComp = Nothing
End If

If Not siebBusObj Is Nothing Then
   Set siebBusObj = Nothing
End If

   Set siebWebApp = Nothing
End If

End Sub
```

Instantiating the Siebel COM Data Server

Because the Siebel COM Data Server acts without the regular Siebel eBusiness Application User Interface, you must use the Login method to set up your Data Server object. You cannot use methods that retrieve active Siebel objects, because there are no current active Siebel objects. You must instantiate your own Siebel objects. Calls made to the Siebel COM Data Server are in process.

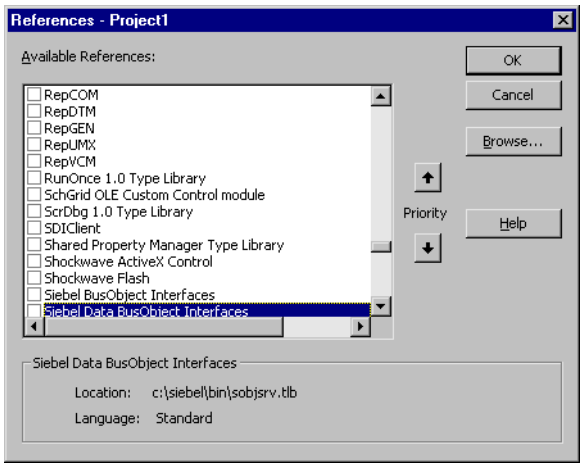
If you use Microsoft Visual Basic 5.0 or later, the required support file, `sobjsrv.tlb`, must be in the same directory as the CFG file you are using for your Siebel application, or the COM Data Server does not work. Take the following steps to make sure you are referring to the correct library.

NOTE: Do not run in the Microsoft VB Debug environment when communicating with the Siebel COM data server.

To set up Microsoft Visual Basic to access the Siebel COM Data Server

1. Start Microsoft Visual Basic.
2. Select Standard EXE.
3. Choose Project > References.

4 In the list box, select (but do not check) Siebel Data BusObject Interfaces. Near the bottom of the dialog box, note the directory in which the file sobjsrv.tlb resides, as shown in the following illustration.



5 Check the Siebel Data BusObject Interfaces entry and click OK.

The following is sample code for the Siebel COM Data Server. Make sure that the DataSource parameter in the CFG file is set to the database to which you want to connect.

NOTE: This code must be written and executed outside of Siebel Tools, for example in Microsoft Visual Basic.

```
Private Sub Command1_Click()
'Siebel Application Object
Dim siebApp As SiebelApplication
Dim siebBusObj As SiebelBusObject
Dim siebBusComp As SiebelBusComp
Dim siebSvcs As SiebelService
Dim siebPropSet As SiebelPropertySet
Dim bool As Boolean
Dim errCode As Integer
Dim errText As String
```

```
Dim connStr As String
Dim lng As String
Dim cfgLoc As String

ChDrive "D:"
ChDir "D:\Server\siebsrvr\bin "

'Create The COM Data Server Object
Set siebApp = CreateObject("SiebelDataServer.ApplicationObject")

If Not siebApp Is Nothing Then

'''COM Data Server
cfgLoc = " D:\Server\siebsrvr\bin \ENU\siebel.cfg,ServerDataSrc"
siebApp.LoadObjects cfgLoc, errCode
If errCode = 0 Then
   'Log Into the Siebel Server
   siebApp.Login "username", "password", errCode
   If errCode = 0 Then
      'Creat A Business Object
      Set siebBusObj = siebApp.GetBusObject("Contact", errCode)
      If errCode = 0 Then
         'Create a Business Component
         Set siebBusComp = siebBusObj.GetBusComp("Contact")
   Else
      errText = siebApp.GetLastErrText
      TheApplication().RaiseErrorText("Business Object Creation
failed: " & errCode & "::" & errText);
   End If

   'Create A New Property Set
   Set siebPropSet = siebApp.NewPropertySet(errCode)
   If errCode = 0 Then
      Set siebPropSet = Nothing
   Else
      errText = siebApp.GetLastErrText
      TheApplication().RaiseErrorText("Property Set Creation
failed: " & errCode & "::" & errText);
   End If

   'Get A Siebel Service
   Set siebSvcs = siebApp.GetService("Workflow Process Manager",
errCode)
   If Not siebSvcs Is Nothing Then
      Set siebSvcs = Nothing
   Else
      errText = siebApp.GetLastErrText
```

```
      TheApplication().RaiseErrorText("Could not Get Siebel
Service: " & errCode & "::" & errText);
   End If

   If Not siebBusComp Is Nothing Then
      Set siebBusComp = Nothing
   End If
   If Not siebBusObj Is Nothing Then
      Set siebBusObj = Nothing
   End If
Else
     errText = siebApp.GetLastErrText
     TheApplication().RaiseErrorText("Login Failed: " & errCode &
"::" & errText);
   End If
Else
   errText = siebApp.GetLastErrText
   TheApplication().RaiseErrorText("Load Objects Failed: " &
errCode & "::" & errText);
End If

Set siebApp = Nothing

End If

End Sub
```

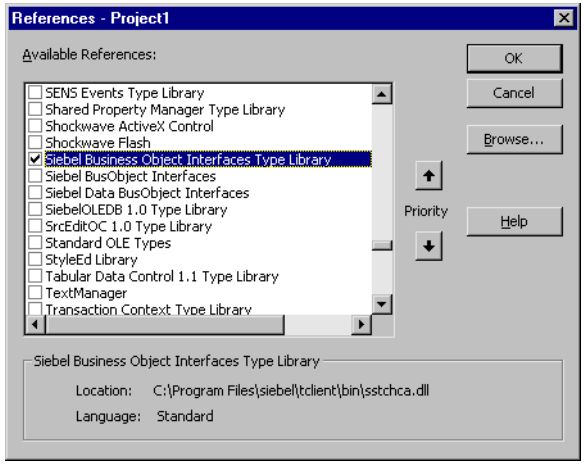
Instantiating the Siebel COM Data Control

To use Siebel Interface methods with the Siebel COM Data Control, use the object browser of your Siebel COM Data Control programming tool to determine the correct method syntax.

To set up Microsoft Visual Basic to access the Siebel COM Data Control Interface

1. Be sure you have installed the Siebel COM Data Control. Read "Installing Siebel Object Interfaces" on page 53.
2. Start Microsoft Visual Basic.
3. Select Standard EXE.
4. Choose Project > References.

5 In the list box, highlight (but do not check) Siebel BusObject Interfaces Type Library. Near the bottom of the dialog box, note the directory in which the file `sstchca.dll` resides, as shown in the following illustration.



6 Open the Object Browser to verify that you can see the Siebel objects.

To instantiate and use the Siebel COM Data Control, you must use the CreateObject and Login methods. You cannot use methods that retrieve active Siebel objects, because there are no current active Siebel objects. You must instantiate your own Siebel objects. Calls made to the Siebel COM Data Control are also in-process.

The following is sample code for the Siebel COM Data Control.

```
Sub CreateDataControl()
Dim errCode As Integer
Set SiebelApplication =
CreateObject("SiebelDataControl.SiebelDataControl.1")
SiebelApplication.Login "host=""siebel//:host/EnterpriseServer/
AppObjMgr/SiebelServer""", "CCONWAY","CCONWAY"
errCode = SiebelApplication.GetLastErrCode()
If errCode <> 0 Then
   ErrText = SiebelApplication.GetLastErrText
   TheApplication().RaiseErrorText(ErrText);
   Exit Sub
End If
```

```
set OpptyB0 = SiebelApplication.GetBusObject("Opportunity",errCode)
set OpptyBC = OpptyBO.GetBusComp("Opportunity", errCode)
End Sub
```

See Table 23 for values to substitute for the placeholders in the login string.

Here is sample code for instantiating the COM Data Control from a server-side ASP script.

NOTE: The symbols <% and %> are used within HTML script to set off an ASP script.

```
<%

   Dim SiebelApplication, BO, BC, ConnStr, logstat
   Dim strLastName, strFirstName, errCode, errText

      Set SiebelApplication =
CreateObject("SiebelDataControl.SiebelDataControl.1")

   ' Test to see if object is created
   If IsObject(SiebelApplication) = False then
      Response.Write "Unable to initiate Siebel Session.
   Else
      connStr = "host=" & Chr(34) & "siebel.tcpip.none.none://
gateway:2320/EntServer/ObjMgr/siebSrvr" & Chr(34) & " lang=" &
Chr(34) & "<lang>" & Chr(34)
      logstat = SiebelApplication.Login ConnStr, "SADMIN",
"SADMIN"

   response.write("Login Status: " & logstat)
   Set BO = SiebelApplication.GetBusObject("Employee")
   Set BC = BO.GetBusComp("Employee")
End If

%>
```

For more information on instantiating the Siebel COM Data Control, read "Connect String" on page 112.

Java Data Bean

Siebel Java Data Bean provides users with a native Java interface to access Siebel Object Manager. It provides functional access to the Siebel applications for both reading and writing data. Siebel Data Bean is a set of Java libraries built using JDK 1.3.1_03. Users can incorporate these libraries to build Java Applications, Applets, Servlets, JSPs, or Enterprise Java Beans into their Java-based applications.

NOTE: The Siebel JAR files (`SiebelJI.jar, SiebelJI_common.jar` and `SiebelJI_<lang>`) need to be added to the CLASSPATH prior to compilation or execution.

Supported Platforms and JDKs

Siebel Systems supports the use of the platforms and JDK versions specified in the system requirements and supported platforms documentation for your Siebel application.

Instantiating the Java Data Bean

To instantiate and use the Siebel Java Data Bean, you must instantiate a new SiebelDataBean Java object and call its login method. You cannot use methods that retrieve active Siebel objects, because there are no current active Siebel objects. You must instantiate your own Siebel objects.

The following is the sample code for the Siebel Java Data Bean.

```
import com.siebel.data.*;
import com.siebel.data.SiebelException;

public class DataBeanDemo
{
   private SiebelDataBean m_dataBean = null;
   private SiebelBusObject   m_busObject = null;
   private SiebelBusComp       m_busComp = null;

   public static void main(String[] args)
   {
      DataBeanDemo demo = new DataBeanDemo();
   }
```

```
   public DataBeanDemo()
   {
      try
      {
         // instantiate the Siebel Data Bean
         m_dataBean = new SiebelDataBean();

         // login to the server
         m_dataBean.login("Siebel://gatewayserver/
enterpriseServer/ObjMgr/SiebelServer", CCONWAY, CCONWAY,"enu");

         // get the business object
         m_busObject = m_dataBean.getBusObject("Opportunity");

         // get the business component
         m_busComp = m_busObject.getBusComp("Opportunity");

         // logoff
         m_dataBean.logoff();
      }

      catch (SiebelException e)
      {
         System.out.println(e.getErrorMessage());
      }
   }
}
```

Java Data Bean and the siebel.properties File

The siebel.properties file, which is located in your classpath, can be used to provide default parameters for client applications connecting to Siebel applications using the Java Data Bean.

Table 9 shows the properties in the siebel.properties file.

Table 9. Properties in the siebel.properties File

Property Type	Property	Description
Siebel Connection Manager Connection properties	siebel.conmgr.txtimeout	Indicates the transaction timeout (in seconds). Defaulted to 2700 = 45m.
	siebel.conmgr.poolsize	Indicates the connection pool size. Connection pool maintains a set of connections to a specific server process. Defaulted to 2. Max connection pool size is 500.
	siebel.conmgr.sesstimeout	Indicates the transaction timeout (in seconds) on the client side. Defaulted to 600 = 10m.
	siebel.conmgr.retry	Indicates the number of open session retries. Defaulted to 3.
	siebel.conmgr.jce	Indicates the usage of Java Cryptography Extension. 1 for jce usage and 0 for no usage.
Siebel Generated code for JCA/JDB properties	siebel.connection.string	Specifies the Siebel connection string.
	siebel.user.name	Specifies the user name to be used for logging in to Object Manager.
	siebel.user.password	Specifies the password to be used for logging in to Object Manager.
	siebel.user.language	Specifies the user's preferred language.
	siebel.user.encrypted	Specifies whether the username and password is encrypted.
	siebel.jdb.classname	Specifies the default JDB classname
Java System Properties	file.encoding	Indicates the code page on the client side. For example, cp1252, utf8, unicodeBig, cp942.

NOTE: Java System Properties are System Properties, not Siebel Properties.

Here is a sample Siebel.properties file:

```
siebel.connection.string = siebel.tcpip.rsa.none://
test.siebel.com/siebel/sseobjmgr_enu/test

siebel.user.name          = User1

siebel.user.password      = password

siebel.user.language      = enu

siebel.user.encrypted     = false

siebel.conmgr.txtimeout   = 3600

siebel.conmgr.poolsize    = 5

siebel.conmgr.sesstimeout = 300000

siebel.conmgr.retry       = 5

siebel.conmgr.jce         = 1
```

Java Data Bean and Codepage Support

For the client and server to communicate correctly, the codepage of the Siebel server and client must be the same. If the client and server default codepages cannot be the same, you can alter the client codepage by setting the system property `file.encoding` to the proper codepage. You can set the system property for the entire JVM (for example, java -Dfile.encoding = ascii < java_application > on the command line or with the use of the environment variable; reference your particular JVM for details) or for the given Java component by adding the following line to your Java component: `System.setProperty("file.encoding", CodePageValue);`.

Table 10 lists codepage mappings for JDB.

Table 10. Codepage Mappings for Java Data Bean

Java Value	Siebel Value
ascii	1
cp1252	1252
iso8859_1	1252
iso8859-1	1252
unicodebig	1201
unicodelittle	1200
utf8	65001
big5	950
cp942	932
cp942c	932
cp943	932
cp943c	932
cp949	949
cp949c	949
cp950	950
cp1250	1250
cp1251	1251
cp1253	1253
cp1254	1254
cp1255	1255
cp1256	1256
cp1257	1257

Table 10. Codepage Mappings for Java Data Bean

Java Value	Siebel Value
cp1258	1258
gbk	936
ms874	874
ms932	932
ms936	936
ms949	949
ms950	950
sjis	932
tis620	874

Encrypting Communication Between JDB and Siebel Server

Siebel eBusiness Applications 7.5 supports the encryption of communication between the Java Data Bean (JDB) and the Siebel Server. Preconfigured, it is possible to encrypt communication between the JDB and the Siebel Server using RSA's encryption libraries. For more information on supported platforms, see the system requirements and supported platforms documentation for your Siebel eBusiness Applications software.

To enable encryption support between the Siebel Server and a component built using the Java Data Bean

1. Enable encryption in the corresponding Object Manager Server Component. Please refer to *Siebel Server Administration Guide*for details on how to enable encryption within an Object Manager Server Component.

2. Set the encryption parameter of the connect string in the Java Data Bean to `rsa`, which enables encryption support. For example, `siebel.tcpip.rsa.none://<gateway>/<enterprise>/<ObjMgr>/<SiebSrvr>`

   After completing the two previous steps, communications between the Java Data Bean and the Siebel Server is encrypted.

To support encryption on platforms not supported by the RSA libraries, Siebel Systems supports the Java Cryptography Extension (JCE) v1.2.1 specification. JCE is designed so that other qualified cryptography libraries can be used as service providers.

To enable JCE support

1. Download and install the JCE v1.2.1 software, policy files and documentation. Please refer to `http://java.sun.com/products/jce/index-121.html` for additional information on obtaining, installing and configuring your JVM for use with JCE. Please note that the Java Data Bean only supports static specification of JCE providers.
2. Modify the `java.security` file to specify your provider of choice and make sure that the necessary provider JAR files are included in the CLASSPATH.
3. Set the `siebel.conmgr.jce` property in the `siebel.properties file` to `1`.

   After completing the three previous steps, communications between the Java Data Bean and the Siebel Server is encrypted.

Login Errors

The Siebel Data Bean may return a login error including the following text.

```
Siebel Exception thrown invoking login Method. Code--1. Message-
Logon request 75 was abandoned after 2ms connection
```

The root cause of this error may be one of the following:

- OM or OM process down
- Hardware reset (OM hardware, router, switch, and so on)
- OS settings or OS networking issue
- Network failure
- NAT timeout

Accessing Siebel CORBA Interfaces

To access the CORBA Object Manager, EnableCORBA must be set to TRUE in your CFG file.

Siebel CORBA Interface syntax is similar to the Siebel ActiveX and Siebel COM Interface syntax.

For information on Siebel CORBA Interfaces and Unicode, read "CORBA Interfaces and Unicode" on page 395.

Initializing ORB and Binding Siebel App Factory (Orbix)

To initialize and start running a client CORBA Object Manager application, an initial reference to the Siebel Application Factory Object must be performed. To obtain an initial reference to the Application Factory Object, the client application must use the ORBIX Naming Service. The Naming Service is a standard CORBA service that provides the ability to bind an object reference to a name. The names are stored in a tree structure very similar to a file systems directory structure in the naming service, but rather than directories it has Naming Contexts and rather than files it has Object References. For example in the following name:

```
Siebel/CorbaOm/Objmgr1
```

Siebel and CorbaOm are naming contexts and Objmgr1 is an object reference. The '/' delimiting the hierarchy of naming contexts is not defined by the CORBA standard, but is convenient for representing a full path to an object reference.

After obtaining a reference to the Siebel Application Factory object, the Siebel Application Object can be created and initialized.

The following code sample shows how to obtain a reference to a Siebel Application Factory object and how to initialize a Siebel Application Object.

```
package SiebelExample;

import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import java.io.*;

public class SiebelClient
{
   public static void main (String args[])
```

```
{
   try {
      // Siebel references
      SiebelAppFactory    appFact     = null;
      SiebelApplication   siebelApp   = null;
      SiebelBusObject     busObj      = null;
      SiebelBusComp       busComp     = null;
      short                viewMode    = 3;

      // -b parameter was CORBAOM, -x parameter was SCOM
      String              nsName       = "SCOM/CORBAOM";

      // CORBA references
      org.omg.CosNaming.NamingContextExt    root_cxt;
      org.omg.CORBA.Object                  obj;
      org.omg.CORBA.ORB                      orb;

   // Initialize the ORB
   orb = ORB.init(args, null);

   // obtain reference to initial naming context
   obj = orb.resolve_initial_references("NameService");
   root_cxt =
org.omg.CosNaming.NamingContextExtHelper.narrow(obj);

   if (root_cxt != null) {
      org.omg.CORBA.Object appFact_obj = null;
      appFact_obj = root_cxt.resolve_str(nsName);
      appFact = SiebelAppFactoryHelper.narrow(appFact_obj);

      if (appFact != null) {
         siebelApp = appFact.CreateSiebelAppObjectW();

         if (siebelApp == null
            System.out.println("siebel app is null");
         else
         {   // Simple example showing how to login, query a
            // business component, logoff, and release the
            // application object
            siebelApp.LoginW("SADMIN", "SADMIN");
            busObj = siebelApp.GetBusObjectW("Account");
            busComp = busObj.GetBusCompW("Account");
            busComp.ActivateFieldW("Name");
            busComp.ActivateFieldW("Location");
            busComp.ActivateFieldW("Type");
            busComp.ClearToQueryW();
            busComp.ViewMode(viewMode);
            busComp.ExecuteQueryW(false);
```

```
                    if (busComp.FirstRecordW()) {
                       System.out.println("Record found: " +
                          busComp.GetFieldValueW("Name") + " / " +
                          busComp.GetFieldValueW("Location") + " / " +
                          busComp.GetFieldValueW("Type"));
                    }

                    siebelApp.LogoffW();
                    appFact.ReleaseW(siebelApp);
                 }
              }
           }
           }
           catch (SiebelExceptionW se) {
              System.out.println("Siebel exception: " + se.desc);
           }
           catch (SystemException ce) {
              System.out.println("CORBA exception: " + ce.toString());
           }
           catch(Exception ex) {
           System.out.println("Unexpected CORBA exception: " +
              ex.toString());
           }
        }
     }
```

Initializing ORB and Binding Siebel App Factory (Visibroker)

When using Inprise Visibroker ORB, the client application must bind to the Siebel Application Factory object as the Orbix clients do. However, the name of the host machine need not be passed on as an additional argument to the bind call.

NOTE: For the correct version of Visibroker to use, read the system requirements and supported platform documentation for your Siebel eBusiness application.

The sample code that follows shows how to get a reference to the application factory and create an application object.

```
SiebelAppFactory_var appFact;
SiebelApplication_var appObj;
char objectRootName[OBJECTNAMELEN];
```

```
try {
   // Initialize the ORB.
   orb = CORBA::ORB_init(argc, argv);

   sprintf(objectRootName, "SiebelObjectFactory";
   // bind to the Factory object in the server.

  appFact = SiebelAppFactory::_bind(objectRootName);

} catch(const CORBA::Exception& excep) {
   // Standard exception.
   cerr << "Error id : " << seblExcep.id << endl;
   cerr << "Error text : " << endl;
   cerr << seblExcep.desc << endl;

   return;
}

try {
   appObj = appFact->CreateSiebelAppObject();
   cout << "Created Siebel object" << endl;
} catch (const SiebelException& seblExcep) {
   cerr << "Error id : " << seblExcep.id << endl;
   cerr << "Error text : " << endl;
   cerr << seblExcep.desc << endl;
   return;
}
```

After the bind and creation of the Siebel Application Object, the application must log on to the Siebel Application Object Manager to access the Siebel business objects. At the end of the session, the client can log off the session using the Logoff method.

The sample code that follows illustrates the invocation of the Login and GetBusObject methods to communicate with the Siebel Application Object Manager.

```
try {
   // call the login method to access the Siebel application
   appObj->Login("CCONWAY", "CCONWAY");
   // Get the reference to the Accounts business object
   pBusObj = appObj->GetBusObject("Account");
} catch (SiebelException& excep) {
```

```
    cerr << "Error code : " << excep.id << endl;
    cerr << "Error text : " << excep.desc << endl;
    return;
}
```

Siebel Object Interface Methods

Several groups of methods are available to Siebel object interface programmers. They are organized according to functional capabilities in the following sections:

- **Locating objects.** These are methods that allow the user to locate instances of objects so that they can be manipulated by other methods.
- **Accessing business components.** These are methods that provide the ability to access and modify data within Siebel applications.
- **Navigation.** These are methods that provide a way to control the flow of the application as it is presented to the user by explicitly setting the focus of the application to the desired view, applet, or control. These methods are useful only when accessing the Siebel object interfaces from Siebel VB and when accessing Siebel as a Mobile/Dedicated Web Client Automation Server. When Siebel is accessed through the COM Data Control, COM Data Server, Java Data Bean, or CORBA Application Object Manager, no Siebel user interface is present.
- **Manipulating controls.** These are the methods that get or set the value of a control. These methods are useful only when accessing controls from Browser Script.
- **Global state properties and functions.** These are methods that get information on the current state.
- **User interaction.** These are methods that provide user interface elements similar to those in standard Windows programs.
- **Creating a Siebel application in CORBA.** There is one method that is used to create a Siebel application object in CORBA.

See Also

"Locating Objects"
"Accessing Business Components" on page 94
"Navigation" on page 99
"User Interaction" on page 99
"Global State Properties and Functions" on page 100
"Creating a Siebel Application Object in CORBA" on page 100

Locating Objects

This set of methods allows the user to locate instances of objects within Siebel applications so they can be used by other methods. Active objects are instances of objects that currently have focus. The active control is the control that currently has the user interface focus. The active applet is the applet that contains the active control. The active business component is the business component associated with the active applet. When located, an object can be used or manipulated by Siebel object interfaces.

For locating objects, use the following methods.

- "ActiveBusObject" on page 153
- "ActiveMode" on page 133
- "ActiveViewName" on page 154
- "BusComp" on page 302
- "BusObject" on page 134
- "GetBusObject" on page 157
- "GetValue" on page 319
- "Name" on page 304
- "TheApplication" on page 329

Accessing Business Components

The Siebel business component object (BusComp) presents a two-dimensional grid of data values much like a table in a relational database. The named fields are analogous to columns in the database table, and the records are analogous to rows. Developers use business components to read data, manipulate it, and write it back into the Siebel database. Business components manage the complexities of multiple-table access to the database and access different types of databases.

Many methods are available to use on business components for getting and setting the values of their fields. Record operations can be performed programmatically by using business component access methods.

These operations invoke Siebel VB or Siebel eScript extension routines. For example, if you have created a Siebel VB or Siebel eScript script that is tied to the NewRecord event on a business component, the script is processed whenever NewRecord in that business component is processed, even if the NewRecord method was called by another Siebel VB or Siebel eScript script or was called from the Siebel object interfaces. Note that events are available only with Siebel VB or Siebel eScript.

Adding and Inserting Records

In the context of a many-to-many relationship, you can use Siebel VB or Siebel eScript to mimic either the Add New Record command, which associates a new child record, or the Insert Record command, which creates a new record in the child business component. To associate a new child record, use GetAssocBusComp and the Associate method. To create a new record in the child, use the NewRecord method in a child business component, or use GetMVGBusComp and the NewRecord method.

Committing Records to the Database

A commit is performed under the following circumstances:

- Explicitly by issuing BusComp.WriteRecord
- Navigating away from the current record by any of the following methods:
  - BusComp.NextRecord
  - BusComp.PreviousRecord

  - BusComp.FirstRecord
  - BusComp.DeleteRecord (DeleteRecord commits automatically, because it moves the cursor to another record.)
  - BusComp.LastRecord
- Closing a BusComp (Set BusComp = Nothing)

Scenarios for Business Components

The two scenarios that follow involve the use of Siebel scripts to work with business components.

The first example shows how to invoke methods on an existing business component when an event is triggered. In this example, the VB script is in the SetFieldValue event of a business component.

```
Sub BusComp_SetFieldValue (FieldName As String)
Dim desc As String
Dim newDesc As String

theApplication.TraceOn "c:\temp\trace.txt", "Allocation", "All"
If FieldName = "Type" Then

   newDesc = "Any valid string which contains the
       new description."
   desc = Me.GetFieldValue("Description")
   theApplication.Trace "The previous description is " & desc
   Me.SetFieldValue "Description", newDesc
   theApplication.Trace "The new description is " & newDesc

End If
theApplication.TraceOff

End Sub
```

The next example shows how to instantiate your own BusObject and BusComp. This example uses the PreSetFieldValue event of the Opportunity BusComp. If the Sales Stage is updated to “07 - Verbal Agreement,” a decision maker must be associated with the opportunity. Otherwise, it is reset to the previous value. The Contacts for the selected opportunity are searched to see if any vice president or president is associated with the opportunity.

The logical flow of instantiating your own BusComp object is as follows

1. GetBusComp
2. SetViewMode (optional, because the BusComp may already be in the correct mode)
3. ActivateField
4. ClearToQuery
5. SetSearchSpec or SetSearchExpr
6. ExecuteQuery

```
Function BusComp_PreSetFieldValue (FieldName As String,
FieldValue As String) As Integer

Dim RetValue As Integer
RetValue = ContinueOperation
Select Case FieldName
   Case "Sales Stage"
      If FieldValue = "08 - Negotiation" Then
         ' Do not allow the sales cycle to be changed to this value
         ' if the decision-maker is not a contact for the Oppty.
         ' Decision-maker defined as anyone with rank VP and above
         Dim oBusObj As BusObject
         Dim sRowId As String
         Dim iViewMode As Integer
         sRowId = GetFieldValue("Id")
         iViewMode = GetViewMode
         Set oBusObj = TheApplication.ActiveBusObject

         ' Because parent-child relationship is established when
         ' BusComps are instantiated from the same BusObject.
         ' The ContactBC has all contact records For the
         ' current Oppty record.
         Set ContactBC = oBusObj.GetBusComp("Contact")
         With ContactBC
            .ActivateField "Job Title"
            .ClearToQuery
            .SetSearchSpec "Job Title", "*VP*"
            .ExecuteQuery ForwardOnly
            If (.FirstRecord = 0) Then
            TheApplication.RaiseErrorText "Found a decision
maker"
            RetValue = CancelOperation
            Else
               RetVal = ContinueOperation
```

```
            End If
         End With
      End If
End Select
BusComp_PreSetFieldValue = RetValue
End Function
```

Methods for Accessing Business Components

To access business components, use the following methods.

- "ActivateMultipleFields" on page 199
- "Associate" on page 201
- "ClearToQuery" on page 203
- "DeactivateFields" on page 204
- "DeleteRecord" on page 205
- "ExecuteQuery" on page 206
- "ExecuteQuery2" on page 209
- "FirstRecord" on page 210
- "FirstSelected" on page 211
- "GetFieldValue" on page 214
- "GetFormattedFieldValue" on page 216
- "GetMultipleFieldValues" on page 220
- "GetMVGBusComp" on page 220
- "GetNamedSearch" on page 222
- "GetPicklistBusComp" on page 222
- "GetSearchExpr" on page 224
- "GetSearchSpec" on page 225
- "GetViewMode" on page 226

- "InvokeMethod" on page 228
- "LastRecord" on page 233
- "NewRecord" on page 234
- "NextRecord" on page 236
- "ParentBusComp" on page 237
- "Pick" on page 238
- "PreviousRecord" on page 239
- "RefineQuery" on page 241
- "SetFieldValue" on page 243
- "SetFormattedFieldValue" on page 245
- "SetMultipleFieldValues" on page 247
- "SetNamedSearch" on page 248
- "SetSearchExpr" on page 249
- "SetSearchSpec" on page 251
- "SetSortSpec" on page 255
- "SetViewMode" on page 258
- "UndoRecord" on page 262
- "WriteRecord" on page 263

Navigation

The navigation methods set the focus for user interaction to the named view. Table 11 identifies the navigation methods.

NOTE: Properties for Siebel objects such as business component applets and business components are stored in the repository and cannot be changed at run time using Siebel VB methods.

Table 11. Navigation Methods

Method
"InvokeMethod" on page 136
"GotoView" on page 163

User Interaction

The following methods allow the Siebel extension routines to interact directly with the user through traditional user interface techniques. These methods are similar to the standard procedures available to Windows programs. User interaction methods are listed in Table 12.

Table 12. User Interaction Methods

Method
"RaiseError" on page 175
"RaiseErrorText" on page 176

Global State Properties and Functions

The application object provides a set of properties and functions that return information about the current state. This information is useful when you are processing rows of data or generating query criteria. Global state methods are listed in Table 13.

Table 13. Global State Methods

Method
“CurrencyCode” on page 155
“EnableExceptions” on page 156
“GetLastErrCode” on page 159
“GetLastErrText” on page 160
“LoginId” on page 169
“LoginName” on page 170
“Logoff” on page 170
“LookupMessage” on page 171
“NewPropertySet” on page 173
“PositionName” on page 174
“SetPositionId” on page 178
“SetPositionName” on page 179

Creating a Siebel Application Object in CORBA

The method listed in Table 14 provides a way to create a Siebel application object in CORBA.

Table 14. Creating a Siebel Application Object in CORBA

Method
“CreateSiebelAppObject” on page 325

Variable Scoping for Siebel Script Variables

Three levels of scope exist for Siebel variables:

- "Local Variables"
- "Module Variables"
- "Global Variables" on page 103

See Also "Inter-Application Variable Methods" on page 103
"Tracing" on page 104

Local Variables

Local variables defined within a Siebel script are the lowest level of variable scoping. These variables are declared using the Dim statement in Siebel VB or the var statement in Siebel eScript, and their values are accessible only within the script in which they were defined.

VB example:

```
Sub Applet_Load
   Dim localStr as String
End Sub
```

eScript example:

```
function Applet_GotFocus ()
{
   var localStr;
{
```

Module Variables

Module variables defined in the (general) (declarations) section of a Siebel object (such as an applet or business component) are the next level of variable scoping. These variables are available as long as the object is instantiated and the values are accessible to scripts in the same object or module. Use Dim statements (for VB) or var statements (for eScript) in the (general) (declarations) section to declare module variables.

VB example:

```
(general) (declarations)
Dim ContactId as String
```

Code in the VB Editor in the (general) (declarations) section is illustrated in Figure 12.

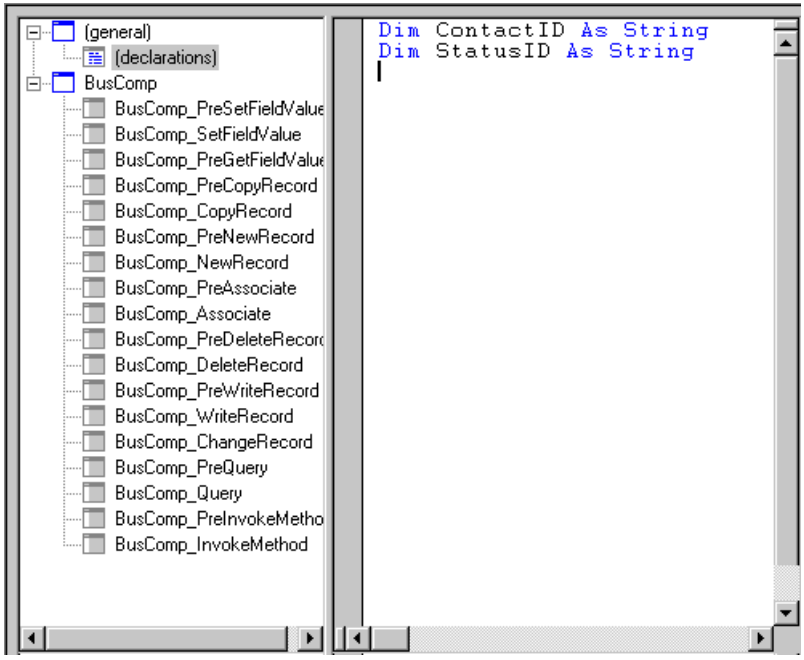


Figure 12. Declarations in the (general) (declarations) Section

Global Variables

The global variables exist at the highest level. You must declare these variables in every module that needs to access their values. Use the Global statement to declare these variables. Avoid using global variables to store Siebel objects such as BusComp and BusObject. If you need to store Siebel objects such as BusComp and BusObject, always set these variables to Nothing whenever the objects are no longer required, or at least in the Application_Close event. Failure to do so may cause memory problems because the objects being referenced cannot be freed from memory while they are still being referenced. If you must create a global variable for a business component, make sure there is a global variable for the business object. Otherwise, the business component is out of scope.

Here is an eScript example:

```
TheApplication().gVar = "some value";
```

Inter-Application Variable Methods

Siebel provides two sets of methods to send values for variables back and forth between the Siebel application and external applications. Table 15 lists inter-application communication methods.

Table 15. Inter-Application Communication Methods

Method
"GetUserProperty" on page 225
"SetUserProperty" on page 257
"GetLastErrCode" on page 159
"SetSharedGlobal" on page 180
"GetProfileAttr" on page 160
"SetProfileAttr" on page 179

Tracing

Table 16 lists Application event methods for controlling debug tracing.

Table 16. Debug Tracing Methods

Method
"Trace" on page 182
"TraceOff" on page 184
"TraceOn" on page 185

Siebel Object Interface Events and Siebel Extension Events

Selected events within the Siebel applications allow the association of extension routines that extend the base behavior of the application. These routines are available in Browser and Server Script. When the Siebel application fires or activates the event, the user-specified procedures are invoked along with the standard Siebel procedures. The event names listed under "Siebel Business Component Events" on page 109 refer to the tag or entry point used to tie the extension routine to a particular action or event.

The following topics cover the object interface events and extension events.

- "Event Method Syntax" on page 105
- "How Your Script Affects Program Flow" on page 106
- "When Events Occur" on page 109
- "Siebel Business Component Events" on page 109
- "Applet Events" on page 111
- "Application Events" on page 112
- "Connect String" on page 112

- "Error Handling" on page 115

Each topic provides the following information.

- The syntax for using the event.
- A brief description of the event.
- A checklist that indicates which interface environments support the event.

Event Method Syntax

The method's syntax uses the following form:

- *ObjectReference_EventName* (*parameters*) As *RetValu*e.
- *ObjectReference* is the variable name of the object on which the event is invoked.
- *EventName* is the event that is being invoked.

The events exposed can be classified into preoperation events or postoperation events. The preoperation events occur before the standard Siebel operation. An example of a preoperation event is PreDeleteRecord. This event occurs before a DeleteRecord event occurs.

The corresponding postoperation event is DeleteRecord. This event is fired *after* the PreDeleteRecord operation has been executed.

You can use preoperation events to alter standard Siebel behavior. For example, the PreDeleteRecord event can be used to perform additional, customer-specific validation on the record about to be deleted, and if the validations fail, the DeleteRecord operation can be canceled.

Postoperation events are useful when the event relies on data that may have been updated in the course of executing the standard Siebel event.

How Your Script Affects Program Flow

For every Siebel operation event handler, there is also a preoperation event handler. Generally, scripts are placed in the preoperation event. You can alter the effect of an event by attaching a script to the preoperation event handler. The events with the most important effects are the PreInvokeMethod events. In a PreInvokeMethod event, you can call a method that substitutes for the internal Siebel code.

As Figure 13 illustrates, you can change the outcome of an event by specifying the return value on the preoperation events. The standard return value for preoperation events is ContinueOperation, which tells the calling Siebel object to continue processing the remaining operations associated with the event, as shown in Step 2 in Figure 13.

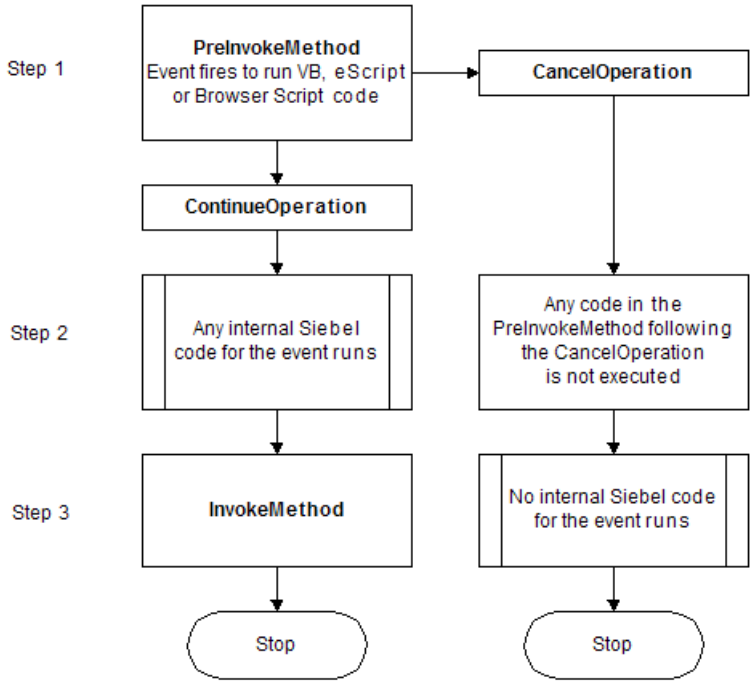


Figure 13. The Effects of CancelOperation and ContinueOperation

If you wish to create an alternative to an existing routine, change the return value in your custom event handler to CancelOperation. This tells the Siebel application to cancel the remaining operations associated with the event. If, for example, the validation in the PreDeleteRecord event fails, set the return value for the event to CancelOperation. If you want to preprocess before the default event method executes, use the return value ContinueOperation.

The post-event handler is rarely scripted, but you may use it for such post-operation events as posting a notice to a log when the event completes successfully.

The following Siebel VB example sets up a validation routine in which a specific field is queried to determine whether the event should fire.

```
Function BusComp_PreSetFieldValue (FieldName As String,
               FieldValue As String) As Integer
' Routine to check if a quote discount > 20%
'       if it is, notify user and cancel the operation
Dim value as Integer
Dim msgtext as String
   If FieldName = "Discount" then
      value = Val(FieldValue)
      If value > 20 then
          msgtext = "Discounts greater than 20% must be approved"
         TheApplication.RaiseErrorText msgtext
         BusComp_PreSetFieldValue = CancelOperation
      Else
         BusComp_PreSetFieldValue = ContinueOperation
       End if
End If
End Function
```

Notice the logical structure of this routine:

```
If (condition is true)
   [perform custom routine]
   returnValue = CancelOperation
Else
   returnValue = ContinueOperation
End If
```

Within this structure, the custom routine is executed only if the condition is true. If the condition is true, the custom routine substitutes for the built-in routine. If it is not true, the built-in routine is executed because the event handler returns ContinueOperation.

The following alternative structure is also acceptable.

```
returnValue = ContinueOperation
If (condition is true)
   [perform custom routine]
   returnValue = CancelOperation
End If
```

Note that in PreInvokeMethod events, the condition should always be a test for the method name; for example,

```
if (methodName = "PushOpportunity")
```

If more than one method may be invoked, you may find it more efficient to use a Select structure (in VB) or a switch structure (in eScript). In Siebel VB:

```
Dim iReturn As Integer
iReturn = ContinueOperation
Select Case methodName
   Case "PushOpportunity"
      [custom routine]
      iReturn = CancelOperation
   Case "Stage3"
      [custom routine]
      iReturn = CancelOperation
End Select
object_PreInvokeMethod = iReturn
```

In Siebel eScript:

```
var iReturn;
switch methodName
{
   case "PushOpportunity":
      [custom routine]
      iReturn = CancelOperation;
      break;
   case "Stage3":
      [custom routine]
      iReturn = CancelOperation;
      break;

   default:
      iReturn = ContinueOperation;
}
return (iReturn);
```

To make your code easier to read and maintain, you can create the custom routines as subprograms or functions in the (general) (declarations) section.

When Events Occur

There is no simple way to determine when various events occur, as many different events can occur when a view becomes current or when an object is invoked. To find out the exact order of events, enable tracing when the application starts (Application_Start event). The syntax resembles the following:

```
theApplication.TraceOn(filename, type, selection)
theApplication.Trace "Event_Name has fired."
```

When the preceding code has been placed on the Application_Start event, place a line of code of the following form in each event handler (including the Pre-event handlers) for the object, including insert, delete, write, business component, and any others that may apply:

```
theApplication.Trace "Event_Name fired."
```

Then perform some simple inserts, updates, and deletes, and make a note of each message as it appears. You then have a list of the order in which events fire on that view or for that object.

Siebel Business Component Events

Events can be invoked from data operations on business components. These are defined on a per-business component basis. Events can be invoked before or after the specified standard behavior.

The only means of trapping modifications to a multivalue field is through the underlying MVG business component. If the multivalue field is modified without popping up the MVG applet, then the PreSetFieldValue and SetFieldValue events for those fields are not triggered. The only way in which the PreSetFieldValue and SetFieldValue events are fired for a multivalue field is if the field is updated within the MVG applet. If the user makes a change to the multivalue field through the MVG applet, then only the events on the MVG business component are called. No events on the parent business component are called.

Table 17 and Table 18 list BusComp events.

Table 17. Server Side BusComp Events

Method
“BusComp_Associate” on page 267
“BusComp_ChangeRecord” on page 267
“BusComp_PreCopyRecord” on page 272
“BusComp_CopyRecord” on page 269
“BusComp_InvokeMethod” on page 270
“BusComp_NewRecord” on page 271
“BusComp_PreAssociate” on page 271
“BusComp_PreDeleteRecord” on page 272
“BusComp_PreGetFieldValue” on page 274
“BusComp_PreInvokeMethod” on page 275
“BusComp_PreNewRecord” on page 275
“BusComp_PreQuery” on page 276
“BusComp_PreSetFieldValue” on page 277
“BusComp_PreWriteRecord” on page 279
“BusComp_Query” on page 280
“BusComp_SetFieldValue” on page 281
“BusComp_WriteRecord” on page 282

Table 18. Browser Side BusComp Events

Method
“BusComp_PreSetFieldValue” on page 277

Applet Events

Events are invoked in response to user interactions. These can be managed on a per-applet basis. Applet events are only supported in High Interactivity mode. Table 19 and Table 20 list the User interface events.

Table 19. Server Side Applet Events

Method
"WebApplet_InvokeMethod" on page 142
"WebApplet_Load" on page 143
"WebApplet_PreCanInvokeMethod" on page 144
"WebApplet_PreInvokeMethod" on page 145

Table 20. Browser Side Applet Events

Method
"Applet_ChangeFieldValue" on page 139
"Applet_ChangeRecord" on page 140
"Applet_InvokeMethod" on page 140
"Applet_PreInvokeMethod" on page 142

Application Events

Application events are listed in Table 21 and Table 22.

Table 21. Server Side Application Events

Method
"Application_InvokeMethod" on page 190
"Application_Navigate" on page 191
"Application_PreInvokeMethod" on page 191
"Application_PreNavigate" on page 193
"Application_Start" on page 194

Table 22. Browser Side Application Events

Method
"Application_InvokeMethod" on page 190
"Application_PreInvokeMethod" on page 191

Connect String

The connect string is a URL containing the information needed to connect to any Siebel Server component. It specifies both the protocol and the details of the Client Application Manager service in the Siebel Servers to which the client connects. The generic form of the syntax for the connect string is:

```
siebel[[.transport][.[encryption][.[compression]]]]://
host[:port]/EnterpriseServer/AppObjMgr[/SiebelServer]
```

Here is an example of a connect string:

```
SiebelApplication.Login "host=""siebel://host/EnterpriseServer/
AppObjMgr/SiebelServer""", "CCONWAY", "CCONWAY"
```

Note that the entire protocol string is optional. You may specify the transport protocol alone and separate it from `siebel` with a single period. However, if you specify any of the other protocols, you must use a period as a placeholder for each protocol not specified, for example:

```
siebel...zlib://host/siebel/AppObjMgr
```

Protocols that are not specified receive their default values, as shown in Table 23.

Make the following substitutions for the placeholders in the example:

Table 23. Substitutions When Logging into a Siebel Server

In Place Of	Insert
transport	One of the following values: ■ `tcpip` (the default) ■ `http`
encryption	One of the following values: ■ `none` (default) ■ `mscrypto` (not supported by Java Data Bean) ■ `rsa` (supported by Java Data Bean)
compression	One of the following values: ■ `none` ■ `zlib` (the default)
host	The name of the computer on which the gateway server is installed
port	The gateway port; by default 2320. This changes only if the Siebel administrator has changed the default during installation

Table 23. Substitutions When Logging into a Siebel Server

In Place Of	Insert
AppObjMgr	The name of the defined Application Object Manager that you want the thin client to access; this can be a user-defined component or one of these predefined components: ■ `ISSObjMgr_<lang>` ■ `SCCObjMgr_<lang>` ■ `SSEObjMgr_<lang>` ■ `SSVObjMgr_<lang>` (For more information, read *Siebel Server Administration Guide*.)
SiebelServer	The Siebel Server to which the client should connect. Note that this value should not be specified if Resonate is being used. This is because the Resonate Server dynamically chooses the Siebel Server with the lowest utilization.

For more information about this method, read "Login" on page 167.

Here is a sample connect string for the COM Data Control operating in Server Mode:

```
   '''COM Data Control : SERVER Mode
   lstr = "host=" + """siebel://frashid/Siebel/SSEObjMgr/frashid"""
   Format of the connect string is
   '"host=" + """siebel://<Gateway>/<Enterprise>/<App. Object Mgr>/<Siebel
Server>"""

   retval = siebDataCtl.Login(lng + lstr, "username", "password")
```

Here is a sample connect string for the COM Data Control operating in Local Mode. When running in Local Mode, the COM Data Control must reside on the same machine as the Mobile Web Client.

```
'   ''COM Data Control : LOCAL Mode
   lstr = "cfg=" + """D:\Client\mwebc\BIN\ENU\siebel.cfg,ServerDataSrc"""

   'Format of the connect string is
   '"cfg=" + """Absolute path of the CFG file, DataSource"""
   'Datasource = ServerDataSrc or Local or Sample

   retval = siebDataCtl.Login(lng + lstr, "username", "password")
```

Here is a sample connect string for the COM Data Control operating in a load-balanced environment:

```
'''COM Data Control : Resonate Load Balancing
lstr = "host=" + """siebel://12.22.20.99/Siebel/SSEObjMgr"""
'Format of the connect string is
'"host=" + """siebel://<VIP>/<Enterprise>/<App. Object Mgr>"""
'Or, the connect string is
'lstr = "host=" + """siebel.tcpip.none.none://12.22.20.99/Siebel/SSEObjMgr"""

retval = siebDataCtl.Login(lng + lstr, "username", "password")
```

Here is a sample connect string for the COM Data Control for PowerBuilder (Char(34) denotes a double quote):

```
ConnStr = "host =" + char(34) + "siebel://HOST/ENTERPRISE_SERVER/SCCObjMgr/
SIEBEL_SERVER" + char(34) + " Lang = " + char(34) + "LANG" + char(34)
```

Error Handling

This section explains the Siebel COM Interfaces error handling differences.

COM Error Handling

The errCode parameter is the standard last parameter for every COM Data Server interface method. It is not available in the COM Data Control, Mobile/Dedicated Web Client Automation Server, Web Client Automation Server, CORBA Object Manager, or Java Data Bean. An example of this is the GetBusObject method. The following section shows the difference between the two methods:

Error Handling Example—COMData Server only

```
GetBusObject (BusObjectName as string, errcode as integer) ->
businessObject
```

Error Handling Example—COM Data Control and Mobile Web Client Automation

```
GetBusObject (BusObjectName as string) -> businessObject
```

Java Error Handling

The Siebel Java interfaces error-handling differences are explained in this section.

Errors in the Siebel Java Data Bean are handled by the SiebelException object. It supports the getErrorCode() and getErrorMessage() methods. The SiebelException object is defined in com.siebel.data.SiebelException.

It is used as follows:

```
...

import com.siebel.data.SiebelException;
import com.siebel.data.SiebelDataBean;
...
SiebelDataBean mySiebelBean=null;
try

{
   mySiebelBean = new SiebelDataBean();
   mySiebelBean.login("Siebel://SOMSERVER/somsiebel/AppObjMgr/
SomServer,"CCONWAY", "CCONWAY","enu");
}
catch (SiebelException e){
   // Exception handling code
   System.out.println (e.getErrorMessage ());
mySiebelBean = null; //avoid using mySiebelBean if login is
unsuccessful
}

...
```

For additional methods on the SiebelException object, refer to the Siebel Java Data Bean JavaDoc installed with Siebel Tools. Note that the JavaDoc is installed only if the "Siebel Java Integration" option is installed. If so, then a zipped file containing the JavaDoc is in the `<tools install>\CLASSES` folder.

Error Message Tracking

For error message tracking in ActiveX and CORBA, you can use either exceptions or methods. The following methods are available:

- EnableExceptions (COM only)
- GetLastErrCode (CORBA and COM only)
- GetLastErrText (CORBA and COM only)

EnableExceptions Method

```
EnableExceptions(enable as integer)
```

The EnableExceptions method allows applications to use the native COM error-handling technique. If the method is about to fail due to error, then a COM exception is generated and the method does not return. The COM host receives the control instead. However, it *may* display the error message (this is default for Microsoft Internet Explorer or VB), but it can be changed by scripting.

GetLastErrCode, GetLastErrText Method

After execution of a method, the GetLastErrCode can be invoked to check if any error was returned from the previous operation. The GetLastErrText method can be invoked to retrieve the text of the error message.

```
GetLastErrCode() ' retrieves errCode As Integer

GetLastErrText() ' retrieves text As String
```

CORBA Exception Handling

Every CORBA Object method raises a SiebelException, except for those noted in Chapter 3, "Interfaces Reference" as "implemented as an attribute." The following code example illustrates how to handle CORBA exceptions where excep.id is the exception code (or id) and excep.desc is the exception description.

```
SiebelAppFactory_var appFact;
SiebelApplication_var appObj;

try {
   appObj = appFact->CreateSiebelAppObject();
} catch (const SiebelException& seblExcep) {
   cerr << "Error id : " << seblExcep.id << endl;
   cerr << "Error text : " << endl;
   cerr << seblExcep.desc << endl;
   return;
}
```

Interfaces Reference 3

This chapter lists the methods and events available to Siebel Object Interfaces.

- "Object Interface Methods Tables" on page 120
- "Object Interface Events" on page 129
- "Siebel Constants" on page 132
- "Applet Methods" on page 133
- "Applet Events" on page 138
- "Application Methods" on page 151
- "Application Events" on page 189
- "Business Component Methods" on page 196
- "Business Component Events" on page 266
- "Business Object Methods" on page 283
- "Business Service Methods" on page 287
- "Business Service Events" on page 295
- "Control Methods" on page 301
- "Property Set Methods" on page 309
- "SiebelAppFactory Methods" on page 325
- "Miscellaneous Methods" on page 327
- "Miscellaneous Methods" on page 327

Object Interface Methods Tables

This section lists the Siebel interface methods, grouped by object interface type.

- "Applet"
- "Application" on page 121
- "Business Component" on page 123
- "Business Object" on page 125
- "Business Service" on page 125
- "Control" on page 126
- "Property Set" on page 127
- "SiebelApp Factory" on page 128
- "Siebel Exceptions" on page 128
- "Miscellaneous Methods and Events" on page 128

Applet

Method	Server Script	Browser Script	Web Client Automation on Server	Mobile/ Dedicated Web Client Automation Server	Siebel COM Data Control	COM Data Server	Java Data Bean	Siebel CORBA Object Manager
ActiveMode		X						
BusComp	X	X						
BusObject	X	X						
FindActiveXControl		X						
FindControl		X						
InvokeMethod	X	X						
Name	X	X						

Application

Method	Server Script	Browser Script	Web Client Automation on Server	Mobile/ Dedicated Web Client Automation Server	Siebel COM Data Control	COM Data Server	Java Data Bean	Siebel CORBA Object Manager
ActiveApplet		X						
ActiveBusComp		X						
ActiveBusObject	X	X		X				
ActiveViewName	X	X		X				
Attach					X		X	
CurrencyCode	X	X		X	X	X	X	X
Detach					X		X	
EnableExceptions				X	X			
FindApplet		X						
GetBusObject	X			X	X	X	X	X
GetDataSource	X			X	X		X	
GetLastErrCode			X	X	X			
GetLastErrText			X	X	X	X		X
GetProfileAttr	X	X		X	X	X	X	X
GetService	X	X	X	X	X	X	X	X
GetSharedGlobal	X			X	X	X	X	X
GotoView	X							
InvokeMethod	X	X		X	X	X	X	X
LoadObjects						X		
LoadUserAttributes	X							
Login					X	X	X	X
LoginId	X			X	X	X	X	X
LoginName	X			X	X	X	X	X

Method	Server Script	Browser Script	Web Client Automation on Server	Mobile/ Dedicated Web Client Automation Server	Siebel COM Data Control	COM Data Server	Java Data Bean	Siebel CORBA Object Manager
Logoff				X	X		X	X
LookupMessage	X							
LookupValue	X			X	X		X	
Name		X	X					
NewPropertySet	X	X	X	X	X	X	X	
PositionId	X			X	X	X	X	X
PositionName	X			X	X	X	X	X
RaiseError	X							
RaiseErrorText	X							
Release								X
SetPositionId	X			X	X	X	X	X
SetPositionName	X			X	X	X	X	X
SetProfileAttr	X	X		X	X	X	X	X
SetSharedGlobal	X			X	X	X	X	X
SWEAlert		X						
Trace	X			X	X	X	X	X
TraceOff	X			X	X	X	X	X
TraceOn	X			X	X	X	X	X

Business Component

Method	Server Script	Browser Script	Web Client Automation on Server	Mobile/ Dedicated Web Client Automation Server	Siebel COM Data Control	COM Data Server	Java Data Bean	Siebel CORBA Object Manager
ActivateField	X			X	X	X	X	X
ActivateMultipleFields	X			X	X	X	X	X
Associate	X			X	X	X	X	X
BusObject	X	X		X	X	X	X	X
ClearToQuery	X			X	X	X	X	X
DeactivateFields	X			X	X	X	X	X
DeleteRecord	X			X	X	X	X	X
ExecuteQuery	X			X	X	X	X	X
ExecuteQuery2	X			X	X	X	X	X
FirstRecord	X			X	X	X	X	X
FirstSelected	X							
GetAssocBusComp	X			X	X	X	X	X
GetFieldValue	X	X		X	X	X	X	X
GetFormattedFieldValue	X	X		X	X	X	X	X
GetLastErrCode				X	X			
GetLastErrText				X	X			
GetMultipleFieldValues	X			X	X	X	X	X
GetMVGBusComp	X			X	X	X	X	X
GetNamedSearch	X			X	X	X	X	X
GetPicklistBusComp	X			X	X	X	X	X
GetSearchExpr	X	X		X	X	X	X	
GetSearchSpec	X	X		X	X	X	X	X
GetUserProperty	X			X	X	X	X	X

Method	Server Script	Browser Script	Web Client Automation on Server	Mobile/ Dedicated Web Client Automation Server	Siebel COM Data Control	COM Data Server	Java Data Bean	Siebel CORBA Object Manager
GetViewMode	X			X	X	X	X	
InvokeMethod	X			X	X	X	X	X
LastRecord	X			X	X	X	X	X
Name	X	X		X	X	X	X	X
NewRecord	X			X	X	X	X	X
NextRecord	X			X	X	X	X	X
NextSelected	X							
ParentBusComp	X			X	X	X	X	X
Pick	X			X	X	X	X	X
PreviousRecord	X			X	X	X	X	X
RefineQuery	X			X	X	X	X	X
Release							X	
SearchExpr								X
SetFieldValue	X	X		X	X	X	X	X
SetFormattedFieldValue	X	X		X	X	X	X	X
SetMultipleFieldValues	X			X	X	X	X	X
SetNamedSearch	X			X	X	X	X	X
SetSearchExpr	X			X	X	X	X	
SetSearchSpec	X			X	X	X	X	X
SetSortSpec	X			X	X	X	X	X
SetUserProperty	X			X	X	X	X	X
SetViewMode	X			X	X	X	X	
UndoRecord	X			X	X	X	X	X

Method	Server Script	Browser Script	Web Client Automation on Server	Mobile/ Dedicated Web Client Automation Server	Siebel COM Data Control	COM Data Server	Java Data Bean	Siebel CORBA Object Manager
ViewMode								X
WriteRecord	X	X		X	X	X	X	X

Business Object

Method	Server Script	Browser Script	Web Client Automation on Server	Mobile/ Dedicated Web Client Automation Server	Siebel COM Data Control	COM Data Server	Java Data Bean	Siebel CORBA Object Manager
GetBusComp	X	X		X	X	X	X	X
GetLastErrCode				X	X			
GetLastErrText				X	X			
Name	X	X		X	X	X	X	X
Release							X	

Business Service

Method	Server Script	Browser Script	Web Client Automation on Server	Mobile/ Dedicated Web Client Automation Server	Siebel COM Data Control	COM Data Server	Java Data Bean	Siebel CORBA Object Manager
GetFirstProperty	X	X		X	X	X	X	X
GetLastErrCode				X				
GetLastErrText				X				
GetNextProperty	X	X		X	X	X	X	X
GetProperty	X	X		X	X	X	X	X

Method	Server Script	Browser Script	Web Client Automation on Server	Mobile/ Dedicated Web Client Automation Server	Siebel COM Data Control	COM Data Server	Java Data Bean	Siebel CORBA Object Manager
InvokeMethod	X	X	X	X	X	X	X	X
Name	X	X	X	X	X	X	X	X
PropertyExists	X	X		X	X	X	X	X
Release							X	
RemoveProperty	X	X		X	X	X	X	X
SetProperty	X	X		X	X	X	X	X

Control

Method	Server Script	Browser Script	Web Client Automation on Server	Mobile/ Dedicated Web Client Automation Server	Siebel COM Data Control	COM Data Server	Java Data Bean	Siebel CORBA Object Manager
Applet		X						
BusComp		X						
GetProperty		X						
GetValue		X						
Name		X						
SetLabelProperty		X						
SetProperty		X						
SetValue		X						

Property Set

Method	Server Script	Browser Script	Web Client Automation on Server	Mobile/ Dedicated Web Client Automation Server	Siebel COM Data Control	COM Data Server	Java Data Bean	Siebel CORBA Object Manager
AddChild	X	X	X	X	X	X	X	
Copy	X	X	X	X	X	X	X	
GetChild	X	X	X	X	X	X	X	
GetChildCount	X	X	X	X	X	X	X	
GetFirstProperty	X	X	X	X	X	X	X	
GetNextProperty	X	X	X	X	X	X	X	
GetProperty	X	X	X	X	X	X	X	
GetPropertyCount	X	X	X	X	X	X	X	
GetType	X	X	X	X	X	X	X	
GetValue	X	X	X	X	X	X	X	
InsertChildAt	X	X	X	X	X	X	X	
PropertyExists	X	X	X	X	X	X	X	
RemoveChild	X	X	X	X	X	X	X	
RemoveProperty	X	X	X	X	X	X	X	
Reset	X	X	X	X	X	X	X	
SetProperty	X	X	X	X	X	X	X	
SetType	X	X	X	X	X	X	X	
SetValue	X	X	X	X	X	X	X	

SiebelApp Factory

Method	Server Script	Browser Script	Web Client Automation on Server	Mobile/ Dedicated Web Client Automation Server	Siebel COM Data Control	COM Data Server	Java Data Bean	Siebel CORBA Object Manager
CreateSiebelAppObject								X
Release								X

Siebel Exceptions

Method	Server Script	Browser Script	Web Client Automation on Server	Mobile/ Dedicated Web Client Automation Server	Siebel COM Data Control	COM Data Server	Java Data Bean	Siebel CORBA Object Manager
desc								X
id								X

Miscellaneous Methods and Events

Method	Server Script	Browser Script	Web Client Automation on Server	Mobile/ Dedicated Web Client Automation Server	Siebel COM Data Control	COM Data Server	Java Data Bean	Siebel CORBA Object Manager
GetErrorCode							X	
GetErrorMessage();							X	
TheApplication	X	X						

Object Interface Events

The object interface events are available in Server Script or Browser Script within Siebel Tools. This section lists the Siebel interface events, grouped by object interface type.

- "Applet"
- "Application" on page 130
- "Business Component Events" on page 130
- "Business Service Events" on page 131

Applet

Event	Server Script	Browser Script	Comments
Applet_ChangeFieldValue		X	
Applet_ChangeRecord		X	
Applet_InvokeMethod		X	
Applet_Load		X	
Applet_PreInvokeMethod		X	
WebApplet_InvokeMethod	X		
WebApplet_Load	X		
WebApplet_PreCanInvokeMethod	X		
WebApplet_PreInvokeMethod	X		
WebApplet_ShowControl	X		Not available in High Interactivity mode
WebApplet_ShowListColumn	X		Not available in High Interactivity mode

Application

Event	Server Script	Browser Script	Comments
Application_Close	X		
Application_InvokeMethod	X	X	
Application_Navigate	X		
Application_PreInvokeMethod	X	X	
Application_PreNavigate	X		
Application_Start	X		

Business Component Events

Event	Server Script	Browser Script	Comments
BusComp_Associate	X		
BusComp_ChangeRecord	X		
BusComp_CopyRecord	X		
BusComp_DeleteRecord	X		
BusComp_InvokeMethod	X		
BusComp_NewRecord	X		
BusComp_PreAssociate	X		
BusComp_PreCopyRecord	X		
BusComp_PreDeleteRecord	X		
BusComp_PreGetFieldValue	X		
BusComp_PreInvokeMethod	X		
BusComp_PreNewRecord	X		
BusComp_PreQuery	X		

Event	Server Script	Browser Script	Comments
BusComp_PreSetFieldValue	X	X	Available only in High Interactivity mode. Requires a field property to be set for the event to be immediately executed on the server.
BusComp_PreWriteRecord	X		
BusComp_Query	X		
BusComp_SetFieldValue	X		
BusComp_WriteRecord	X		

Business Service Events

Event	Server Script	Browser Script	Comments
Service_InvokeMethod	X	X	
Service_PreCanInvokeMethod	X	X	
Service_PreInvokeMethod	X	X	

Siebel Constants

The Siebel programming languages provide constants for the convenience of programmers. These constants appear in the table that follows. Use the constant names, rather than their integer values in your code. Use of these constant names makes your code more readable by others, because it clarifies your intentions. However, the integer values are included to aid in debugging, as the integer values are what appear in the Debugger.

Used With	Constant Name	Integer Value
Pre Event Handler Methods	ContinueOperation	1
	CancelOperation	2
Search Methods	ForwardBackward	0
	ForwardOnly	1
NewRecord Method	NewBefore	0
	NewAfter	1
	NewBeforeCopy (Not available with Java Data Bean)	2
	NewAfterCopy (Not available with Java Data Bean)	3
SiebelViewMode Methods	SalesRepView	0
	ManagerView	1
	PersonalView	2
	AllView	3
	OrganizationView	5
	GroupView	7
	CatalogView	8
	SubOrganizationView	9

Applet Methods

In the following methods, the placeholder *oApplet* in the syntax refers to a variable representing a specific applet.

- "ActiveMode"
- "BusComp" on page 134
- "BusObject" on page 134
- "FindActiveXControl" on page 135
- "FindControl" on page 136
- "InvokeMethod" on page 136
- "Name" on page 138

ActiveMode

ActiveMode returns a string containing the name of the current Web Template mode.

Syntax *oApplet*.ActiveMode

Argument	Description
Not applicable	

Returns A string containing the name of the current Web Template mode.

Used With Browser Script

Example Here is an example for Browser Script:

```
function Applet_PreInvokeMethod (name, inputPropSet)

{
   if (name == "GetActiveMode") {
      var currMode = this.ActiveMode();
```

```
      TheApplication().SWEAlert ("The active mode for the
selected applet is: " + currMode);
      return ("CancelOperation");
   }
   else
      return ("ContinueOperation")
}
```

BusComp

BusComp returns the business component that is associated with the applet.

Syntax *oApplet*.BusComp();

Argument	Description
Not applicable	

Returns The business component associated with the applet.

Used With Browser Script, Server Script

BusObject

BusObject returns the business object for the business component of the applet.

Syntax *oApplet*.BusObject()

Argument	Description
Not applicable	

Returns The business object for the applet's business component.

Used With Browser Script, Server Script

Example Here is a Siebel VB example:

```
Sub WebApplet_Load
      Dim oBusObject As BusObject
      Dim oBusComp As BusComp
      Set oBusObject = Me.BusObject

End Sub
```

Here is an eScript example:

```
function WebApplet_Load ()
{
   var busObj = this.BusObject();
}
```

FindActiveXControl

FindActiveXControl returns a reference to a DOM element based upon the name specified in the name argument.

Syntax *oApplet*.FindActiveXControl(*controlName*)

Argument	Description
controlName	Literal string or string variable containing the name of the desired control

Returns The control object identified in *controlName*.

Used With Browser Script

Example Here is a Browser Script example, which interacts with the Microsoft slide control:

```
var elem = FindActiveXControl("SliderControl");
   TheApplication().SWEAlert ("element id = " + elem.id);   // id
of the element
   TheApplication().SWEAlert ("Max ticks = " + elem.Max);   //
max number of ticks in the slider control
   elem.SelStart = 2;   // setting SelStart property
   elem.Refresh();   // invoking the Refresh method on the
control
```

FindControl

FindControl returns the control whose name is specified in the argument. This applet must be part of the displayed view.

Syntax *oApplet*.FindControl(*controlName*)

Argument	Description
controlName	Literal string or string variable containing the name of the desired control

Returns The control object identified in *controlName*.

Usage FindControl does not find controls for MVG applets, Pick applets, Associate applets, or detail applets that are not on the view's applet list.

Used With Browser Script

InvokeMethod

The InvokeMethod method invokes the specialized or custom method specified by its argument.

Browser Script Syntax *oApplet*.InvokeMethod(*methodName, methodArgs_PropSet);*

Argument	Description
methodName	The name of the method
methodArgs	Property set containing the method arguments

Server Script Syntax

Applet.InvokeMethod(*methodName, methodArgs*);

Argument	Description
methodName	The name of the method
*methArg1, methArg2, …, methArg*N	One or more strings containing arguments to *methodName*

Returns

In Server Script, returns a string containing the result of the method.

In Browser Script, returns a property set.

Usage

Available to Browser and Server scripting. If the method to be invoked exists in the Browser, it executes in the browser. Otherwise, the request is sent to the server for execution.

NOTE: The InvokeMethod method should be used only with documented specialized methods. Siebel Systems does not support calling specialized methods with InvokeMethod, unless they are listed in this book.

Used With

Browser Script, Server Script

Example

Here is a Siebel VB example:

```
Function WebApplet_PreInvokeMethod (MethodName As String) As
Integer
' Invoke a Siebel SmartScript from a custom button
' using the applet.InvokeMethod method
   If (MethodName = "InvokeSScriptFromButton") Then
      Dim sArgs(3) As String
      sArgs(0) = "Demo Opportunity Profile"
      sArgs(1) = ""
      sArgs(2) = ""
      Me.InvokeMethod "RunCallScript", args
      WebApplet_PreInvokeMethod = CancelOperation
   Else
      WebApplet_PreInvokeMethod = ContinueOperation
   End If
End Function
```

Name

The Name method returns the name of the applet.

Syntax *oApplet*.Name()

Argument	Description
Not applicable	

Returns A string containing the applet object name.

Used With Browser Script, Server Script

Example Here is a Siebel VB example:

```
Function WebApplet_PreInvokeMethod (MethodName As String) As
Integer

' Display the name of the applet from a custom button using the
' applet.Name() method to obtain the name of the applet
   If (MethodName = "displayAppletName") Then
      Dim appletName As String
      appletName = Me.Name()
      TheApplication.RaiseErrorText("The name of the applet is: "
& appletName)
      WebApplet_PreInvokeMethod = CancelOperation
   Else
      WebApplet_PreInvokeMethod = ContinueOperation
   End If
End Function
```

Applet Events

The following topics describe applet events.

- "Applet_ChangeFieldValue" on page 139
- "Applet_ChangeRecord" on page 140
- "Applet_InvokeMethod" on page 140

- "Applet_Load" on page 141
- "Applet_PreInvokeMethod" on page 142
- "WebApplet_InvokeMethod" on page 142
- "WebApplet_Load" on page 143
- "WebApplet_PreCanInvokeMethod" on page 144
- "WebApplet_PreInvokeMethod" on page 145
- "WebApplet_ShowControl" on page 145
- "WebApplet_ShowListColumn" on page 148

Applet_ChangeFieldValue

The ChangeFieldValue event fires after the data in a field changes.

Syntax *Applet_ChangefieldValue(fieldname, fieldValue)*

Parameter	Description
FieldName	A string representing the name of the field whose value changed
FieldValue	A string representing the new value assigned to FieldName

Returns Not applicable

Usage ChangeFieldValue fires after the data in a field changes, but not when a user moves to a different record without changing a value in the previous record. If a user changes the value in a field, and other dependent fields, such as calculated fields, change as a result, the event fires once for each field whose value changed.

Used With Browser Script

Example Here is a Browser Script example:

```
function Applet_ChangefieldValue(field, value)
{
   TheApplication().SWEAlert("Changed field: " + field + " to " +
value);
}
```

See Also "Applet_ChangeRecord"

Applet_ChangeRecord

The ChangeRecord event is called when the user moves to a different row or view.

Syntax Applet_ChangeRecord()

Parameter	Description
Not applicable	

Returns Not applicable

Used With Browser Script

Example Here is a Browser Script example:

```
function Applet_ChangeRecord ()

{
TheApplication().SWEAlert ("Moving from one record to another");
}
```

See Also "Applet_ChangeFieldValue" on page 139

Applet_InvokeMethod

The InvokeMethod event is triggered by a call to applet.InvokeMethod or a specialized method, or by a user-defined menu.

Syntax Applet_InvokeMethod

Argument	Description
Not applicable	

Returns Not applicable

Usage Typical uses include showing or hiding controls, or setting a search specification. When accessing a business component from this event handler, use this.BusComp(), rather than theApplication.ActiveBusComp.

Used With Browser Script

See Also "Applet_PreInvokeMethod" on page 142
"Application_InvokeMethod" on page 190

Applet_Load

The Load event is triggered after an applet has loaded and after data is displayed.

Syntax Applet_Load()

Argument	Description
Not applicable	

Returns Not applicable

Usage You can use this event to dynamically hide or manipulate controls or set properties on an ActiveX Control. The following controls can be dynamically modified: CheckBox, ComboBox, TextBox, TextArea, Label.

Used With Browser Script

Applet_PreInvokeMethod

The PreInvokeMethod event is called before a specialized method is invoked, by a user-defined applet menu, or by calling InvokeMethod on an applet.

Syntax Applet_PreInvokeMethod(*Name, inputPropSet*)

Argument	Description
inputPropSet	A property set containing parameters to be passed to the PreInvokeMethod event

Returns ContinueOperation or CancelOperation

Usage The PreInvokeMethod event is called just before a specialized method is invoked on the applet. If implementing a new method (not defined by the built-in functions), the Basic script should return CancelOperation to avoid invoking an "Unknown Method Name" error. Specialized methods are methods based on applet or business component classes other than CSSFrame and CSSBusComp, respectively— that is, specialized classes.

CancelOperation does not stop the execution of the code following it, but it does prevent the execution of any built-in code associated with this event. Applet_PreInvokeMethod should return CancelOperation when you are handling the event entirely through scripting and don't want the built-in code to execute. However, if there is code in the same script following CancelOperation, that code runs regardless of the CancelOperation.

Used With Browser Script

See Also "How Your Script Affects Program Flow" on page 106

WebApplet_InvokeMethod

The InvokeMethod event is called after a specialized method on the Web applet has been executed. WebApplet_InvokeMethod triggers for Siebel-defined methods only, it does not trigger for user-defined methods.

Syntax WebApplet_InvokeMethod(*methodName)*

Argument	Description
methodName	String variable or literal containing the name of the method invoked.

Returns Not applicable

Used With Server Script

See Also "Applet_InvokeMethod" on page 140
"Application_InvokeMethod" on page 190
"WebApplet_PreCanInvokeMethod" on page 144

WebApplet_Load

The Load event is triggered just after an applet is loaded.

Syntax WebApplet_Load()

Argument	Description
Not applicable	

Returns Not applicable

Usage Do not call TheApplication().ActiveBusObject from WebApplet_Load because it returns a null. Instead use this.BusObject() to obtain a reference to the current business object.

Used With Server Script

See Also "Applet_InvokeMethod" on page 140
"Application_InvokeMethod" on page 190
"WebApplet_PreCanInvokeMethod" on page 144

Example Here is an eScript example:

```
function WebApplet_Load ()

{
try
{
var currBC = this.BusComp();
currBC.SetViewMode(5);
currBC.ClearToQuery;
currBC.SetSearchSpec("Last Name", "A*");
currBC.ExecuteQuery(ForwardBackward);
return (CancelOperation);
}
catch (e)
{
TheApplication().RaiseErrorText(e.errText);
}
}
```

WebApplet_PreCanInvokeMethod

The PreCanInvokeMethod event is called before the PreInvokeMethod, allowing the developer to determine whether or not the user has the authority to invoke the Applet method.

Syntax WebApplet_PreCanInvokeMethod(*MethodName, &CanInvoke*)

Argument	Description
MethodName	A string representing the name of the method to be executed.
&CanInvoke	A string representing whether or not the Applet method can be invoked. Valid values are TRUE or FALSE.

Returns CancelOperation or ContinueOperation

Used With Server Script

WebApplet_PreInvokeMethod

The PreInvokeMethod event is called before a specialized method for the Web applet is invoked or a user-defined method is invoked through *oWebApplet*.InvokeMethod.

Syntax WebApplet_PreInvokeMethod(*methodName*)

Argument	Description
methodName	String variable or literal containing the name of the method invoked

Returns "ContinueOperation" or "CancelOperation"

Usage The PreInvokeMethod event is called just before a specialized method is invoked on the Web applet. If implementing a new method (not defined by the built-in functions), the script should return CancelOperation to avoid invoking an "Unknown Method Name" error.

CancelOperation does not stop the execution of the code following it, but it does prevent the execution of any built-in code associated with this event. WebApplet_PreInvokeMethod should return CancelOperation when you are handling the event entirely through scripting and you don't want the built-in code to execute. However, if there is code in the same script following CancelOperation, that code runs regardless of the CancelOperation.

Used With Server Script

WebApplet_ShowControl

This event allows scripts to modify the HTML generated by the Siebel Web Engine to render a control on a Web page in an application running in Standard Interactivity mode.

Syntax WebApplet_ShowControl (*controlName, property, mode, HTML*)

Parameter	Description
controlName	A string indicating the name of the control to be rendered.
property	A string indicating the value of the property attribute of the `swe:control` or `swe:this` tag that triggers this event; it can also be an empty string if this attribute is not specified for the tag.
mode	The mode of the applet that is being shown; possible modes are: ■ Base ■ Edit ■ New ■ Query ■ Sort
HTML	The HTML generated by the Siebel Web Engine for the `swe:control` or `swe:this` tag that triggers this event.

Returns Not applicable

Usage The generated HTML depends on the control, the property being shown, and the mode of the applet. The script can modify the value of the HTML parameter, and the Siebel Web Engine sends the modified value back to the Web browser.

Customer applications render the layout of applets using template files (.swt files). These are HTML files that contain special placeholder tags that indicate where a control is to be rendered. These control placeholder tags (`<swe:control>`) can be included in the following two ways:

- The `<swe:control>` tag by itself is used to show a control:

```
<swe:control id="1" property="DisplayName"/>
```

- The `<swe:control>` tag and `<swe:this>` tag are used to show a control.

```
<swe:control id="1">
.
.
.
```

```
<swe:this property="DisplayName"/>
.
.
.
</swe:control>
```

In the first instance, if the control ID is mapped to an actual control in the applet using Siebel Tools, Siebel Web Engine renders the DisplayName property of the control at the point where this tag is placed in the template file.

In the second instance, the Siebel Web Engine renders the DisplayName property of the control at the point where the `<swe:this>` tag is placed in the template file. The outer `<swe:control>` tag in this case is used only to check if the control ID is mapped to an actual control in the applet.

The Siebel Web Engine converts these tags into HTML to render the controls on the Web page. The WebApplet_ShowControl event is triggered for each of these tags after the Siebel Web Engine has generated the HTML for rendering the control, but before the generated HTML is sent back to the browser. This gives the scripts a chance to modify the generated HTML before it is shown.

In the first example, the event fires only once, after the Siebel Web Engine generates the HTML for the `<swe:control>` tag. In the second example, this event gets fired twice. The event is first fired when the Siebel Web Engine has generated the HTML for the `<swe:this>` tag. The event is fired again when the Siebel Web Engine has generated the HTML for the outer `<swe:control>` tag; that is, after everything between the `<swe:control>` and `</swe:control>` tags, including the `<swe:this>` tag, is converted into HTML. The script can distinguish between these two event calls by the value of the property attribute of the tag that is passed as a parameter to the event.

The WebApplet_ShowControl event is supported in Standard Activity applications only.

Used With

Server Script

Example

This Siebel eScript script displays negative amounts in red in a read-only form.

```
function WebApplet_ShowControl (ControlName, Property, Mode, &HTML)
```

```
{
   var BC = me.BusComp;
   if( ControlName == "Amount" && Mode == "Base" && Property ==
"FormattedHTML")
   {
      var amount = ToNumber(BC.GetFieldValue ("Transaction
Amount"));
         If amount < 0
            HTML = "<FONT Color=Red> " + HTML + " </FONT>";
   }
}
```

WebApplet_ShowListColumn

This event allows scripts to modify the HTML generated by the Siebel Web Engine to render a list column on a Web page in an application running in High Interactivity mode.

Syntax WebApplet_ShowListColumn (*columnName*, *property*, *mode*, *HTML*)

Parameter	Description
columnName	A string indicating the name of the list column to be rendered
property	A string indicating the value of the property attribute of the `swe:control` or `swe:this` tag that triggers this event; it can also be a empty string if this attribute is not specified for the tag.
mode	The mode of the applet that is being shown; possible modes are: ■ Base ■ Edit ■ New ■ Query ■ Sort
HTML	The HTML generated by the Siebel Web Engine for the `swe:control` or `swe:this` tag that triggers this event

Returns Not applicable

Usage The generated HTML depends on the list column, the property being shown, and the mode of the applet. The script can modify the value of the HTML parameter, and the Siebel Web Engine sends the modified value back to the Web browser.

Customer applications render the layout of applets using template files (.swt files). These are HTML files that contain special placeholder tags that indicate where a control is to be rendered. These control placeholder tags (`<swe:control>`) can be included in the following two ways:

- The `<swe:control>` tag by itself is used to show a list column:

```
<swe:control id="1" property="DisplayName"/>
```

- The `<swe:control>` tag and `<swe:this>` tag are used to show a list column:

```
<swe:control id="1">
.
.
.
<swe:this property="DisplayName"/>
.
.
.
</swe:control>
```

In the first instance, if the list column ID is mapped to a list column in the applet using Siebel Tools, Siebel Web Engine renders the DisplayName property of the list column at the point where this tag is placed in the template file.

In the second instance, the Siebel Web Engine renders the DisplayName property of the list column at the point where the `<swe:this>` tag is placed in the template file. The outer `<swe:control>` tag in this case is used only to check if the list column ID is mapped to an actual list column in the applet.

The Siebel Web Engine converts these tags into HTML to render the list columns on the Web page. The WebApplet_ShowListColumn event is triggered for each of these tags after the Siebel Web Engine has generated the HTML for rendering the list column, but before the generated HTML is sent back to the browser. This gives the scripts a chance to modify the generated HTML before it is shown.

In the first example, the event fires only once, after the HTML for the `<swe:control>` tag is generated by the Siebel Web Engine. In the second example, this event is triggered twice. The event is first triggered when the Siebel Web Engine has generated the HTML for the `<swe:this>` tag. The event is fired again when the Siebel Web Engine has generated the HTML for the outer `<swe:control>` tag; that is, after everything between the `<swe:control>` and `</swe:control>` tags, including the `<swe:this>` tag, is converted into HTML. The script can distinguish between these two event calls by the value of the property attribute of the tag that is passed as a parameter to the event.

The WebApplet_ShowListColumn event is supported in Standard Activity applications only.

Used With Server Script

Example This Siebel VB script displays negative amounts in a list in red.

```
   Sub WebApplet_ShowListColumn (ColumnName As String, Property As
   String, Mode As String, HTML As String)

      Dim BC as BusComp
      Dim amount as Double

      Set BC = me.BusComp

      If ColumnName = "Amount" and Mode = "Base" and Property = _
      "FormattedHTML" Then
            If HTML < 0 Then
            HTML = "<FONT Color=Red> " + HTML + " </FONT>"
         End If
   End If

End Sub
```

Here is an example in eScript:

```
   function WebApplet_ShowListColumn (ColumnName, Property, Mode,
   &HTML)
   {
      var bc = this.BusComp();
      if ((ColumnName == 'Amount') && (Mode == "Base") && (Property
   == "FormattedHTML")) {
            var val - HTML.valueOf();
            if (val < 0)
```

```
                HTML = "<FONT Color=Red> " + HTML + " </FONT>";
            }
}
```

Application Methods

In the method and property descriptions that follow, *Application* represents a constant that stands for the Siebel application. When called from an external interface, this constant is `SiebelApplication`. From within Siebel Tools, the constant is `theApplication` or `TheApplication` in Siebel VB, `TheApplication()` in Siebel eScript, and `theApplication` in Browser Script.

- "ActiveApplet" on page 153
- "ActiveBusComp" on page 153
- "ActiveBusObject" on page 153
- "ActiveViewName" on page 154
- "Attach" on page 155
- "CurrencyCode" on page 155
- "Detach" on page 156
- "EnableExceptions" on page 156
- "FindApplet" on page 157
- "GetBusObject" on page 157
- "GetDataSource" on page 158
- "GetLastErrCode" on page 159
- "GetLastErrText" on page 160
- "GetProfileAttr" on page 160
- "GetService" on page 161
- "GetSharedGlobal" on page 162

- "GotoView" on page 163
- "InvokeMethod" on page 164
- "LoadObjects" on page 165
- "Login" on page 167
- "LoginId" on page 169
- "LoginName" on page 170
- "Logoff" on page 170
- "LookupMessage" on page 171
- "LookupValue" on page 172
- "Name" on page 172
- "NewPropertySet" on page 173
- "PositionId" on page 173
- "PositionName" on page 174
- "RaiseError" on page 175
- "RaiseErrorText" on page 176
- "Release" on page 178
- "SetPositionId" on page 178
- "SetPositionName" on page 179
- "SetProfileAttr" on page 179
- "SetSharedGlobal" on page 180
- "SWEAlert" on page 181
- "Trace" on page 182
- "TraceOff" on page 184
- "TraceOn" on page 185

ActiveApplet

ActiveApplet returns a reference to the applet that currently has focus.

Syntax theApplication().ActiveApplet();

Argument	Description
Not applicable	

Returns The name of the applet instance that has focus

Usage Use this method to determine which applet currently has focus. The applet typically has a blue border to show that it is active.

Used With Browser Script

ActiveBusComp

ActiveBusComp returns the business component associated with the active applet.

Syntax theApplication().ActiveBusComp();

Argument	Description
Not applicable	

Returns The business component associated with the active applet

Used With Browser Script

ActiveBusObject

ActiveBusObject returns the business object for the business component of the active applet.

Syntax *Application*.ActiveBusObject

Argument	Description
Not applicable	

Returns The business object that contains the business component associated with the active applet

Usage Do not use ActiveBusObject in any event handler that may be initiated by the COM Data Server, COM Data Control, CORBA Object Manager, or Java Data Bean.

Used With Browser Script, Mobile/Dedicated Web Client Automation Server, Server Script

Example Here is a Browser Script example:

```
var oBusObj;

oBusObj = TheApplication().ActiveBusObject()
TheApplication().SWEAlert("The active business object is " +
oBusObj.Name() + ".")
```

ActiveViewName

ActiveViewName returns the name of the active view.

Syntax *Application*.ActiveViewName

Argument	Description
Not applicable	

Usage Do not use the ActiveViewName method in any event handler that may be initiated by the COM Data Server, COM Data Control, CORBA Object Manager, and Java Data Bean.

Returns A string containing the active view name

Used With Browser Script, Mobile/Dedicated Web Client Automation Server, Server Script

Attach

The Attach method allows an external application to re-connect to an existing Siebel session.

Syntax Application.Attach(*sessionString*)

Argument	Description
sessionString	A string containing the Siebel Session Id. The sessionString is typically the output of the Detach method or a value returned from the Siebel cookie.

Returns Boolean indicating whether or not the method was successfully executed

Used With COM Data Control, Java Data Bean

NOTE: The Attach method requires that Resonate Central Dispatch be installed. For more information on installing Resonate Central Dispatch, read *Siebel Server Installation Guide for Microsoft Windows* or *Siebel Server Installation Guide for UNIX*.

CurrencyCode

CurrencyCode returns the operating currency code associated with the division to which the user's position has been assigned.

Syntax *Application*.CurrencyCode

Argument	Description
Not applicable	

Returns A string containing the currency code; for example, USD for U.S. dollars, EUR for the euro, JPY for the Japanese yen.

Used With Browser Script, COM Data Control, COM Data Server, CORBA Object Manager, Web Client Automation Server, Server Script

Detach

The Detach method returns a string containing the Siebel session Id.

Syntax Application.Detach

Argument	Description
Not applicable	

Usage The string returned by the Detach method should only be used with the Attach method.

Returns String containing the Siebel session Id

Used With COM Data Control, Java Data Bean

NOTE: The Detach method requires that Resonate Central Dispatch be installed. For more information on installing Resonate Central Dispatch, read *Siebel Server Installation Guide for Microsoft Windows* or *Siebel Server Installation Guide for UNIX*.

EnableExceptions

The EnableExceptions method enables or disables native COM error handling.

Syntax *ApplicationControl*.EnableExceptions(*bEnable*)

Argument	Description
bEnable	A Boolean: TRUE or FALSE

Returns Not applicable

Usage Setting the argument to TRUE enables native error handling. This allows applications to intercept and display the exception ID and description. Native COM error handling is disabled by default.

Used With COM Data Control, Mobile/Dedicated Web Client Automation Server

FindApplet

FindApplet returns the applet that is identified by the *appletName* argument.

Syntax *Application*.FindApplet(*appletName*)

Argument	Description
appletName	String variable or literal containing the name of the desired applet.

Returns The applet identified in *appletName*

Usage The only applets available are applets visible in the active view.

Used With Browser Script

GetBusObject

GetBusObject method instantiates and returns a new instance of the business object specified in its argument.

Syntax *Application*.GetBusObject(*busObjectName*)

Argument	Description
busObjectName	String variable or literal containing the name of the business object to instantiate.

Returns The business object instance specified in the argument

Usage Set the business object to Nothing to destroy the instantiated business object after it is no longer needed.

Used With COM Data Control, COM Data Server, CORBA Object Manager, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script

Example Here is a Siebel VB example:

```
Dim oActionBO as BusObject
Set oActionBO = theApplication.GetBusObject("Action")
Set oActionBO = Nothing
```

GetDataSource

Returns the name of the data source, as defined in the CFG file, that is being used for the session.

Syntax dataSrc = TheApplication().InvokeMethod("GetDataSource")

Argument	Description
none	

Returns A string containing the value of the data source currently used by the application.

Used With COM Data Control, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script

Example The following eScript example detects the data source and displays its name in a dialog box.

```
var dataSrc = TheApplication().InvokeMethod("GetDataSource");
TheApplication().RaiseErrorText(dataSrc);
```

GetLastErrCode

The GetLastErrCode method returns the last error execution status.

Syntax *Application*.GetLastErrCode

Argument	Description
Not applicable	

Returns A short integer containing the last error execution status: 0 indicates no error.

Usage After execution of a method, the GetLastErrCode can be invoked to check if any error was returned from the previous operation. GetLastErrText method can be invoked to retrieve the text of the error message. Each method invocation resets the execution status.

Used With COM Data Control, Mobile/Dedicated Web Client Automation Server, Web Client Automation Server

Example Here is a COM Data Control example:

```
errcode = SiebelApplication.GetLastErrCode
If errcode <> 0 Then
   ErrText = SiebelApplication.GetLastErrText
   TheApplication().RaiseErrorText(ErrText);
   Exit Sub
End If
```

See Also "GetLastErrText" on page 160

GetLastErrText

The GetLastErrText method returns the last error text message.

Syntax *Application*.GetLastErrText

Argument	Description
Not applicable	

Returns The last error text message as a string

Used With COM Data Control, COM Data Server, CORBA Object Manager, Mobile/Dedicated Web Client Automation Server, Web Client Automation Server

Example Here is an example using the COM Data Control:

```
errcode = SiebelApplication.GetLastErrCode
If errcode <> 0 Then
   ErrText = SiebelApplication.GetLastErrText
   TheApplication().RaiseErrorText(ErrText);
   Exit Sub
End If
```

See Also "GetLastErrCode" on page 159

GetProfileAttr

GetProfileAttr returns the value of an attribute in a user profile.

Syntax GetProfileAttr(*name*)

Parameter	Description
name	A string indicating the name of the attribute

Returns The value of the attribute *name*

Usage GetProfileAttr is used in personalization to retrieve values of attributes in a user profile.

Used With Browser Script, COM Data Control, COM Data Server, CORBA Object Manager, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script

See Also "SetProfileAttr" on page 179

GetService

The GetService method returns a specified service. If the service is not already running, it is constructed.

Syntax *Application*.GetService(*serviceName*)

Parameter	Description
serviceName	The name of the service to start

Returns A reference to the requested business service

Usage This method finds the business service indicated by *serviceName*; it constructs the service if it is not already running. It first searches through the built-in services that are stored in the repository. If the service is not found, GetService searches through services defined in the run-time Business Services table.

A business service is normally deleted from memory as soon as every reference to it, such as local or global variables, are cleared by setting them to another value. However, if the Cache flag on the business service is set, the service remains in memory as long as the Siebel application is running.

To invoke a business service using the Web Client Automation Server and Browser Script, the business service must first be registered in the application configuration file (such as uagent.cfg, sfs.cfg, and so on). This prevents Service Not Found errors. To register a business service in the application configuration file, navigate to the [SWE] section, and add entries like the following examples.

```
ClientBusinessService0 = "XML Converter"
ClientBusinessService1 = "Siebel Account"
```

ClientBusinessService entries must be sequential, starting at 0 and incrementing by 1.

Used With Browser Script, COM Data Control, COM Data Server, CORBA Object Manager, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script, Web Client Automation Server

Example For an example, read "SetProperty" on page 294.

GetSharedGlobal

Shared global variables are unique to the user and the user's associated session. One user's global variables are not visible to other users. The variables are global to the current user and session only. The GetSharedGlobal method gets the shared user-defined global variables.

Syntax *Application*.GetSharedGlobal(*varName*)

Argument	Description
varName	String literal or variable containing the name of the global variable

Returns A string containing the user-defined global variables

Usage

```
GetSharedGlobal("varName")
```

retrieves the string set by:

```
SetSharedGlobal "varName", "stringValue".
```

Used With COM Data Control, COM Data Server, CORBA Object Manager, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script

Example Here is a Siebel VB example:

```
theApplication.SetSharedGlobal "myVar", "a value"
myVar2 = theApplication.GetSharedGlobal("myVar")
```

Here is a COM example:

```
oleVar = SiebelApplication.GetSharedGlobal("myVar", errCode)
SiebelApplication.SetSharedGlobal "myVar", "a value", errCode
```

See Also "SetSharedGlobal" on page 180

GotoView

GotoView activates the named view and its BusObject. As a side effect, this method activates the view's primary applet and its BusComp and activates the primary applet's first tab sequence control. Further, this method deactivates any BusObject, BusComp, applet, or control objects that were active prior to this method call.

Syntax *Application*.GotoView(*ViewName*[, *BusinessObjectName*])

Argument	Description
ViewName	The name of the view for the Siebel application to display
BusinessObjectName	An optional parameter to specify the business object to use for displaying the view. You cannot specify the current active business object as an argument to GotoView. If this argument is not supplied, or is specified as Nothing, a new business object is loaded in the normal fashion.

Returns Not applicable

Usage If a business object has not been instantiated, *BusinessObjectName* should have the value Nothing.

NOTE: The GotoView method is not supported in the Navigate and PreNavigate events.

An example use of the GotoView method is to programmatically navigate to the Opportunity List view.

```
theApplication.GotoView "Opportunity List View", Nothing
```

When this method is used from an external program, the usage is as follows:

```
ObjectReference.GotoView("Opportunity List View", Nothing,
errCode)
```

Alternatively, if your application has already instantiated an Opportunity object with the object reference of objOppty, the appropriate usage in Siebel VB is:

```
theApplication.GotoView"Opportunity List View", objOppty
```

When this method is used from an external program, the usage is as follows:

```
ObjectReference.GotoView("Opportunity List View", OpptyObj,
errCode)
```

NOTE: When this method is used in a Siebel VB or eScript script, regardless of where it appears in the script, it is executed last. However, when GotoView is invoked from an external program, it is executed immediately. The GotoView method goes to a new screen, with the result that the screen in which the method is executing is destroyed, and the executing script along with it, causing a crash. To avoid this behavior, make the call to GotoView the last executable statement in the script.

Used With Server Script

InvokeMethod

InvokeMethod calls a specialized method specified by its argument.

Browser Script Syntax

Application.InvokeMethod(*methodName, methodArgs_PropSet*);

Argument	Description
methodName	The name of the method.
methodArgs	One or more strings containing arguments to *methodName.*

Server Script Syntax

Application.InvokeMethod(*methodName, methodArgs*);

Argument	Description
methodName	The name of the method.
*methArg1, methArg2, …, methArg*N	One or more strings containing arguments to *methodName.*

Returns

In Server Script, returns a string containing the result of the method

In Browser Script, returns a Boolean

Usage

InvokeMethod allows you to call methods on an Application object that is exposed directly through the Application interface.

NOTE: The InvokeMethod method should be used only with documented specialized methods. Siebel Systems does not support calling specialized methods with InvokeMethod unless they are listed in this book.

Used With

Browser Script, COM Data Control, COM Data Server, CORBA Object Manager, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script

Example

For an example, read "InvokeMethod" on page 136.

LoadObjects

The LoadObjects method is used to start the COM Data Server object, and returns a reference to the Application object. This method must be the first call to the COM Data Server.

Syntax *Application*.LoadObjects(*pathName**CFGfileName*)

Argument	Description
pathName	Directory path to the configuration (CFG) file.
CFGfileName	Name of the CFG file to open.

Returns The Application object opened on start-up

Usage Prior to calling LoadObjects, you must change the current directory to the Siebel\bin directory.

Used With COM Data Server

Example Here is a COM Data Server example.

```
Private Sub LoadConfig_Click()
   Dim errCode As Integer
   LoadConfig.Enabled = False
   SiebelApplication.LoadObjects "C:\siebel\bin\uagent.cfg", _
      errCode

   If errCode = 0 Then
      ConfigOK = 1
   End If

   Status.Text = SiebelApplication.GetLastErrText

End Sub
```

LoadUserAttributes

The LoadUserAttributes method loads a user profile into the session.

Syntax LoadUserAttributes(*string*)

Argument	Description
row-id	The row-id of the person whose profile needs to be loaded.

Returns Not applicable

Usage If this function is called with no parameter, it unloads the loaded user profile. This loaded profile can be accessed as the "You" profile from personalization rules. For more information, read *Personalization Administration Guide*

Used With Server Script

Example The following VB example shows a method that loads a user profile into the session. The function is exposed on the Siebel Application Object.

```
Function LoadUserProfile As Integer
TheApplication.InvokeMethod ("LoadUserAttributes",0-10N07)
End Function
```

This function has only one parameter: the row-id of the person whose profile needs to be loaded. If this function is called with empty parameters, it unloads the loaded user profile.

```
Function LoadUserProfile As Integer
TheApplication.InvokeMethod ("LoadUserAttributes", “”)
End Function
```

Login

The Login method allows external applications to log in to the COM Data Server, COM Data Control, Java Data Bean, or CORBA Object Manager and access the Siebel objects. The Login method allows the end user to invoke the Siebel application without being prompted for a login and password. The Login method determines the privileges granted, and the role and responsibility of the end user for that session.

Syntax *Application*.Login([*connectString*,] *userName*, *password*)

Argument	Description
connectString	Token-based connect string
userName	Username for login
password	User password for login

Returns A string containing the error code

Usage Verify that the Siebel\bin directory is the current directory. To access the Data Control, make sure the default Data Source points to the database that you wish to access and set EnableOLEAutomation to TRUE in your CFG file (this is the default value for the parameter). To access the CORBA Object Manager, EnableCORBA must be set to TRUE.

For information on formatting the connect string, read "Connect String" on page 112.

Used With COM Data Control, COM Data Server, CORBA Object Manager, Java Data Bean

Example The Connect string for the COM Data Control is token-based; for example:

```
host = "Siebel://my_computer/SIEBEL/objsrvr/my_computer" lang =
"ENU"
```

Because most languages use quotes to enclose text strings, you must use quotes inside quotes; for example:

To use the COM Data Control in Visual Basic:

```
m_dataBean.login("siebel.tcpip.none.none://gateway:gatewayport/
enterpriseserver/SCCObjMgr/siebelserver", "username",
"password");
```

To use the COM Data Control in C++:

```
Login("host=\"siebel//:my_computer/SIEBEL/objsvr/my_computer\"
lang = \"ENU\"","user","password");
```

The following code sample illustrates how to log in to the server and check for errors.

```
   Call SiebelAppControl.Login("host=""siebel://gtwy/enterprise/
ObjMgr/SiebSrvr""", "SADMIN", "SADMIN")

   //Check for errors
      If SiebelAppControl.GetLastErrCode <> 0 Then
         frmMain.txtStatus.Text = SiebelAppControl.GetLasErrText
      Else
         frmMain.txtStatus.Text = "Connected successfully..."
      End If
```

LoginId

The LoginId method returns the login ID of the user who started the Siebel application.

Syntax *Application*.LoginId

Argument	Description
Not applicable	

Returns A string containing the login ID

Usage The login ID is the row ID of the user's login in the Employee table. Once obtained, the login ID can be conveniently used as a search specification.

Used With COM Data Control, COM Data Server, CORBA Object Manager (implemented as an attribute), Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script

Example In this Siebel VB example of BusComp_PreSetFieldValue, the LoginId method is used to determine whether the user has the right to modify a record.

```
Function BusComp_PreSetFieldValue (FieldName As String,
         FieldValue As String) As Integer
   Dim iReturn as integer
   iReturn = ContinueOperation
   Select Case FieldName
      Case "Account Status"
         if Me.GetFieldValue("Created By") <> _
            TheApplication.LoginId then
            TheApplication.RaiseErrorText("*** You cannot change
Account Status _
               because you did not create the record***")
            iReturn = CancelOperation
         end if
   End Select
   BusComp_PreSetFieldValue = iReturn
End Function
```

LoginName

The LoginName method returns the login name of the user who started the Siebel application (the name typed in the login dialog box).

Syntax *Application*.LoginName

Argument	Description
Not applicable	

Returns A string containing the user's login name

Used With COM Data Control, COM Data Server, CORBA Object Manager (implemented as an attribute), Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script

Example For examples, read "ExecuteQuery" on page 206 and "TheApplication" on page 329.

See Also "Login" on page 167

Logoff

The Logoff method disconnects the client from the server.

Syntax *Application*.Logoff

Argument	Description
Not applicable	

Returns Not applicable

Usage For clients with user interfaces, Logoff destroys every window except for the topmost window. Logoff also deletes every object, except for the topmost object, on both client and server. Logoff is usually used to disconnect from the server and then reconnect as a different user.

Logoff is called automatically if you destroy the main object.

Used With COM Data Control, CORBA Object Manager, Java Data Bean, Mobile/Dedicated Web Client Automation Server

LookupMessage

The LookupMessage method returns the translated string for the specified key, in the current language, from the specified category. The optional parameters are used to format the string if it contains any substitution parameters (%1,%2).

Syntax TheApplication().LookupMessage (*category, key, [param1], [param2],...., [paramN]*)

Parameter	Description
Category	Name of the Message Category object, as defined in Siebel Tools, that is the parent of Key value.
Key	Name of the Message object, as defined in Siebel Tools, whose text contains the value to be investigated.
param1, param2, ..., paramN	Optional parameters used to format the error message if it contains any substitution parameters (%1, %2).

Returns A string containing the localized message text.

Usage Useful for retrieving locale specific custom error messages.

Used With Server Script

LookupValue

Finds a row in S_LST_OF_VAL where the TYPE column matches the type argument, the CODE column matches the lang_ind_code argument, and the LANG_ID column matches the language code of the currently active language. This function is used to obtain the translation of the specified untranslated value in the specified LOV into the currently active language.

Syntax val = TheApplication().InvokeMethod("LookupValue", type, lang_ind_cd)

Argument	Description
type	Type as specified in the List of Values administration view.
lang_ind_cd	Language independent code value as specified in the List of Values administration view.

Returns Returns a string containing the display value (the VAL column) for the row.

Used With COM Data Control, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script

Example The following eScript example returns the display value for currently active language for the account role of "Consultant".

```
var LOVText=TheApplication().InvokeMethod
("LookupValue","ACCOUNT_ROLE","Consultant");
```

Name

The Name method returns name of the application.

Syntax Application.Name()

Argument	Description
Not applicable	

Returns A string containing the name of the application

Used With Browser Script, Web Client Automation Server

NewPropertySet

The NewPropertySet method constructs a new property set object.

Syntax *Application*.NewPropertySet()

Parameter	Description
Not applicable	

Returns A property set

Usage NewPropertySet is used primarily to construct input and output arguments for business services.

NOTE: When using NewPropertySet on an existing PropertySet object, old references to this PropertySet are lost. When reusing a PropertySet, use the Reset method on the PropertySet itself.

Used With Browser Script, COM Data Control, COM Data Server, Java Data Bean, Mobile/ Dedicated Web Client Automation Server, Server Script, Web Client Automation Server

Example For an example, read "SetProperty" on page 294.

PositionId

The PositionId property returns the position ID (ROW_ID from S_POSTN) of the user's current position. This is set by default when the Siebel application is started and may be changed (through Edit > Change Position) if the user belongs to more than one position.

Syntax *Application*.PositionId

Argument	Description
Not applicable	

Returns A string row ID

Used With COM Data Control, COM Data Server, CORBA Object Manager (implemented as an attribute), Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script

PositionName

The PositionName property returns the position name of the user's current position. This is set by default when the Siebel application is started.

Syntax *Application*.PositionName

Argument	Description
Not applicable	

Returns A string containing the user's position

Used With COM Data Control, COM Data Server, CORBA Object Manager (implemented as an attribute), Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script

Example This Siebel VB example checks for the position of a user changing the sales stage, and prevents changes if the user is not of the appropriate position.

```
Function BusComp_PreSetFieldValue (FieldName As String, FieldValue
As String) As Integer

Dim sPosName As String sMsgText As String
Dim iReturn As Integer
iReturn = ContinueOperation
```

```
Select Case FieldName
   Case "Sales Stage"
      If FieldValue = "Approved" Then
         ' Do not allow the sales cycle to be changed to
         ' this value if the User is not a manager or VP.
         sPosName = TheApplication.PositionName
         If NOT ((PosName="Manager") OR (PosName="VP"))Then
            TheApplication.RaiseErrorText("Only a Manager or Vice
President can approve _
            a Pipeline Item. Please notify your Manager that you _
            want to have this Pipeline item approved.")
         iReturn = CancelOperation
         End If
   BusComp_PreSetFieldValue = iReturn
End Select

End Function
```

RaiseError

The RaiseError method raises a scripting error message to the browser. The error code is a canonical number. The error text is based on the specified key, looked up for the current language from the User-Defined Errors category. You can define these errors in Tools using the Message Category object. The optional parameters are used to format the string if it contains any substitution parameters (%1, %2).

Syntax Application.RaiseError((" < *key* > ", [*param1*], [*param2*],...., [*paramN*]))

Parameter	Description
Key	
param1, param2, ..., paramN	Optional parameters used to format the error message if it contains any substitution parameters (%1, %2).

Returns Not applicable

Usage When invoked, the RaiseError method causes execution of the script to terminate, and sends a notification to the browser.

Internally, the RaiseError/RaiseErrorText methods raise a Server Script exception. Therefore, if you have implemented error handling in your scripts, please note that the error handling can suppress RaiseError/RaiseErrorText functionality. In the following eScript example, the RaiseErrorText results in a scripting execution being raised, transferring control to the catch statement. To display the error message, the error must be thrown using the throw statement.

```
function BusComp_PreDeleteRecord ()
{
   try {
      var status = this.GetFieldValue("Account Status");

      if (status == "Gold") {
         TheApplication().RaiseError (<user defined error name>);
         return (CancelOperation);

      }
      else {
         return (ContinueOperation);
      }
   }
   catch (e) {
      throw e;
   }

}
```

If you have implemented error handling in Siebel VB, remember that if you are using “On Error Goto …”, the RaiseError and RaiseErrorText methods results in the script transferring execution to the error handler. “On Error Resume Next” suppresses the RaiseError and RaiseErrorText methods.

Used With Server Script

RaiseErrorText

The RaiseErrorText method raises a scripting error message to the browser. The error text is the specified literal string. The optional parameters are used to format the string if it contains any substitution parameters (%1, %2).

Syntax TheApplication().RaiseErrorText(*value*, [*param1*], [*param2*],...., [*paramN*])

Parameter	Description
value	
param1, param2, ..., paramN	Optional parameters used to format the error message if it contains any substitution parameters (%1, %2).

Returns Not applicable

Usage When invoked, the RaiseErrorText method stops execution of the script.

Internally, the RaiseError/RaiseErrorText methods raise a Server Script exception. Therefore, if you have implemented error handling in your scripts, the error handling can suppress RaiseError/RaiseErrorText functionality. In the following eScript example, the RaiseErrorText results in a scripting execution being raised, transferring control to the catch statement. For the error message to be displayed, the error must be thrown using the throw statement.

```
function BusComp_PreDeleteRecord ()
{
   try {
      var status = this.GetFieldValue("Account Status");

      if (status == "Gold") {
         TheApplication().RaiseErrorText("Unable to delete Gold Account");
         return (CancelOperation);

      }
      else {
         return (ContinueOperation);
      }
   }
   catch (e) {
      throw e;
   }

}
```

If you have implemented error handling in Siebel VB and are using “On Error Goto ...”, the RaiseError and RaiseErrorText methods result in the script transferring execution to the error handler. “On Error Resume Next” suppresses the RaiseError and RaiseErrorText methods.

Used With Server Script

Release

The Release method destroys subordinate objects and returns their resources. This method invalidates every object reference to any subordinate objects held by the client. As a result, new references must be obtained to use any of these objects.

Syntax *Application.Release()*

Parameter	Description
Not applicable	

Returns Not applicable

Used With CORBA Object Manager

SetPositionId

SetPositionID sets the active position to the Position Id specified in the argument.

Syntax Application.SetPositionId(positionId)

Parameter	Description
positionId	A string containing the Position Id you would like to change to

Returns A Boolean denoting whether or not the operation was successfully completed

Usage When invoking the SetPositionId method, the "positionId" parameter must contain a Position Id that has already been associated with the current, logged-in user.

Used With COM Data Server, COM Data Control, CORBA Object Manager, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script

SetPositionName

SetPositionName sets the active position to the position name specified in the argument. Returns a Boolean indicating whether or not method succeeded.

Syntax *Application.SetPositionName(positionName)*

Parameter	Description
positionName	A string containing the name of the position.

Returns A Boolean denoting whether or not the operation was successfully completed

Usage When invoking the SetPositionName method, the "positionName" parameter must contain a Position name that has already been associated with the current, logged-in user.

Used With COM Data Server, COM Data Control, CORBA Object Manager, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script

SetProfileAttr

SetProfileAttr is used in personalization to assign values to attributes in a user profile.

Syntax SetProfileAttr *name*, *value*

Parameter	Description
name	A string indicating the name of the attribute
value	The value of *name*

Returns Not applicable

Usage SetProfileAttr assigns the value *value* to the attribute in a user profile indicated by *name*. If the profile attribute specified in the parameter string already exists, the corresponding persistent profile attribute in the application is updated with the new value. If the profile attribute specified in the parameter string does not exist in the list of persistent profile attributes, it is created as a dynamic profile attribute, without quotation marks encompassing the name.

Used With Browser Script, COM Data Control, COM Data Server, Server Script, CORBA Object Manager, Java Data Bean, Mobile/Dedicated Web Client Automation Server

See Also "Name" on page 172. For more information on user profile attributes, read *Applications Administration Guide*.

Example Here is an example for Browser Script:

```
function Applet_PreInvokeMethod (name, inputPropSet)

{
   if (name == "hobbyReq") {
      var hobby = theApplication().GetProfileAttr("Hobby");


      if (hobby == "") {
         hobby = prompt("Please enter your favorite
hobby");
         theApplication().SetProfileAttr("Hobby", hobby);
      }
      return ("CancelOperation");
   }
   else
      return ("ContinueOperation");
}
```

SetSharedGlobal

Shared global variables are unique to the user and the user's associated session. One user's global variables are not visible to other users. The variables are global to the current user and session only. The SetSharedGlobal property sets a shared user-defined global variable, which may be accessed using GetSharedGlobal.

Syntax *Application*.SetSharedGlobal(*varName*, *value*)

Argument	Description
varName	String variable or literal containing the name of the shared global variable to set
value	String variable or literal containing the value to set the variable to set

Returns Not applicable

Used With COM Data Control, COM Data Server, CORBA Object Manager, Mobile/Dedicated Web Client Automation Server, Server Script

Example In Siebel VB:

```
theApplication.SetSharedGlobal "myVar", "FOO"
myVar2 = theApplication.GetSharedGlobal("myVar")
```

In COM:

```
comVar = SiebelApplication.GetSharedGlobal("myVar", errCode)
SiebelApplication.SetSharedGlobal "myVar", "BLAH", errCode
```

See Also "GetLastErrCode" on page 159

SWEAlert

SWEAlert displays a modal dialog box containing a message to the user.

Syntax TheApplication().SWEAlert(message)

Returns Undefined (similar to returning nothing)

Usage Use SWEAlert instead of Alert. With Alert, popup applets such as Mvg and Pick applets are hidden (sent to the background) when a JavaScript Alert() is raised by a Browser side event. With SWEAlert, the dialog's parent applet is not sent to the foreground.

Used With Browser Script

Example The following example displays a status message to the user.

```
function BusComp_PreSetFieldValue (fieldName, value) {

   if (fieldName == "Account Status") {
      var cVolume = this.GetFieldValue("Current Volume");

      if ((value == "Inactive") && (cVolume > 0)) {
         TheApplication().SWEAlert("Unable to inactivate an
account that has a current volume greater than 0");

         return ("CancelOperation");
      }
      else
         return ("ContinueOperation");
   }
   else
      return ("ContinueOperation");
}
```

Trace

The Trace method appends a message to the trace file. Trace is useful for debugging SQL query execution. This tracing is not the same as the tracing that can be activated is the application's CFG file. For more information, read "Script Tracing" on page 41.

Syntax *Application*.Trace(*message*)

Argument	Description
message	String variable or literal containing message text to append to the trace file

Returns Not applicable

Used With COM Data Control, COM Data Server, CORBA Object Manager, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script

Example Here is a COM Data Server example:

```
Private Sub TraceOn_Click()
   Dim ErrCode As Integer
   SiebelApplication.TraceOn "c:\temp\trace.txt", "allocation",
      "all",       ErrCode
   If (ErrCode = 0) Then SiebelApplication.TraceOn
      "c:\temp\trace.txt",      "SQL", "",ErrCode
   If (ErrCode = 0) Then SiebelApplication.Trace
      "Start of Tracing!",
      ErrCode
End Sub
```

Here is a Siebel VB example:

```
Sub Button2_Click
   theApplication.TraceOn "C:\temp\trace.txt", "allocation",
      "all"
   theApplication.TraceOn "C:\temp\trace.txt", "sql", ""
   theApplication.Trace "start of tracing!"
End Sub
```

Sample output of an Allocation trace section:

```
03/05/98,17:27:47,START,4.0.4 [1425_P3] ENU
03/05/98,17:27:47,ALLOC,1,BusObject,Account,Basic
03/05/98,17:27:48,ALLOC,2,BusComp,Account,Basic
03/05/98,17:27:48,RELEASE,1
03/05/98,17:27:48,RELEASE,2
```

Sample output of a SQL trace section:

```
01/22/98,21:03:49,START,4.0.2 [1416] ENU
01/22/98,21:04:02,COMMENT,Start of Tracing!
01/22/98,21:04:10,SQLSTMT,1,SELECT,"SELECT
   T1.ROW_ID,
   T1.MODIFICATION_NUM,
   T1.CREATED_BY,
   T1.LAST_UPD_BY,
   T1.CREATED,
   T1.LAST_UPD,
   T1.CONFLICT_ID,
   T1.NAME,
   T1.DESC_TEXT,
   T1.PRIV_FLG,
   T1.QUERY_STRING
FROM
   DEV32.S_APP_QUERY T1
WHERE
```

```
      (T1.CREATED_BY = :1 OR T1.PRIV_FLG = :2) AND
      ((T1.NAME LIKE :3 OR T1.NAME LIKE :4 OR T1.NAME LIKE :5 OR
         T1.NAME LIKE :6) AND UPPER(T1.NAME) = UPPER(:7))
      ORDER BY
         T1.NAME, T1.DESC_TEXT"
   01/22/98,21:04:10,SQLBIND,1,1,1-6NF
   01/22/98,21:04:10,SQLBIND,1,2,N
   01/22/98,21:04:10,SQLBIND,1,3,ac%
   01/22/98,21:04:10,SQLBIND,1,4,Ac%
   01/22/98,21:04:10,SQLBIND,1,5,aC%
   01/22/98,21:04:10,SQLBIND,1,6,AC%
   01/22/98,21:04:10,SQLBIND,1,7,Account
```

See Also

"TraceOff"
"TraceOn" on page 185

TraceOff

TraceOff turns off the tracing started by the TraceOn method.

Syntax

Application.TraceOff

Argument	Description
Not applicable	

Returns

Not applicable

Used With

COM Data Control, COM Data Server, CORBA Object Manager, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script

Example

This Siebel VB example sets the value in the Sales Stage field to the default, that is, to the first value in the field's picklist, and uses tracing to track the result.

```
Sub BusComp_NewRecord
   TheApplication.TraceOn "C:\lvpick.doc", "SQL", ""
   Dim oBC as BusComp
   set oBC = me.GetPickListBusComp("Sales Stage")
```

```
   With oBC
      .SetViewMode AllView
      .ClearToQuery
      .ActivateField "Sales Stage Order"
      .SetSortSpec "Sales Stage Order"
      .ExecuteQuery ForwardOnly
      if .FirstRecord then
         .Pick
      end if
   End With

   set oBC = Nothing

   TheApplication.TraceOff

End Sub
```

TraceOn

TraceOn turns on the tracking of allocations and deallocations of Siebel objects and SQL statements generated by the Siebel application.

Syntax *Application*.TraceOn(*filename, type, selection*)

Argument	Description

filename	Output filename for the trace messages. If this parameter is not specified, tracing information is logged to the Object Manager log file for that user session.
	
	The filename parameter can take two additional inline parameters: $p and $t. The $p parameter substitutes the process id to the filename, and $t substitutes the thread id to the file name. For example:
	
	`TheApplication().TraceOn("d:\\temp\\trace_$p_$t.txt", "Allocation", "All");`
	
	would log trace files to `d:\temp\trace\trace_1496_1412.txt`. Place a separator between the $p and $t parameters to make sure that the filename parameter is unique. For example, if user A had a process id of 1 and a thread of 12 without using a separator, the tracing file would be
	
	`d:\temp\trace_112.txt`
	
	If user B had a process id of 11, and a thread id of 2, their tracing file would be
	
	`d:\temp\trace_112.txt`
	
	As a result, both users would attempt to log to the same file. Adding a separator between the process and thread id keeps the filenames unique:
	
	`d:\temp\trace_1_12.txt`
	
	`d:\temp\trace_11_2.txt`

Argument	Description
type	Specifies the type of tracing to start. This can have the following values: ■ **Allocation.** Traces allocations and deallocations of Siebel objects. This option is useful if you suspect memory leaks in your code. ■ **SQL.** Traces SQL statements generated by the Siebel application.
selection	Indicates which Siebel objects should be traced for the Allocation trace type. This argument should be "" if the trace type is SQL. ■ **Script.** Traces VB and eScript objects. ■ **OLE.** Traces allocations for data server or automation server programs. ■ **All.** Traces all objects. The All value does not trace the Siebel objects managed implicitly by Siebel's declarative configuration use. *All* traces the Siebel objects constructed by scripting. ■ **CORBA.** Traces allocation for CORBA object manager programs.

Returns Not applicable

Usage Always issue TraceOff to turn off tracing. If you attempt to call TraceOn with a different filename without calling TraceOff first, trace information is written to the new trace filename. You can issue multiple TraceOn statements to the same trace file.

Used With COM Data Control, COM Data Server, Server Script, CORBA Object Manager, Java Data Bean, Mobile/Dedicated Web Client Automation Server

Example Sample code for COM Data Server:

```
Private Sub TraceOn_Click()
   Dim ErrCode As Integer
   SiebelApplication.TraceOn "c:\temp\trace.txt", "allocation",
      "all",       ErrCode
   If (ErrCode = 0) Then SiebelApplication.TraceOn
      "c:\temp\trace.txt",      "SQL", "",ErrCode
   If (ErrCode = 0) Then SiebelApplication.Trace
      "Start of Tracing!",
      ErrCode
End Sub
```

Sample code for eScript:

```
function BusComp_PreSetFieldValue (FieldName, FieldValue)

{
TheApplication().TraceOn("d:\\temp\\trace.txt", "Allocation",
"All");
TheApplication().TraceOn("d:\\temp\\trace.txt", "SQL", "");
TheApplication().Trace("start tracing!");

return (ContinueOperation);
}
```

Sample code for Siebel VB:

```
Sub Button2_Click
   theApplication.TraceOn "C:\temp\trace.txt", "allocation",
      "all"
   theApplication.TraceOn "C:\temp\trace.txt", "sql", ""
   theApplication.Trace "start of tracing!"
End Sub
```

Sample output of an Allocation trace section:

```
03/05/98,17:27:47,START,4.0.4 [1425_P3] ENU
03/05/98,17:27:47,ALLOC,1,BusObject,Account,Basic
03/05/98,17:27:48,ALLOC,2,BusComp,Account,Basic
03/05/98,17:27:48,RELEASE,1
03/05/98,17:27:48,RELEASE,2
```

Sample output of a SQL trace section:

```
01/22/98,21:03:49,START,4.0.2 [1416] ENU
01/22/98,21:04:02,COMMENT,Start of Tracing!
01/22/98,21:04:10,SQLSTMT,1,SELECT,"SELECT
   T1.ROW_ID,
   T1.MODIFICATION_NUM,
   T1.CREATED_BY,
   T1.LAST_UPD_BY,
   T1.CREATED,
   T1.LAST_UPD,
   T1.CONFLICT_ID,
   T1.NAME,
   T1.DESC_TEXT,
   T1.PRIV_FLG,
   T1.QUERY_STRING
FROM
```

```
   DEV32.S_APP_QUERY T1
WHERE
   (T1.CREATED_BY = :1 OR T1.PRIV_FLG = :2) AND
   ((T1.NAME LIKE :3 OR T1.NAME LIKE :4 OR T1.NAME LIKE :5 OR
      T1.NAME LIKE :6) AND UPPER(T1.NAME) = UPPER(:7))
   ORDER BY T1.NAME, T1.DESC_TEXT"
01/22/98,21:04:10,SQLBIND,1,1,1-6NF
01/22/98,21:04:10,SQLBIND,1,2,N
01/22/98,21:04:10,SQLBIND,1,3,ac%
01/22/98,21:04:10,SQLBIND,1,4,Ac%
01/22/98,21:04:10,SQLBIND,1,5,aC%
01/22/98,21:04:10,SQLBIND,1,6,AC%
01/22/98,21:04:10,SQLBIND,1,7,Account
```

See Also "Trace" on page 182
"TraceOff" on page 184

Application Events

The following topics describe application events.

- "Application_Close"
- "Application_InvokeMethod" on page 190
- "Application_Navigate" on page 191
- "Application_PreInvokeMethod" on page 191
- "Application_PreNavigate" on page 193
- "Application_Start" on page 194

Application_Close

The Close event is called before the application exits. This allows Basic scripts to perform last-minute cleanup (such as cleaning up a connection to a COM server). It is called when Windows notifies the application that it should close, but not if the process is terminated directly.

Syntax Application_Close

Argument	Description
Not applicable	

Returns Not applicable

Used With Server Script

NOTE: Siebel Business Processes invokes this event. For more information, read *Siebel Business Process Designer Administration Guide*.

Application_InvokeMethod

The Application_InvokeMethod event is called after a specialized method is invoked.

Server Script Syntax Application_InvokeMethod(*methodName*)

Argument	Description
methodName	Name of the method invoked

Browser Script Syntax Application_InvokeMethod(*name, inputPropSet*)

Argument	Description
inputPropSet	A property set containing parameters to be passed to the InvokeMethod event.

Usage The InvokeMethod event is called just after a specialized method is invoked on the application.

If you are implementing a new method (not defined by the built-in functions), the script should return CancelOperation to avoid invoking an *Unknown Method Name* error.

The Browser script implementation does not return a property set.

Returns Returns TRUE if the call succeeds or FALSE if it does not succeed.

Used With Browser Script, Server Script

See Also "How Your Script Affects Program Flow" on page 106
"Application_PreInvokeMethod"

Application_Navigate

The Application_Navigate event is called after the client has navigated to a view.

Syntax Application_Navigate

Argument	Description
Not applicable	

Returns Not applicable

Used With Server Script

Application_PreInvokeMethod

The PreInvokeMethod event is called before a specialized method is invoked by a user-defined applet menu or by calling InvokeMethod on the application.

Server Script Syntax Application_PreInvokeMethod(*methodName*)

Argument	Description
methodName	String variable or literal containing the name of the method invoked

Browser Script Syntax

Application_PreInvokeMethod (*methodName*, *inputPropSet*)

Argument	Description
methodName	String variable or literal containing the name of the method invoked.
inputPropSet	A property set containing parameters to be passed to the event.

Returns

"ContinueOperation" or "CancelOperation"

Usage

The PreInvokeMethod event is called just before a specialized method is invoked on the application. If implementing a user-defined method, the script should return CancelOperation if you wish to handle the event entirely through your own scripting.

Specialized methods are methods based on applet or business component classes other than CSSFrame and CSSBusComp, respectively, that is, specialized classes.

When the method to be invoked is part of an If statement, this function's return value must be assigned before the End If statement, as in the following code fragment:

```
If MethodName = "ResetQuery" then
   Application_PreInvokeMethod = CancelOperation
End If
```

CancelOperation stops the execution of the underlying Siebel code associated with the event. However, if there is code in the same script following CancelOperation, that code runs regardless of the CancelOperation.

Used With

Browser Script, Server Script

Example

Here is a sample implementation of the PreInvokeMethod:

```
Function Application_PreInvokeMethod (MethodName _
         As String) As Integer

   Dim i As Integer
   Dim iReturn As Integer
   iReturn = ContinueOperation
```

```
      Select Case MethodName
         Case "LaunchWord"
            i = Shell("C:\Program Files\Microsoft Office _
                  \Office\WINWORD.EXE",1)
            iReturn = CancelOperation

         Case "LaunchExcel"
            i = Shell("C:\Program Files\Microsoft Office _
                  \Office\EXCEL.EXE",1)
            iReturn = CancelOperation
      End Select

      Application_PreInvokeMethod = iReturn

   End Function
```

Here is the equivalent sample in Siebel eScript. Note that for this script to run, the entire Clib.system statement must appear on a single line in the Editor.

```
function Application_PreInvokeMethod (MethodName)

   var iReturn = ContinueOperation;

   switch (MethodName)
   {
      case "LaunchWord":
         Clib.system("\"C:\\Program Files\\Microsoft Office
\\Office\\WINWORD.EXE"",1);
         iReturn = CancelOperation;
         break;

      case "LaunchExcel":
         Clib.system("\"C:\\Program Files\\Microsoft Office
\\Office\\EXCEL.EXE"",1);
         iReturn = CancelOperation;
   }

   return (iReturn)
}
```

See Also "How Your Script Affects Program Flow" on page 106

Application_PreNavigate

The Application_PreNavigate event is called before the client navigates to a view.

Syntax Application_PreNavigate(DestViewName, DestBusObjName As String) As Integer

Argument	Description
DestViewName	Name of the View to which the user is navigating
DestBusObjName	Business object of the destination view

Returns CancelOperation or ContinueOperation

Used With Server Script

Example In the following code sample the script checks for the current BO (contact) and sets the current contact id as global variable (can be used for keeping context).

```
function Application_PreNavigate (DestViewName, DestBusObjName)
{
   try
   {
      var currentView = this.ActiveViewName();
      var BO = this.ActiveBusObject();
      if(BO.Name() == "Contact")
      {
         var BC = BO.GetBusComp("Contact");
         var id = BC.GetFieldValue("Id");
         TheApplication().SetSharedGlobal("ContactId", id);
      }
   }
   catch (e)
   {
      this.Trace("Exception caught: "+e.toString());
   }
   return (ContinueOperation);
}
```

Application_Start

The Start event is called when the client starts and again when the user interface is first displayed.

Syntax Application_Start(*commandline*)

Argument	Description
commandline	Text of the command line with which the Siebel application was started.

NOTE: Siebel Business Processes invokes this event. For more information, read *Siebel Business Process Designer Administration Guide*.

Returns Not applicable

Used With Server Script

Example This Siebel VB code should be placed in the Application_Start procedure for the application of your choice. This example retrieves the first and last name of the user logging into the Siebel application.

```
Sub Application_Start(CommandLine As String)

   Dim oEmpBusComp as BusComp
   Dim sLoginName as String
   Dim sUserName as String

   sLoginName = TheApplication.LoginName
   Set oEmpBusComp = oEmpBusObj.GetBusComp("Employee")

   With oEmpBusComp
      .ActivateField("Login Name")
      .ActivateField("First Name")
      .ActivateField("Last Name")
      .ClearToQuery
      .SetSearchSpec "Login Name", sLoginName
      .ExecuteQuery

      If .FirstRecord Then
         sUserName = .GetFieldValue("First Name")
         sUserName = sUserName + " " + _
            .GetFieldValue("Last Name")
      End If
   End With
```

```
    Set oEmpBusComp = Nothing
    Set oEmpBusObj = Nothing

End Sub
```

Business Component Methods

In the methods described in this section, the placeholder *oBusComp* refers to a BusComp variable.

- "ActivateField" on page 198
- "ActivateMultipleFields" on page 199
- "Associate" on page 201
- "BusObject" on page 202
- "ClearToQuery" on page 203
- "DeactivateFields" on page 204
- "DeleteRecord" on page 205
- "ExecuteQuery" on page 206
- "ExecuteQuery2" on page 209
- "FirstRecord" on page 210
- "FirstSelected" on page 211
- "GetAssocBusComp" on page 212
- "GetFieldValue" on page 214
- "GetFormattedFieldValue" on page 216
- "GetLastErrCode" on page 219
- "GetLastErrText" on page 219
- "GetMultipleFieldValues" on page 220

- "GetMVGBusComp" on page 220
- "GetNamedSearch" on page 222
- "GetPicklistBusComp" on page 222
- "GetSearchExpr" on page 224
- "GetSearchSpec" on page 225
- "GetUserProperty" on page 225
- "GetViewMode" on page 226
- "InvokeMethod" on page 228
- "LastRecord" on page 233
- "Name" on page 234
- "NewRecord" on page 234
- "NextRecord" on page 236
- "NextSelected" on page 237
- "ParentBusComp" on page 237
- "Pick" on page 238
- "PreviousRecord" on page 239
- "RefineQuery" on page 241
- "Release" on page 241
- "SearchExpr" on page 242
- "SetFieldValue" on page 243
- "SetFormattedFieldValue" on page 245
- "SetMultipleFieldValues" on page 247
- "SetNamedSearch" on page 248
- "SetSearchExpr" on page 249

- "SetSearchSpec" on page 251
- "SetSortSpec" on page 255
- "SetUserProperty" on page 257
- "SetViewMode" on page 258
- "UndoRecord" on page 262
- "ViewMode" on page 262
- "WriteRecord" on page 263

ActivateField

ActivateField allows queries to retrieve data for the argument-specified field.

Syntax

BusComp.ActivateField(*FieldName*)

Argument	Description
FieldName	String variable or literal containing the name of the field to activate

Returns

Not applicable

Usage

FieldName must be enclosed in double quotes and must be spelled exactly as the field name appears in Siebel Tools, using the same case. You must activate fields using ActivateField prior to executing a query for the business component.

NOTE: If you are writing an event handler on a business component, you must make sure that the field has already been activated by specifying the ForceActive user property on the control.

By default, fields are inactive except when:

- They are displayed on the applet and the business component is the instance on which the applet is based.

- They are System fields (which include Id, Created, Created By, Updated, and Updated By).
- Their ForceActive property is set to TRUE.
- The method ActivateField has been invoked with the *FieldName*.
- They have the Link Specification property set to TRUE.

After a business component has been executed, if additional fields are activated, the business component must be requeried before field values can be accessed. Failure to requery the business component results in a value of 0 being returned. The ActivateField method destroys the context of a query when it is used after the ExecuteQuery method.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example Here is a Siebel VB example. For an equivalent Siebel eScript example, read "ClearToQuery" on page 203.

```
Dim oEmpBusComp As BusComp
Dim sLoginName As String

Set oEmpBusObj = TheApplication.ActiveBusObject
Set oEmpBusComp = TheApplication.ActiveBusComp

oEmpBusComp.ActivateField("Login Name")
oEmpBusComp.SetViewMode AllView
oEmpBusComp.ClearToQuery
oEmpBusComp.SetSearchSpec "Login Name", sLoginName
oEmpBusComp.ExecuteQuery

Set oEmpBusComp = Nothing
```

See Also "DeactivateFields" on page 204

ActivateMultipleFields

Use ActivateMultipleFields to retrieve data for the fields specified in the property set.

Syntax Boolean ActivateMultipleFields(SiebelPropertySet sps)

Argument	Description
SiebelPropertySet	Property set containing a collection of properties representing the fields that are to be activated

Returns TRUE if success; FALSE if failure

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example Here is an example for Java Data Bean:

```
import com.siebel.data.*;
...
//Create Siebel Data Bean.
//login into Siebel Data Bean
...
//Create Siebel Bus Object.
//Get the Bus Object from SiebelDataBean
...
//Create Siebel Bus Comp siebBusComp
//Get the business component using SiebelBusObject

SiebelPropertySet ps = new SiebelPropertySet();
ps.setProperty("Account Products","");
ps.setProperty("Agreement Name","");
ps.setProperty("Project Name","");
ps.setProperty("Description","");
ps.setProperty("Name","");
siebBusComp.activateMultipleFields(ps);
...
```

See Also "SetMultipleFieldValues" on page 247
"GetMultipleFieldValues" on page 220

Associate

The Associate method creates a new many-to-many relationship for the parent object through an association business component (see GetAssocBusComp).

Syntax

BusComp.Associate(*whereIndicator*)

Argument	Description
whereIndicator	This argument should be one of the following predefined constants or the corresponding integer: NewBefore (0) or NewAfter (1), as in NewRecord.

Returns

Not applicable

Usage

To set field values on a child record that has been associated to a parent record, use the context of the MVGBusComp.

Used With

Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

See Also

"NewRecord" on page 234
"FirstSelected" on page 211
"GetMVGBusComp" on page 220

Example

The following VB example adds the position CEO to the current record of the parent business component, and updates the Opportunity Assignment Type field. The parent business component can be any business component that includes the Sales Rep multi-value group.

```
Dim oParentBC as BusComp
Dim oMvgBC as BusComp
Dim oAssocBC as BusComp

Set oParentBC = me.BusComp
Set oMvgBC = OpBC.GetMVGBusComp("Sales Rep")
Set oAssocBC = oMvgBC.GetAssocBusComp
```

```
With oAssocBC
   .SetSearchSpec "Id", newPosId
   .ExecuteQuery
   .Associate NewAfter
End With

oMvgBC.SetFieldValue "Opportunity Assignment Type", NewType

Set oAssocBC = Nothing
Set oMvgBC = Nothing
Set oParentBC = Nothing
```

Here is the equivalent example in Siebel eScript.

```
var oParentBC = this.BusComp();
var oMvgBC = oParentBC.GetMVGBusComp("Sales Rep");
var oAssocBC = oMvgBC.GetAssocBusComp();

with (oAssocBC)
{
   SetSearchSpec("Id", newPosId);
   ExecuteQuery();
   Associate(NewAfter);
}

oMvgBC.SetFieldValue("Opportunity Assignment Type", NewType);

oAssocBC = null;
oMvgBC = null;
oParentBC = null;
```

BusObject

The BusObject method returns the business object that contains the business component.

Syntax BusComp.BusObject

Argument	Description
Not applicable	

Returns The business object that contains the business component

Used With Browser Script, Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager (Implemented as an attribute)

Example For an example, read "GetViewMode" on page 226.

See Also "ActiveBusObject" on page 153

ClearToQuery

The ClearToQuery method clears the current query but does not clear sort specifications on the BusComp.

Syntax BusComp.ClearToQuery

Argument	Description
Not applicable	

Returns Not applicable

Usage Any fields to be queried must be activated before ClearToQuery. For more information, read "ActivateField" on page 198.

Search and sort specifications sent to the business component are cumulative; the business component retains and logically ANDs query qualifications since the last ClearToQuery, except for new search specifications on a field for which a search specification has previously been set. In that circumstance, the new specification replaces the old.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example Here is a Siebel eScript example. For Siebel VB examples, read "Applet_PreInvokeMethod" on page 142, "ActivateField" on page 198, and "ExecuteQuery" on page 206. For another eScript example, read "GotoView" on page 163.

```
var oEmpBusComp TheApplication().ActiveBusObject();
var oEmpBusObj = TheApplication().ActiveBusComp();
var sLoginName;

oEmpBusComp.ActivateField("Login Name");
oEmpBusComp.ClearToQuery();
oEmpBusComp.SetSearchSpec("Login Name", sLoginName);
oEmpBusComp.ExecuteQuery();

oEmpBusComp = null;
oEmpBusObj = null;
```

See Also "RefineQuery" on page 241

DeactivateFields

DeactivateFields deactivates the fields that are currently active from a business component SQL query statement, except those that are not ForceActive, required for a link, or required by the BusComp class.

Syntax BusComp.DeactivateFields

Argument	Description
Not applicable	

Returns Not applicable

Usage You must activate fields using ActivateField prior to executing a query for the business component.

By default, fields are inactive except when:

- They are displayed on the applet and the business component is the instance on which the applet is based.
- They are System fields (which include Id, Created, Created By, Updated, and Updated By).
- Their ForceActive property is set to TRUE.

- The method ActivateField has been invoked with the *FieldName*.
- They have the Link Specification property set to TRUE.

After fields have been deactivated, the business component must be reexecuted or the application crashes.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

See Also "ActivateField" on page 198

DeleteRecord

DeleteRecord removes the current record from the business component.

Syntax BusComp.DeleteRecord

Argument	Description
Not applicable	

Returns Not applicable

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example This Siebel VB example illustrates how to delete accounts with a status of Inactive.

```
Sub DeleteInactiveAccounts()
   Dim objBO as BusObject
   Dim objBC as BusComp

   Set objBO = theApplication.GetBusObject("Account")
   Set objBC = objBO.GetBusComp("Account")
   With objBC
      .ClearToQuery
      .SetSearchSpec "Status", "Inactive"
      .ExecuteQuery ForwardBackward
      Do While .FirstRecord
```

```
            .DeleteRecord
         Loop
      End With
      Set objBC = Nothing
      Set objBO = Nothing
End Sub
```

NOTE: The cursor is moved to the next record after DeleteRecord is executed. Therefore, it is not necessary to execute NextRecord after DeleteRecord. Do not use NextRecord after DeleteRecord in a loop because this causes the deletion of the last record in the loop to be skipped.

ExecuteQuery

ExecuteQuery returns a set of BusComp records using the criteria established with methods such as SetSearchSpec.

Syntax BusComp.ExecuteQuery (*cursorMode*)

Argument	Description
cursorMode	An integer. An optional parameter that must be one of the following constants (provided in Siebel VB as well as COM Servers): ■ **ForwardBackward.** Selected records can be processed from first to last or from last to first. This is the default if no value is specified. ■ **ForwardOnly.** Selected records can be processed only from the first record to the last record. Focus cannot return to a record.

Returns Not applicable

Usage Use a cursorMode of ForwardOnly wherever possible to achieve maximum performance. If you use ForwardOnly, make sure that your application code does not attempt to navigate backward using PreviousRecord or FirstRecord without a requery. Do not use ForwardOnly when operating on UI business components unless the application code requeries using a cursorMode of ForwardBackward.

When using the ForwardBackward cursor mode, and the query matches over 10,000 records, the object manager returns this error message: “There were more rows than could be returned. Please refine your query to bring back fewer rows.” To remove the 10,000 record restriction, set the following parameter in the data source section of the relevant CFG file for the data source:
MaxFetchArraySize = -1

For the connected Web Client, navigate to Server Administration > Enterprise Configuration > Enterprise Profile Configuration > Component Profiles > ServerDataSrc. Set the DSMaxFetchArraySize parameter to -1.

To reduce the number of queries needed, you can use the parent-child relationships for business components that are set up in business objects. For example, an Opportunity business object sets up a parent-child relationship between the Opportunity business component and the Contact business component. If you query on the Opportunity business component you can read values from the corresponding records in the Contact business component without any additional queries. Before querying a child business component, you must query its parent, otherwise the query returns no records.

Used With

Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example

This Siebel VB example sets up and executes a query to find the primary on the account team. Only the primary can change the primary address. For other examples, read “Applet_PreInvokeMethod” on page 142, “GotoView” on page 163, and “ClearToQuery” on page 203.

```
(general) (declarations)
Option Explicit

Function BusComp_PreSetFieldValue (FieldName As String,
               FieldValue As String) As Integer
   Dim iReturn As Integer, i As Integer
   Dim iFoundP As Integer ' 1 = found (TRUE), 0 = not found (FALSE)
   Dim oMVGBC as BusComp
   iReturn = ContinueOperation
   iFoundP = FALSE

   Select Case FieldName
      Case "SSA Primary Field"
      set oMVGBC = me.ParentBusComp.GetMVGBusComp("Sales Rep")
```

```
      With oMVGBC ' this is the position BC
         .ClearToQuery
         .ActivateField "Active Login Name"
         .ExecuteQuery ForwardBackward
         i = .FirstRecord
         Do While i <> 0
            if .GetFieldValue("SSA Primary Field") = "Y" then
            iFoundP = TRUE 'mark that found a primary
            if.GetFieldValue("Active Login Name") <> _
                TheApplication.LoginName then
                RaiseErrorText"You cannot change the Primary address
                because you are not the Primary on the Account Team")
                  iReturn = CancelOperation
               end if
               Exit Do
            else
               i = .NextRecord
            end if
         Loop

         if iFoundP = FALSE then
            .FirstRecord
            RaiseErrorText("No Primary Found - Contact an
Administrator")
         end if
      End With
   End Select

   set oMVGBC = Nothing
   BusComp_PreSetFieldValue = iReturn
End Function
```

This CORBA example looks up a customer name.

```
int customer::lookUp(char* fName, char* lName, SiebelApplication*
pSeblApp)
{

   SiebelBusObject* pBusObj;
   SiebelBusComp_ptr pBusComp;

   pBusObj = pSeblApp->GetBusObject("Contact");
   pBusComp = pBusObj->GetBusComp("Contact");

   //Activate fields necessary for customer look-up
   pBusComp->ActivateField("First Name");
   pBusComp->ActivateField("Last Name");
```

```
   //Prepare BusComp for query, clear existing SearchSpecs
   pBusComp->ClearToQuery();

   //Apply current SearchSpec
   pBusComp->SetSearchSpec("Last Name", lName);
   pBusComp->SetSearchSpec("First Name", fName);

   //Execute SearchSpec
   pBusComp->ExecuteQuery(0);

   //If the contact is found, return a 1. Otherwise return a 0.

   if (pBusComp->FirstRecord())
   {
      id = pBusComp->GetFieldValue("Id");
      return 1;
   }
   else
   {
      return 0;
   }

}
```

See Also "ClearToQuery" on page 203
"SetSearchSpec" on page 251

ExecuteQuery2

ExecuteQuery2 returns a set of BusComp records using the criteria established with methods such as SetSearchSpec.

Syntax BusComp.ExecuteQuery2 (*cursorMode*, *ignoreMaxCursorSize*)

Argument	Description
cursorMode	An integer. An optional parameter that can be one of the following two constants (provided in Siebel VB as well as COM Servers): ■ **ForwardBackward.** Selected records may be processed from first to last or from last to first. This is the default if no value is specified. ■ **ForwardOnly.** Selected records can be processed only from the first record to the last record. Focus cannot return to a record.
ignoreMaxCursorSize	■ **TRUE.** Retrieves every row from a business component. This option may result in lower performance. ■ **FALSE.** Retrieves the number of rows specified by the MaxCursorSize parameter in the CFG file.

Returns Not applicable

Usage ExecuteQuery2 is specific to Microsoft SQL Server.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

FirstRecord

FirstRecord moves the record pointer to the first record in a business component, making that record current and invoking any associated script events.

Syntax BusComp.FirstRecord

Argument	Description
Not applicable	

Returns An integer in Siebel VB: 1 or non-zero if there was a first record (the query returned results) and 0 if there are no records; a Boolean in Siebel eScript, COM, ActiveX, and CORBA.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example Here is a Siebel VB example. For the equivalent eScript example, read "NextRecord" on page 236.

```
Function BusComp_PreQuery () As Integer

   Dim I As Integer
   Dim isRecord As Integer
   I = 0
   With me
      .ClearToQuery
      .SetSearchSpec "Name", "*"
      .ExecuteQuery ForwardBackward
      isRecord = .FirstRecord
      Do While isRecord
         I = I + 1
         isRecord = .NextRecord
      Loop
   End With
   BusComp_PreQuery = ContinueOperation
End Function
```

See Also "NextRecord" on page 236

FirstSelected

FirstSelected moves the focus to the first record of the multiple selection in the business component, invoking any associated Basic events.

Syntax BusComp.FirstSelected

Argument	Description
Not applicable	

Returns An integer in Siebel VB: 1 or nonzero if there was a first record (the query returned results) and 0 if there are no records; a Boolean in Siebel eScript, COM, ActiveX, and CORBA.

Used With Siebel VB, Siebel eScript

Example This Siebel VB example calls the email client upon the click of a button. It uses a function, `Email`, that is not shown.

```
Sub Button1_Click
   Dim oAccounts as BusComp
   Dim oCurrAcct As BusComp, sName as String
   Dim sSendList As String, sDescription as String
   'Get the current business object
   Set oCurrAcct = TheApplication.ActiveBusComp
   With oCurrAcct
      If .FirstSelected <> 0 then
         sName = .GetFieldValue("Email Address")
         Set oAccounts = oCurrAcct
         If .FirstSelected <> 0 then
            sSendList = .GetFieldValue("Email Address")
            do while .NextSelected <> 0
               sDescription = .GetFieldValue("Email Address")
               sSendList = sSendList + ";" + sDescription
               TheApplication().RaiseErrorText(SDescription);
            loop
         end if
      end if
   End With
CallEmail(sSendList)
End Sub
```

GetAssocBusComp

GetAssocBusComp returns the association business component. The association business component can be used to operate on the association using the normal business component mechanisms.

Syntax BusComp.GetAssocBusComp

Argument	Description
Not applicable	

Returns The association business component for the current business component

Usage This method and the Associate method make sense only for many-to-many relationships, which are based on intersection tables, for example Account and Industry. In the context of a many-to-many relationship, you can use Siebel VB to either a*dd* a new record (that is, associate a new child record), or *insert* a record (that is, create a new record) in the child business component. To *add* a record, use GetAssocBusComp and the Associate method. To *insert* a record, use GetMVGBusComp and the NewRecord method. The GetAssocBusComp should be set to Nothing after use.

GetAssocBusComp can also be applied to the Child Business Component of a Master Detail View (rather than upon the MVG BusComp) when a N:M Link is used and the Child Applet has an Association Applet defined.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example The following Siebel VB sample code using GetAssocBusComp adds a new industry to an account record:

```
Dim oAssocBC As BusComp

Set oAssocBC = oMainBc.GetMVGBusComp("Industry").GetAssocBusComp
With oAssocBC
   .ActivateField "SIC Code"
   .SetSearchExpr "[SIC Code] = ""5734"" "
   .ExecuteQuery ForwardOnly

   If .FirstRecord Then .Associate NewBefore

End With

Set oAssocBC = Nothing
```

Here is the equivalent Siebel eScript code.

```
var oAssocBC = oMainBc.GetMVGBusComp("Industry").GetAssocBusComp();
with (oAssocBC)
{
   ActivateField("SIC Code");
   SetSearchExpr("[SIC Code] = ""5734"" ");
   ExecuteQuery(ForwardOnly)

   If (FirstRecord)
      Associate(NewBefore);
}

oAssocBC = null;
```

See Also

"GetMVGBusComp" on page 220
"GetPicklistBusComp" on page 222

GetFieldValue

GetFieldValue returns the value for the field specified in its argument for the current record of the business component. Use this method to access a field value.

Syntax

BusComp.GetFieldValue(*FieldName*)

Argument	Description
FieldName	String variable or literal containing the name of the field

Returns

A string containing the field value of the field identified in *FieldName*, or an error message if the field is inactive or empty.

Usage

Only fields that were active at the time of the BusComp query contain values. For more information, read "ActivateField" on page 198. If this method is used on fields that are not active, or on fields that are empty, an error message is returned.

CAUTION: If a value from a business component that is a child of the current business component is desired, the Link Specification property for that field must be set to TRUE in Siebel Tools. Otherwise, the parent business component cannot access the value in the child business component. For more information, read the Object Types Reference topics within *Siebel Tools Online Help*.

The *FieldName* must be enclosed in double quotes and must be spelled exactly as the field name appears in Siebel Tools, with the correct case; for example,

```
GetFieldValue("ActivityCreatedByName")
```

The name `"Person who created the activity"`, as shown in the status bar, does not work; nor does the column head `"Created By"`.

NOTE: In Browser Script, GetFieldValue can be used only for the fields exposed in the applet.

Used With

Browser Script, Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example

This Siebel VB sample implementation of the PreSetFieldValue event illustrates the use of GetFieldValue.

```
Function BusComp_PreSetFieldValue (FieldName As String,
FieldValue As String) As Integer

   Dim bcOppty As BusComp
   Dim boBusObj As BusObject
   Dim srowid As String

   srowid = GetFieldValue("Id")
   Set boBusObj = theApplication.GetBusObject("Opportunity")
   Set bcOppty = boBusObj.GetBusComp("Opportunity")
   With bcOppty
      .SetViewMode SalesRepView
      .ActivateField "Sales Stage"
```

```
         .SetSearchSpec "Id", srowid
         .ExecuteQuery ForwardOnly
      End With

      Set bcOppty = Nothing
      Set boBusObj = Nothing

   End Function
```

Here is the equivalent example in Siebel eScript.

```
   function BusComp_PreSetFieldValue (FieldName, FieldValue)

      var boBusObj = theApplication().GetBusObject("Opportunity");
      var bcOppty = boBusObj.GetBusComp("Opportunity");
      var srowid = GetFieldValue("Id");

      with (bcOppty)
      {
         SetViewMode(SalesRepView);
         ActivateField("Sales Stage");
         SetSearchSpec("Id", srowid);
         ExecuteQuery(ForwardOnly);
      }

      bcOppty = null;
      boBusObj = null;
   }
```

See Also "ActivateField" on page 198
"GetFormattedFieldValue"

GetFormattedFieldValue

GetFormattedFieldValue returns the field value in the current local format; it returns values in the same format as the Siebel UI.

Syntax BusComp.GetFormattedFieldValue(*FieldName*)

Argument	Description
FieldName	String variable or literal containing the name of the field to obtain the value from

Returns A string containing the value of the requested field, in the same format as displayed in the user interface, or an empty string ("") if the field is inactive or empty.

Usage GetFormattedFieldValue is useful for code that is used in multiple countries with different formats for currency, date, and number. This method can be used only on fields that have been activated using ActivateField.

Some special behavior is associated with particular data types.

DTYPE_PHONE. When used on fields of DTYPE_PHONE, these methods return formatted phone numbers.

Example 1:

```
phone = bc.GetFieldValue("Main Phone Number")
theApplication.Trace "The number is " & phone
```

Result:

```
The number is 8869629123
```

Example 2:

```
phone = bc.GetFormattedFieldValue("Main Phone Number")
theApplication.Trace "The number is " & phone
```

Result:

```
The number is (886) 962-9123
```

DTYPE_DATE. When used on fields of DTYPE_DATE, these methods are the same as GetFieldValue and SetFieldValue, except that the result is in the format of the Regional Setting.

Table 24 shows the standard formats used by GetFieldValue and SetFieldValue to return data.

Table 24. Date and Time Formats

Type of Data	Format
Dates	mm/dd/yyyy

Table 24. Date and Time Formats

Type of Data	Format
Times	hh:nn:ss
Date-times	mm/dd/yyyy hh:nn:ss

If you attempt to use SetFieldValue and your Regional Setting format is different, you receive an error like this:

```
Error: The value '31-Dec-99' can not be converted to a date time
value.
```

This error can be avoided by using the GetFormattedFieldValue and SetFormattedFieldValue methods.

Used With

Browser Script, Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example

The following Siebel VB example demonstrates how to use the GetFormattedFieldValue function and how to calculate the number of days between two dates.

```
Sub Button_Click
   Dim DateDiff as Integer
   Dim oBC as BusComp
   Set oBC= me.BusComp
   x = oBC.GetFormattedFieldValue("Start Date")
   y = oBC.GetFormattedFieldValue("Done")
   dx = DateValue(x)
   dy = DateValue(y)
   DateDiff = dy - dx
End Sub
```

See Also

"ActivateField" on page 198
"GetFieldValue" on page 214
"SetFieldValue" on page 243
"SetFormattedFieldValue" on page 245

GetLastErrCode

The GetLastErrCode method returns the most recent error code.

Syntax BusComp.GetLastErrCode

Argument	Description
Not applicable	

Returns The last error code as a short integer. 0 indicates no error.

Usage After execution of a method, the GetLastErrCode can be invoked to check if any error was returned from the previous operation. The GetLastErrText method can be invoked to retrieve the text of the error message.

Used With Mobile/Dedicated Web Client Automation Server, COM Data Control

GetLastErrText

The GetLastErrText method returns the last error text message.

Syntax BusComp.GetLastErrText

Argument	Description
Not applicable	

Returns The most recent error text message as a String

Usage After execution of a method, the GetLastErrCode can be invoked to check if any error was returned from the previous operation. The GetLastErrText method can be invoked to retrieve the text of the error message.

Used With Mobile/Dedicated Web Client Automation Server, COM Data Control

See Also "GetLastErrCode"

GetMultipleFieldValues

GetMultipleFieldValues returns a value for the fields specified in the property set.

Syntax Boolean GetMultipleFieldValues(SiebelPropertySet *fieldNames*, SiebelPropertySet *fieldValues*);

Argument	Description
fieldNames	A property set containing a collection of properties representing the fields
fieldValues	A property set containing a collection of properties representing the values for the fields specified in the *fieldNames* parameter

Returns TRUE if success; FALSE if failure

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

See Also "SetMultipleFieldValues" on page 247

GetMVGBusComp

GetMVGBusComp returns the MVG business component associated with the business component field specified by *FieldName*. This business component can be used to operate on the multi-value group using the normal business component mechanisms.

Syntax BusComp.GetMVGBusComp(*FieldName*)

Argument	Description
FieldName	Name of the field with a multi-value group attached, used to obtain the multi-value group business component

Returns The multi-value group business component of the current business component and identified field

Usage A multi-value group is a set of detail records attached to the current record in the business component that holds the corresponding multivalue field.

The GetMVGBusComp should be set to Nothing after use.

NOTE: In the context of a many-to-many relationship, you can use Siebel VB to either add a new record, that is, associate a new child record, or insert a record, that is, create a brand-new record in the child business component. To *add* a record, use GetAssocBusComp and the Associate method. To *insert* a record, use GetMVGBusComp and the NewRecord method.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example The following sample code using GetMVGBusComp adds a new address to the "Hong Kong Flower Shop" account record. For other examples, read "ExecuteQuery" on page 206 and "FirstSelected" on page 211.

```
Dim AccntBO as BusObject
Dim AccntBC as BusComp
Dim AddrBC as BusComp
Set AccntBO = theApplication.GetBusObject("Account")
Set AccntBC = AccntBO.GetBusComp("Account")

With AccntBC
   .SetViewMode SalesRepView
   .ActivateField "Name"
   .ClearToQuery
   .SetSearchSpec "Name", "Hong Kong Flower Shop"
   .ExecuteQuery
   Set AddrBC = .GetMVGBusComp("Street Address")
End With

With AddrBC
   .NewRecord NewAfter
   .SetFieldValue "City", "Denver"
   .WriteRecord
End With
```

```
Set AccntBO = Nothing
Set AccntBC = Nothing
Set AddrBC = Nothing
```

See Also "FirstSelected" on page 211
"GetPicklistBusComp"

GetNamedSearch

GetNamedSearch returns the named search specification specified by *searchName*.

Syntax BusComp.GetNamedSearch(*searchName*)

Argument	Description
searchName	Name of the search specification that references the search string.

Returns A string containing the value specified in the search specification identified in *searchName*

Usage The search specification uses the same syntax as that used in predefined queries.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

See Also "GetSearchExpr" on page 224
"GetSearchSpec" on page 225
"SetNamedSearch" on page 248

GetPicklistBusComp

GetPicklistBusComp returns the pick business component associated with the specified field in the current business component.

Syntax BusComp.GetPicklistBusComp(*FieldName*)

Argument	Description
FieldName	Name of the field with a picklist specified; used to obtain the pick business component

Returns The pick business component of the current business component and identified field

Usage The returned pick business component can be used to operate on the picklist. The GetPickListBusComp should be destroyed after use by using the Nothing function.

NOTE: When a record is picked on a constrained picklist using the GetPickListBusComp and Pick methods, the constraint is active. Therefore, the retrieved picklist business component contains only those records that fulfill the constraint.

To pick a value from a picklist in Siebel VB

1. Use GetPicklistBusComp to create an instance of the pick list business component.
2. Navigate in the picklist business component to the record you want to pick.
3. Use Pick to pick the value.
4. Use `Set objBCPickList = Nothing` to explicitly destroy the picklist business component instance.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example Here is a Siebel VB example.

```
If Me.GetFieldValue("City") = "San Mateo" Then
   Set oBCPick = Me.GetPicklistBusComp("State")
   With oBCPick
      .ClearToQuery
```

```
         .SetSearchSpec "Value", "CA"
         .ExecuteQuery ForwardOnly
         If .FirstRecord Then .Pick
      End With
      Set oBCPick = Nothing
   End If
```

Here is the equivalent example in Siebel eScript.

```
if (this.GetFieldValue("City") = "San Mateo")
{
   var oBCPick = this.GetPicklistBusComp("State");
   with (oBCPick)
   {
      ClearToQuery();
      SetSearchSpec("Value", "CA");
      ExecuteQuery(ForwardOnly);
      if(FirstRecord())
         Pick();
   }
   oBCPick = null;
}
```

See Also

"FirstSelected" on page 211
"GetMVGBusComp" on page 220

GetSearchExpr

GetSearchExpr returns the current search expression for the business component.

Syntax

BusComp.GetSearchExpr

Argument	Description
Not applicable	

Returns

A string containing the current search expression. An example of a returned search expression string is "Revenue > 10000 AND Probability > .5".

Used With

Browser Script, Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

See Also

"GetNamedSearch" on page 222
"GetSearchSpec"
"SetSearchExpr" on page 249

GetSearchSpec

GetSearchSpec returns the search specification for the field specified by the *FieldName* argument.

Syntax

BusComp.GetSearchSpec(*FieldName*)

Argument	Description
FieldName	Contains the name of the field from which to obtain the associated search specification.

Returns

A string containing the search specification for the field identified in *FieldName*. An example of a returned search specification string is "> 10000".

Used With

Browser Script, Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

See Also

"GetNamedSearch" on page 222
"GetSearchExpr" on page 224
"SetSearchSpec" on page 251

GetUserProperty

GetUserProperty returns the value of a named user property.

Syntax

BusComp.GetUserProperty(*propertyName*)

Argument	Description
propertyName	Contains the name of the user property to obtain.

Returns

The user property

Usage The value of a user property is set using SetUserProperty. The user properties act like instance variables of a business component. The advantage of user properties is that they can be accessed from anywhere in the code (even from other applications through COM) using GetUserProperty. An instance variable, on the other hand, can be accessed only from within Siebel VB from the same object on which the variable is declared.

The value of the property is reset every time you instantiate a new business component.

NOTE: GetUserProperty does not interact directly with user properties defined in Siebel Tools.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

See Also "SetUserProperty" on page 257

GetViewMode

GetViewMode returns the current visibility mode for the business component. This effects which records are returned by queries according to the visibility rules. For more information, read "SetViewMode" on page 258.

Syntax BusComp.GetViewMode

Argument	Description
Not applicable	

Returns An integer constant that identifies a visibility mode

mode Where *mode* is a SiebelView Mode constant or its corresponding integer value. The constants shown are defined in three environments. For details on each SiebelViewMode constant, read "SetViewMode" on page 258.

- SalesRepView (0)
- ManagerView (1)
- PersonalView (2)
- AllView (3)
- OrganizationView (5)
- GroupView (7)
- CatalogView (8)
- SubOrganizationView (9)

Usage GetViewMode() returns NoneSetView mode until a business component is executed or has its view mode set through SetViewMode(). The NoneSetViewMode value indicates that the business component has not yet had any visibility rules applied to it. A business component that has just been created through a call to GetBusComp() is in this state, so if a specific view mode is desired, it must be explicitly set through SetViewMode(). Otherwise, the first time the business component is executed, its view mode is set according to some internal rules.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

See Also "SetViewMode" on page 258

InvokeMethod

InvokeMethod calls the specialized method or user-created method named in the argument.

VB Syntax

BusComp.InvokeMethod *methodName, methodArgs*

Argument	Description
methodName	The name of the method. For more information on the available methods, read "InvokeMethod Methods" on page 230.
methodArgs	A single string or a string array (COM only) containing arguments to *methodName*

eScript Syntax

BusComp.InvokeMethod(*methodName, methArg1, methArg2, …, methArg*n);

Argument	Description
methodName	The name of the method
*methArg1, methArg2, …, methArg*n	One or more strings containing arguments *to methodName*

Returns

In Server Script, returns a string containing the result of the method.

In Browser Script, returns a property set.

Usage

Use InvokeMethod to call methods on a business component object that are not exposed directly through the object interface.

Specialized methods are typically methods implemented in applet or business component classes other than CSSFrame and CSSBusComp, respectively, that is, specialized classes.

NOTE: The InvokeMethod method should be used only with documented specialized methods. Siebel Systems does not support calling specialized methods with InvokeMethod, unless they are listed in this book.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, Java Data Bean, CORBA Object Manager

Example Here is a Siebel VB example.

```
(general) (declarations)
Option Explicit

Sub Button1_Click
   InvokeMethod "Select All"
End Sub

Function BusComp_PreInvokeMethod (MethodName As String) As
Integer
   BusComp_PreInvokeMethod = ContinueOperation
   On Error GoTo Leave
   If MethodName = "Select All" Then
      Dim oCurBC as BusComp
      Set oCurBC = Me.BusComp
      If oCurBC is not nothing Then
         oCurBC.ClearToQuery
         oCurBC.ExecuteQuery
         BusComp_PreInvokeMethod = CancelOperation
      End If
   End If
Leave:
End Function
```

Here is the equivalent example in Siebel eScript.

```
function BusComp_PreInvokeMethod (MethodName)
{
   var iReturn = ContinueOperation;
   If (Clib.errno() != 0)
      break Leave;
   if (MethodName = "Select All")
   {
      var oCurBC = Me.BusComp
      if (oCurBC != null)
         oCurBC.ClearToQuery();
         oCurBC.ExecuteQuery();
         iReturn = CancelOperation;
      }
   }
```

```
   return (iReturn);
Leave:
}
```

InvokeMethod Methods

Siebel applications provide multiple methods for manipulating files stored in the Siebel File System. These methods may be invoked using server script (Siebel VB, eScript) or using one of our programmatic interfaces (Mobile/Dedicated Web Client Automation Server – connected mode only, COM Data Control, Java Data Bean). The methods available for manipulating the file system always store or retrieve the file to and from the local file system. For example, if you construct a Java client using the Java Data Bean to manipulate the file system, all files must be accessible from the Siebel Server. You can use UNC naming conventions (for example: \\server\dir\file.txt) or standard DOS directories (for example: D:\dir\file.txt) for file access, but the UNC path or mounted file system must be accessible to the Siebel Server. These methods do not serialize the files from a remote client and place them in the Siebel file system.

Methods that manipulate files are available for business components whose Class is 'CSSBCFile'. The methods can be accessed using Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, and the Java Data Bean.

The following methods are available for use with InvokeMethod.

- "CreateFile"
- "GetFile" on page 231
- "PutFile" on page 232
- "SetAdminMode" on page 232

CreateFile

To create a file in the Siebel file system from an external source, use the business component CreateFile method. Before calling CreateFile, make sure that a new business component record has been created using the NewRecord method for the business component.

Syntax retVal = BusComp.InvokeMethod("CreateFile", filePath, KeyField, keepLink)

Argument	Description
SrcFilePath	The fully qualified path of the file on the Siebel Server or Mobile Web Client.
KeyFieldName	The name of the field in the business component that contains the File Name. For example: AccntFileName field in the Account Attachment business component.
KeepLink	Applies to URLs. Either Y or N depending on whether a link to the file is stored as an attachment instead of the actual file.

Returns A string containing the values of "Success" or "Error" depending on whether or not the operation succeeded.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, Java Data Bean

GetFile

Obtains a file from the Siebel file system and places that file on the local file system of the Siebel Server or Mobile Client. Note that you must be properly positioned on the desired file attachment record to get the file and have it placed on the local file system's temporary directory.

Syntax retVal = BusComp.InvokeMethod("GetFile", KeyField)

Argument	Description
KeyFieldName	The name of the field in the business component that contains the File Name. For example: AccntFileName field in the Account Attachment business component.

Returns A string containing "Success, <outFilePath>" if the operation succeeded. OutFilePath is the fully qualified path of the file on the Client/Server machine in the user's temp directory. The return value is "Error" if the operation failed.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, Java Data Bean

PutFile

Updates a file in the Siebel file system with a newer file. Note that you must be properly positioned on the desired file attachment record to update the file in the file system.

Syntax retVal = BusComp.InvokeMethod("GetFile", SrcFilePath, KeyField)

Argument	Description
SrcFilePath	This is the fully qualified path of the file on the Siebel Server or Mobile Web Client.
KeyFieldName	This is the name of the field in the business component that contains the File Name. For example: AccntFileName field in the Account Attachment business component.

Returns A string containing the values of "Success" or "Error" depending on whether or not the operation succeeded.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, Java Data Bean

SetAdminMode

This method is particularly useful if you need to replicate the behavior enforced by the 'Admin' property of the View object by disabling all visibility rules for the business component.

Syntax retVal = BusComp.InvokeMethod("SetAdminMode", flag)

Argument	Description
flag	"TRUE" or "FALSE". Flag to specify whether the business component should be executed in Admin mode.

Returns Not Applicable

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, Java Data Bean

LastRecord

LastRecord moves to the last record in the business component.

Syntax *oBusComp*.LastRecord

Argument	Description
Not applicable	

Returns An integer in Siebel VB; a Boolean in Siebel eScript, COM, ActiveX, and CORBA: 1 or non-zero if there was a last record (the query returned results) and 0 if there are no records. LastRecord requires an explicit SortSpec.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example Here is a Mobile/Dedicated Web Client Automation Server example.

```
Private Sub LastRecord_Click()

   Dim errCode As Integer
   Dim oBusComp as SiebelBusComp
   FieldValue.Text = ""
   HourGlassStart
   oBusComp.LastRecord errCode

   If errCode = 0 Then
      FieldValue.Text = oBusComp.GetFieldValue(FieldName.Text, _
         errCode)
   End If

   HourGlassStop

   Status.Text = SiebelApplication.GetLastErrText
End Sub
```

See Also "FirstRecord" on page 210
"NextRecord" on page 236

Name

The Name property contains the name of the business component.

Syntax *oBusComp*.Name

Argument	Description
Not applicable	

Returns A string containing the business component name

Used With Browser Script, Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager*

Example Here is a Browser Script example:

```
function BusComp_PreSetFieldValue (fieldName, value)
{
   TheApplication().SWEAlert(this.Name());
}
```

NewRecord

NewRecord adds a new record (row) to the business component.

Syntax *oBusComp*.NewRecord(*whereIndicator*)

Argument	Description
whereIndicator	Predefined constant or corresponding integer indicating where the new row is added. This value should be one of the following: ■ 0 (or NewBefore) ■ 1 (or NewAfter) ■ 2 (or NewBeforeCopy) ■ 3 (or NewAfterCopy) With Java Data Bean the values are: ■ FALSE (equivalent to NewBefore) ■ TRUE (equivalent to NewAfter)

Returns Not applicable

Usage This new row becomes the current row, either before or after the previously current record, depending on the value you selected for WhereIndicator.

You can use NewRecord to copy a record. To place the copy before the original record use the following command.

```
Object.NewRecord NewBeforeCopy
```

To place the copy after the original record, use the following command.

```
Object.NewRecord NewAfterCopy
```

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example Here is a Siebel VB example:

```
Dim oBusObj as BusObject
Dim oBC as BusComp
```

```
Set oBusObj = theApplication.ActiveBusObject
Set oBC = oBusObj.GetBusComp("Action")
oBC.NewRecord NewAfter
oBC.SetFieldValue "Type", "To Do"
oBC.SetFieldValue "Description", "Find Decision Makers")
oBC.WriteRecord

set oBC = Nothing
set oBusObj = Nothing
```

NextRecord

NextRecord moves the record pointer to the next record in the business component, making that the current record and invoking any associated script events.

Syntax *oBusComp*.NextRecord

Argument	Description
Not applicable	

Returns An integer in Siebel VB; a Boolean in Siebel eScript and COM: 1 if the current record was moved to the next record, 0 if the current record was already the last record.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example Here is a Siebel eScript example. For the equivalent Siebel VB example, read "FirstRecord" on page 210.

```
var i = 0;
var isRecord;

with (this)
{
   ClearToQuery();
   SetSearchSpec("Name", "*");
   ExecuteQuery(ForwardBackward);
   isRecord = FirstRecord();
}
while (isRecord)
{
```

```
        i++;
        isRecord = BusComp.NextRecord();
    }
```

See Also "FirstRecord" on page 210

NextSelected

NextSelected moves the focus to the next record of the current multiple selection.

Syntax *oBusComp*.NextSelected

Argument	Description
Not applicable	

Returns An integer: 1 if there is another record in the multiple selection, 0 otherwise.

Used With Siebel eScript, Siebel VB

Example For examples, read "InvokeMethod" on page 164 and "FirstSelected" on page 211.

ParentBusComp

ParentBusComp returns the parent (master) business component when given the child (detail) business component of a Link.

Syntax *oBusComp*.ParentBusComp

Argument	Description
Not applicable	

Returns The parent business component of the Link

Usage ParentBusComp allows you to write code in the child business component that accesses field values and performs actions on the parent business component using the normal business component mechanisms.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example Here is a Siebel VB example. For another example, read "ExecuteQuery" on page 206.

```
Dim strParentName as String
...
strParentName = Me.ParentBusComp.GetFieldValue("Name")
```

Pick

The Pick method places the currently selected record in a picklist business component into the appropriate fields of the parent business component.

Syntax *oBusComp*.Pick

Argument	Description
Not applicable	

Returns Not applicable

Usage Pick must be invoked on the picklist's business component. When a record is picked on a constrained picklist using the GetPickListBusComp and Pick methods, the constraint is active. Therefore, only records that fulfill the constraint can be retrieved.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object ManagerExample

This Siebel VB example sorts the values in the Sales Stage field.

```
Sub BusComp_NewRecord
   Dim oBC as BusComp
   set oBC = me.GetPickListBusComp("Sales Stage")

   With oBC
      .ClearToQuery
      .ActivateField "Sales Stage Order"
      .SetSortSpec "Sales Stage Order"
      .ExecuteQuery ForwardOnly
      if .FirstRecord then .Pick
   End With

   set oBC = Nothing
End Sub
```

Here is the same example in Siebel eScript.

```
function BusComp_NewRecord ()
{
   var oBC = this.GetPickListBusComp("Sales Stage");

   with (oBC)
      ClearToQuery();
      ActivateField("Sales Stage Order");
      SetSortSpec("Sales Stage Order");
      ExecuteQuery(ForwardOnly);
      if (FirstRecord())
         Pick();
   }

   oBC = null;
}
```

See Also "GetPicklistBusComp" on page 222

PreviousRecord

PreviousRecord moves to the previous record in the business component, invoking any associated Basic events.

Syntax *oBusComp*.PreviousRecord

Argument	Description
Not applicable	

Returns An integer in Siebel VB; Siebel eScript, a Boolean in COM, ActiveX, and CORBA: 1 or non-zero if the current record was moved to the previous record, 0 if the current record was already the first record.

Usage PreviousRecord may be used only on a business component that has been queried using the FowardBackward CursorMode.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example Here is a Mobile/Dedicated Web Client Automation Server example.

```
(general) (declarations)
Option Explicit

Private Sub PreviousRecord_Click()
   Dim errCode As Integer
   Dim oBusComp as BusComp
   FieldValue.Text = ""
   HourClassStart
   SBusComp.PreviousRecord errCode
   If errCode = 0 Then
      FieldValue.Text = SBusComp.GetFieldValue(FieldName.Text, _
         errCode)
   End If

   HourClassStop
   Status.Text = SiebelApplication.GetLastErrText

End Sub
```

See Also "ExecuteQuery" on page 206

RefineQuery

This method refines a query after the query has been executed.

Syntax *oBusComp*.RefineQuery

Argument	Description
Not applicable	

Returns Not applicable

Usage Unlike ClearToQuery, RefineQuery retains the existing query specification and allows you to add search conditions based only on those fields that have not been set by previous search expressions. RefineQuery may be most useful when used in conjunction with GetNamedSearch.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example The following Siebel VB code fragment shows how RefineQuery might be used:

```
me.ActivateField "Status"
me.SetSearchSpec "Status", "Open"
me.ClearToQuery
me.ExecuteQuery
me.RefineQuery
me.SetSearchSpec "Substatus", "Assigned"
me.ExecuteQuery
```

See Also "ClearToQuery" on page 203
"GetNamedSearch" on page 222

Release

The Release() method enables the release of the business component and its resources on the Siebel Server.

Syntax release()

Argument	Description
not applicable	

Returns Not applicable

Used With Java Data Bean

Example Here is an example for Java Data Bean:

```
import com.siebel.data.*;
{
   …
// create Siebel Data Bean
   // login into Siebel Data Bean
…
   // Create Siebel Bus Object.
// get the Bus Object from SiebelDataBean
   …
   // Create Siebel Bus Comp siebBusComp
// Get the business component using SiebelBusObject
   …
// Use the bus. Component
…
// Be sure to release the business component and its resources on the server
side   siebBusComp.release();
// release the resources occupied by Siebel Bus Object and Siebel Data Bean after
their use.
}
```

See Also "Logoff" on page 170
"Release" on page 327

SearchExpr

SearchExpr sets and gets the search specification for the business component.

Syntax *oBusComp*.SearchExpr(*SearchString*)

Argument	Description
SearchString	String containing the search specification

Returns The current search specification

Usage If an argument is passed in, the method sets the search specification according to the argument. If there is no argument when the method is called, it returns the current search specification.

Used With CORBA Object Manager

SetFieldValue

SetFieldValue assigns the new value to the named field for the current row of the business component.

Syntax *oBusComp*.SetFieldValue *FieldName, FieldValue*

Argument	Description
FieldName	String containing the name of the field to assign the value to
FieldValue	String containing the value to assign

Returns Not applicable

Usage This method can be used only on fields that are active. For details, read "ActivateField" on page 198. For applications in Standard Interactivity mode, write the record immediately after using SetFieldValue by calling WriteRecord.

FieldName must be enclosed in double quotes, and must be spelled exactly as the field name appears in Siebel Tools (*not* in the status line of the application or the column head), with the correct case; for example,

```
SetFieldValue "Name", "Acme"
```

FieldValue must not have a length that exceeds the defined length of the field. For example, passing a 20 character string into a field that is defined as being 16 characters long results in the runtime error “Value too long for field 'xxxxx' (maximum size nnn).” A good practice is to check the length of the string against the length of the destination field before using SetFieldValue.

Do not use the SetFieldValue method on a field that has a pick list. Instead, use the following procedure:

1. Use GetPicklistBusComp(...) to get a reference to the picklist business component for the Last Name field.
2. Set the required SearchSpec on the pick list business component so that a single unique record is returned.
3. Execute the query on the pick list business component.
4. Call picklistbuscomp.Pick to emulate the user picking the record.

NOTE: SetFieldValue cannot be used with calculated fields and cannot be used recursively.

Used With

Browser Script, Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example

Here is a Siebel VB example.

```
Dim CurrOppty as BusComp
Set CurrOppty = TheApplication.ActiveBusComp
If Val(CurrOppty.GetFieldValue("Rep %")) < 75 Then
   CurrOppty.SetFieldValue "Rep %", "75"
End If
```

Here is the same example in Siebel eScript.

```
var CurrOppty = TheApplication().ActiveBusComp()
if (ToInteger(CurrOppty.GetFieldValue("Rep %")) < 75)
   CurrOppty.SetFieldValue("Rep %", "75");
```

See Also

"ActivateField" on page 198
"SetFormattedFieldValue"
"Pick" on page 238
"GetPicklistBusComp" on page 222

SetFormattedFieldValue

SetFormattedFieldValue assigns the new value to the named field for the current row of the business component. SetFormattedFieldValue accepts the field value in the current local format.

Syntax

oBusComp.SetFormattedFieldValue *FieldName, FieldValue*

Argument	Description
FieldName	String containing the name of the field to assign the value to.
FieldValue	String containing the value to assign.

Returns

Not applicable

Usage

This method is useful when you write code for a Siebel configuration that is used in multiple countries with different currency, date, and number formats. This method can be used only on fields that have been activated using ActivateField.

Used With

Browser Script, Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example

This Siebel VB example is a fragment from a program designed to track the velocity of an opportunity through its sales stages.

```
(general) (declarations)
Option Explicit

Dim OpportunityBO as BusObject, StageBC as BusComp
Dim OppStageId as String, SalesRep as String, Stage as String
Dim StagePrev As String, StageDate as String, StageDatePrev as String
Dim Dx as Double, Dy as Double, Diff as Double, DiffStr as String
Dim OppID As String, OppStageId as String, StageID As String
Dim SalesStageBO as BusObject, SalesStageBC as BusComp
```

```
Set SalesStageBO = TheApplication.GetBusObject ("Sales Cycle Def")
Set SalesStageBC = SalesStageBO.GetBusComp("Sales Cycle Def")

With SalesStageBC
   .SetViewMode AllView
   .ClearToQuery
   .SetSearchSpec "Sales Cycle Stage", StagePrev
   .ExecuteQuery ForwardOnly
   .FirstRecord
   StageId = .GetFieldValue("Id")
End With

'Instantiate stage BC
Set StageBC = OpportunityBO.GetBusComp("Opportunity Stage")

'Check that we do not already have a record for the stage

   With StageBC
      .SetViewMode AllView
      .ClearToQuery
      .SetSearchSpec "Sales Stage Id", StageId
      .ExecuteQuery ForwardOnly

'Proceed further only if we do not already have record
'opportunity sales stage

      If (.FirstRecord = 0) Then
         'Create a new stage record and write it out
            .NewRecord 1
            'Record Id for future use
            OppStageId = .GetFieldValue("Id")
            .SetFieldValue "Opportunity Id", OppId
            .SetFieldValue "Sales Stage Id", StageId
            .SetFieldValue "Sales Rep", SalesRep
            .SetFormattedFieldValue "Entered Date", StageDatePrev
            .SetFormattedFieldValue "Left Date", StageDate
            Dx = DateValue (StageDatePrev)
            Dy = DateValue (StageDate)
            Diff = Dy - Dx
            DiffStr = Str(Diff)
            .SetFieldValue "Days In Stage", DiffStr
            .WriteRecord
      End If
   End With
```

See Also

"ActivateField" on page 198
"SetFieldValue" on page 243

SetMultipleFieldValues

SetMultipleFieldValues assigns a new value to the fields specified in the property set for the current row of the business component.

Syntax *oBusComp.SetMultipleFieldValues oPropertySet*

Argument	Description
oPropertySet	Property set containing a collection of properties representing the fields to be set, and their values

Returns Not applicable

Usage This method can be used only on fields that are active. The FieldName parameter in the property must be set exactly as the field name appears in Siebel Tools, with the correct case. For example, in

```
oPropertySet.SetProperty "Name","Acme"
```

the FieldName is “Name” and the FieldValue is “Acme”.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example Here is an eScript example:

```
var bo = TheApplication().GetBusObject("Account");
var bc = bo.GetBusComp("Account");
var propSet = TheApplication().NewPropertySet();

   with (bc) {
   with (propSet) {
      SetProperty("Name", "");
      SetProperty("Location", "");
      SetProperty("Account Status");
      SetProperty("Type", "");
   }

   //Activate all fields that are to be set
   ActivateMultipleField(propSet);
   NewRecord (NewBefore);
```

```
   propSet.Reset();

with (propSet) {
      SetProperty ("Name", Acme");
      SetProperty ("Location", "Portsmouth");
      SetProperty("Account Status", "Active");
      SetProperty("Type", "Commercial");
}

SetMultipleFieldValues (propSet);
WriteRecord();

   }

bc = null;
bo = null;
propSet = null;
```

See Also "ActivateMultipleFields" on page 199
"GetMultipleFieldValues" on page 220

SetNamedSearch

SetNamedSearch sets a named search specification on the business component. A named search specification is identified by the *searchName* argument.

Syntax *oBusComp*.SetNamedSearch *searchName*, *searchSpec*

Argument	Description
searchName	String containing the name of the named search specification
searchSpec	String containing the search specification string corresponding to the name

Returns Not applicable

Usage A named search specification is a search criterion that is not cleared by the ClearToQuery; for example, a predefined query or business component search specification.

A named search specification can be modified only programmatically; it cannot be modified through the UI. This specification is applied in conjunction with the existing search specification. Once set, the named search specification is applied every time ExecuteQuery is called. ClearToQuery does not clear the named search specification. To clear it, explicitly set the searchSpec argument to "". Note that when a new instance of the BusComp is created, the named search specification is cleared.

The *searchSpec* argument assigned to SetNamedSearch is the same argument that is used after the equal sign in a predefined query. The maximum length of a predefined query is 2000 characters. For details on how to set up the search specification, read "SetSearchExpr" and "SetSearchSpec" on page 251.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

See Also "GetNamedSearch" on page 222
"SetSearchSpec" on page 251

SetSearchExpr

SetSearchExpr sets an entire search expression on the business component, rather than setting one search specification per field. Syntax is similar to that on the Predefined Queries screen.

Syntax *oBusComp*.SetSearchExpr *searchSpec*

Argument	Description
searchSpec	Search specification string field

Returns Not applicable

Usage Call this method after ClearToQuery and before ExecuteQuery.

The maximum length of a predefined query is 2000 characters. The argument assigned to SetSearchExpr is the same as that used after the equal sign in a predefined query. For example, the first line following is a search specification in a predefined query; the second is the equivalent search specification used with the various interface methods. Note that Name is a field on the business component and therefore must be enclosed in brackets, [ ].

```
'Account'.Search = "[Name] ~ LIKE ""A. C. Parker"" "
```

```
BC.SetSearchExpr "[Name] ~ LIKE ""A. C. Parker"" "
```

If field values have search keywords such as NOT, AND, and OR, use two pairs of double quotes around the field value. For example, if a field Sub-Status can have the string “Not an Issue” as a field value, then use the following syntax to avoid an SQL error:

```
substatus = GetFieldValue("Sub-Status")
searchst = "[Value] = """ & substatus & """"
BC.SetSearchExpr searchst
```

The following syntax generates an SQL error:

```
substatus = GetFieldValue("Sub-Status")
searchst = "[Value] = " & substatus
BC.SetSearchExpr searchst
```

Use both SetSearchExpr and SetSortSpec to build a query that includes both a search specification and a sort specification. You cannot set a sort specification with SetSearchExpr by itself. Do not use SetSearchExpr and SetSearchSpec together, they are mutually exclusive.

Used With

Browser Script, COM Data Control, COM Data Server, CORBA Object Manager, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script

See Also

“ClearToQuery” on page 203
“ExecuteQuery” on page 206
“SetSearchSpec” on page 251
“SetSortSpec” on page 255

Example

Here is an eScript example:

```
var Ob = TheApplication().ActiveBusObject();
var BC = Ob.GetBusComp("Opportunity");

var Account = "Turston Steel";
var Oppty = "CAD/CAM implementation";
var searchst = "[Name] = '" + Oppty + "' AND [Account] = '" +
Account + "'";

TheApplication().TraceOn("c:\temp\trace.txt", "Allocation",
"All");
TheApplication().Trace("the search expression is: " + searchst);

BC.ClearToQuery();
BC.SetSearchExpr(searchst);
BC.ExecuteQuery();
```

SetSearchSpec

SetSearchSpec sets the search specification for a particular field. This method must be called before ExecuteQuery.

Syntax *oBusComp*.SetSearchSpec *FieldName*, *searchSpec*

Argument	Description
FieldName	String containing the name of the field on which to set the search specification.
searchSpec	String containing the search specification.

Returns Not applicable

Usage If ClearToQuery is not called before SetSearchSpec, the business component performs a logical AND on the new search specification and the existing search specification. The maximum length of a predefined query is 2000 characters. Do not use SetSearchExpr and SetSearchSpec together, they are mutually exclusive.

Using logical and comparison operators. Any search specification that can be created in the user interface can be duplicated in Siebel VB or eScript. Both logical operators and comparison operators may be used, provided that they are handled correctly. Here is an example:

```
BC.SetSearchSpec "Status", "<> 'Closed' AND ([Owner] = LoginName
() OR [Refer To] = LoginName ()) OR ([Owner] IS NULL AND [Support
Group] = 'TS-AE')"
```

Using special characters. If the search specification contains any of the following characters:

```
= > < ( ) , ~ " ' [
```

it must be enclosed in quotes. This rule applies to operators that are part of the search expression as well as text to search for. If the search expression contains quotes, those quotes must be doubled. For example, in the preceding line of code, notice that the entire search specification is enclosed in double quotes, whereas fields and values referred to within the specification each have single quotes.

If the search object includes a *single* double quote, that quote must be doubled; for example, if you wanted to search for text containing

```
"We must
```

the search specification would take this form:

```
SetSearchSpec "Comments", "'""We must'"
```

so that the initial quote is doubled, and the string containing it is placed within single quotes, and the entire expression, including the single quotes, is placed within double quotes.

If the search specification includes single quotes (including apostrophes), the expression must be placed within single quotes, apostrophes must be doubled, and double quotes must be placed around the entire string. Thus, for example, if you wanted to search for “Phillie’s Cheese Steaks” in the Name field, you would have to enter the specification as follows:

```
SetSearchSpec "Name", "'Phillie''s Cheese Steaks'"
```

NOTE: eScript and Browser Script require backslashes instead of double quotes for marking special characters. For example: `SetSearchSpec("Comments", "\'\"We must\'");` and `SetSearchSpec("Name", "\'Phillie\'s Cheese Steaks\'");`

Searching for text in non-text fields. If the search expression queries a field of any type other than text, or if it is an expression other than a field-level query, text must be placed within quotes if it contains any characters other than the following:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz _ ? \ " ' [
```

As with text field search expressions, quotes must be doubled.

Retrieving all records. To retrieve all records efficiently, use ClearToQuery followed by ExecuteQuery, without using SetSearchSpec.

Searching for a null field. To search for null fields, use the following form:

```
SetSearchSpec "Account", "is NULL"
```

If your search specification requests an empty string, then the search returns every record. For example:

```
SetSearchSpec "Account", ""
```

Used With

Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example

For Siebel VB examples, read "FirstRecord" on page 210, "SetFormattedFieldValue" on page 245, and "BusComp_PreQuery" on page 276. For a Siebel eScript example, read "ClearToQuery" on page 203.

Example

This code searches for a contact by name and then navigates to the record displayed in a view.

```
(general) (declarations)
Option Explicit

Sub Button1_Click

   Dim theCurrComp As BusComp
   Dim TargetView As String
   Dim TargetBusObj As String
   Dim TargetBusComp As String
   Dim NewBusObj As BusObject
   Dim NewComp As BusComp
```

```
   Dim RecId1 As String
   Dim RecId2 As String
   Dim RecId3 As String

   TargetView = "Visible Contact List View"
   TargetBusObj = "Contact"
   TargetBusComp = "Contact"

   Set theCurrComp = TheApplication.ActiveBusComp
   RecId1 = theCurrComp.GetFieldValue("Last Name")
   RecId2 = theCurrComp.GetFieldValue("First Name")
   RecId3 = theCurrComp.GetFieldValue("Account Id")

   Set NewBusObj = TheApplication.GetBusObject(TargetBusObj)
   Set NewComp = NewBusObj.GetBusComp(TargetBusComp)

   NewComp.ActivateField "Last Name"
   NewComp.ActivateField "First Name"
   NewComp.ActivateField "Account Id"



   NewComp.SetSearchSpec "Last Name", RecId1
   NewComp.SetSearchSpec "First Name", RecId2
   NewComp.SetSearchSpec "Account Id", RecId3

   NewComp.ExecuteQuery ForwardBackward

   TheApplication.GotoView TargetView , NewBusObj

End Sub
```

Here is an example in Siebel eScript.

```
var oAccntBO = theApplication().GetBusObject("Account");
var oAccntBC = AccntBO.GetBusComp("Account");
var oAddrBC;

with (oAccntBC)
{
   SetViewMode(SalesRepView);
   ActivateField("Name");
   ClearToQuery();
   SetSearchSpec("Name", "Hong Kong Flower Shop");
   ExecuteQuery()
   oAddrBC = GetMVGBusComp("Street Address")
}
```

```
with (oAddrBC)
{
   NewRecord(NewAfter);
   SetFieldValue("City", "Denver");
   WriteRecord();
}

oAddrBC = null;
oAccntBC = null;
oAccntBO = null;
```

See Also

"ExecuteQuery" on page 206
"ClearToQuery" on page 203
"SetSearchExpr" on page 249
"SetSortSpec"

SetSortSpec

SetSortSpec sets the sorting specification for a query.

Syntax

oBusComp.SetSortSpec *sortSpec*

Argument	Description
sortSpec	String containing the sort specification

Returns

Not applicable

Usage

SetSortSpec, if used, must be called after ClearToQuery and before ExecuteQuery. The sortSpec argument is a string of the form:

```
"fieldName1,fieldName2,...(ASCENDING)"
```

or

```
"fieldName1,fieldName2,...(DESCENDING)"
```

The entire string must be placed in quotes. You can sort on various fields in different orders by separating the field names and order specifications with commas, as in the example.

The argument assigned to SetSortSpec is the same used after the equal sign in a predefined query. For example, the first line following is a sort specification in a predefined query; the second is the equivalent sort specification used with the various interface methods. Note that *Name* is the name of a business component field.

```
'Account'.Sort = "Name(ASCENDING)"
```

```
BC.SetSortSpec "Name(ASCENDING)"
```

Used With

Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example

This Siebel VB example sorts the Opportunity list first by Account in reverse order, then in alphabetical order by Site. Note that the column names in the Opportunity list applet are not the same as those in the underlying business component.

NOTE: This example merely demonstrates how to sort in ascending and descending order. In actual practice you should not sort in both directions in a single sort specification, as it degrades performance considerably.

```
Function BusComp_PreQuery As Integer

With Me
   .ActivateField("Account")
   .ActivateField("Account Location")
   .ClearToQuery
   .SetSortSpec "Account(DESCENDING), Account Location(ASCENDING)"
   .ExecuteQuery
End With

   BusComp_PreQuery = ContinueOperation

End Function
```

Here is the same example in Siebel eScript.

```
Function BusComp_PreQuery

with (this)
{
   ActivateField("Account");
```

```
   ActivateField("Account Location");
   ClearToQuery();
   SetSortSpec("Account(DESCENDING), Account Location(ASCENDING)");
   ExecuteQuery();
}

   return (ContinueOperation);

}
```

See Also "SetSearchExpr" on page 249
"SetSearchSpec" on page 251

SetUserProperty

Sets the value of a named business component user property. The user properties are similar to instance variables of a BusComp.

Syntax *oBusComp*.SetUserProperty *propertyName*, *newValue*

Argument	Description
propertyName	String containing the name of the user property to set
newValue	String containing the property value

Returns Not applicable

Usage The advantage of user properties is that they can be accessed from anywhere in the code (including from other applications through COM) using GetUserProperty. An instance variable, on the other hand, can be accessed only from within Siebel VB from the same object on which the variable is declared.

The value of the property is reset every time you instantiate a new business component.

NOTE: SetUserProperty does not interact directly with user properties defined in Siebel Tools.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example Here is a Siebel VB example.

```
Sub BusComp_SetFieldValue (FieldName As String)
   Select Case FieldName
      Case "Committed"
         me.SetUserProperty "Flagged", "Y"
   End Select
End Sub
```

Here is the same example in Siebel eScript.

```
function BusComp_SetFieldValue (FieldName)
{
   switch (FieldName)
   {
      case "Committed":
         this.SetUserProperty("Flagged", "Y");
   }
}
```

See Also "GetUserProperty" on page 225

SetViewMode

SetViewMode sets the visibility type for the business component. This is used prior to a query.

Siebel VB Syntax *oBusComp*.SetViewMode *mode*

Siebel eScript Syntax *oBusComp*.SetViewMode (*mode)*

where *mode* is a SiebelView Mode constant or its corresponding integer value. The constants shown are defined in three environments.

SiebelViewMode Constant	Integer Value	Comments
SalesRepView	0	Applies single position or sales team access control, and displays records owned by the user's position or records whose sales team contains the user's position, as determined by the business component's Visibility field or Visibility MVField. To use this visibility applet type, the business component must have a view mode with an Owner Type of Position.
ManagerView	1	Displays records that the user and the user's direct reports have access to. Example: My Team's Accounts. Typically used by managers. If the business component on which the view is based uses single position access control, then this constant displays records associated directly with the user's active position and with subordinate positions. If the business component on which the view is based uses sales team access control, then this constant displays records for which the user's active position is the primary position on the team or a subordinate position is the primary member on the team. If a user's position has no subordinate positions, then no data is displayed, not even the user's own data. To use this visibility applet type, the business component must have a view mode with an Owner Type of Position.
PersonalView	2	Displays records the user has direct access to, as determined by the business component's Visibility field. To use this visibility applet type, the business component must have a view mode with an Owner Type of Person. Example: My Accounts. Typically used by individual contributors.
AllView	3	Displays records for the organization where a valid owner has been assigned to the record. Example: All Accounts. Typically used by administrators.

SiebelViewMode Constant	Integer Value	Comments
OrganizationView	5	Applies single organization or multiple organization access control, as determined by the business component's Visibility field or Visibility MVField. To use this visibility applet type, the business component must have a view mode with an Owner Type of Organization. Displays records for organizations where a valid owner has been assigned to the record. Example: All Accounts Across Organizations. Typically used by executives.
GroupView	7	Displays either a list of the category's first level subcategories (child categories) to which the user has access or displays records in the current category, depending on the applet being used. If the user is at the catalog level, then this displays the first level categories.
CatalogView	8	Displays a flat list of records in categories across every catalog to which the user has access. To use this visibility applet type, the business component must have a view mode with an Owner Type of Catalog Category. Typically used in product pick lists and other lists of products, such as a recommended product list.
SubOrganizationView	9	If the business component on which the view is based uses single organization access control, then this constant displays records associated directly with the user's active organization or with a descendent organization. Descendent organizations are defined by the organization hierarchy. To use this visibility applet type, the business component must have a view mode with an Owner Type of Organization. If the business component on which the view is based uses multiple organization access control, then this constant displays records for which the user's active organization or a descendent organization is the primary organization. Example: All Opportunities Across My Organization. Typically used by executives.

Returns Not applicable

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

See Also "GetViewMode" on page 226

Example Here is a Siebel VB example. For another example, see "BusComp_PreDeleteRecord" on page 272.

```
(general) (declarations)
Option Explicit

Dim oBO as BusObject
Dim oBC as BusComp
Set oBO = theApplication.GetBusObject(Me.BusObject.Name)
Set oBC = oBO.GetBusComp(Me.Name)
With oBC
   .SetViewMode SalesRepView
   .ClearToQuery
   .ActivateField "Name"
   .SetSearchSpec "Name", Me.GetFieldValue("Name")
   .SetSearchSpec "Id", "<> " & Me.GetFieldValue("Id")
   .ExecuteQuery ForwardOnly
   If .FirstRecord Then
      theApplication.Trace"Entry for name " &
Me.GetFieldValue("Name")exists already"
   End If
End With
Set oBC = Nothing
Set oBO = Nothing
```

Here is the same example in Siebel eScript.

```
var oBO = TheApplication().GetBusObject(this.BusObject.Name);
var oBC = oBO.GetBusComp(this.Name);
with (oBC)
{
   SetViewMode(SalesRepView);
   ClearToQuery();
   ActivateField("Name");
   SetSearchSpec("Name", this.GetFieldValue("Name"));
   SetSearchSpec("Id", "<> " + this.GetFieldValue("Id");
   ExecuteQuery(ForwardOnly);
   if (FirstRecord)
      theApplication().Trace("Entry for name " +
this.GetFieldValue("Name") + " exists already")
```

```
}
oBC = null;
oBO = null;
```

UndoRecord

UndoRecord reverses any changes made to the record that are not committed. This includes reversing uncommitted modifications to fields, as well as deleting an active record that has not yet been committed to the database.

Syntax *oBusComp*.UndoRecord

Argument	Description
Not applicable	

Returns Not applicable

Usage If you are using UndoRecord to delete a new record, it is useful only after NewRecord has been called and before the new record has been committed. If you are using UndoRecord to reverse changes made to field values, it is useful only before the changes have been committed through a call to WriteRecord, or before the user has stepped off the record through the user interface. UndoRecord reverses uncommitted changes to a record. Therefore, if you wish to have a fine degree of control over which changes are reversed, place the code in the PreNewRecord, PreSetFieldValue, or PreWriteRecord event, and issue a CancelOperation to cancel the change invoked by the particular event.

Used With Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

See Also "NewRecord" on page 234

ViewMode

ViewMode sets and gets the visibility type for the business component.

Syntax *oBusComp*.ViewMode(*mode*)

mode	Where *mode* is a SiebelView Mode constant or its corresponding integer value. The constants shown are defined in the three environments. For details on each SiebelViewMode constant, read "SetViewMode" on page 258. ■ SalesRepView (0) ■ ManagerView (1) ■ PersonalView (2) ■ AllView (3) ■ OrganizationView (5) ■ GroupView (7) ■ CatalogView (8) ■ SubOrganizationView (9)

Returns Not applicable

Used With CORBA Object Manager

See Also "SetViewMode" on page 258 for a list of supported mode values.

WriteRecord

Commits to the database any changes made to the current record.

Syntax *oBusComp*.WriteRecord

Argument	Description
Not applicable	

Returns Not applicable

Usage After creating new records and assigning values to fields, call WriteRecord to commit the new record to the database.

Used With Browser Script, Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

Example This Siebel VB example inserts an activity if the Sales Stage field is set to 02. For other examples, see "GetMVGBusComp" on page 220 and "NewRecord" on page 234.

```
(general) (declarations)
Option Explicit

Sub BusComp_SetFieldValue (FieldName As String)
   ' Run this code from the Opportunities Activities view.
   ' Opportunity is presumed to be the parent business component.

   Select Case FieldName
      Case "Sales Stage"
      if Me.GetFieldValue(FieldName) LIKE "02*" Then
         ' reference the Action business component
         Dim oBCact as BusComp
         Set oBCact = me.BusObject.GetBusComp("Action")
         With oBCact
            .NewRecord NewAfter
            .SetFieldValue "Type", "Event"
            .SetFieldValue "Description", "THRU SVB, Stage _
               changed to 02"
            .SetFieldValue "Done", Format(Now(), _
               "mm/dd/yyyy hh:mm:ss")
            .SetFieldValue "Status", "Done"
            .WriteRecord
         End With
         set oBCact = Nothing
      end if
   End Select
End Sub
```

This CORBA example adds a new customer record. The error-handling routines have been omitted for clarity.

```
int customer::addCustomer(char* fName, char* lName, char* street,
char* city, char* state, char* zip, SiebelApplication* pSeblApp) {

   SiebelBusObject* pBusObj;
   SiebelBusComp_ptr pBusComp;
   short err = 0;

   pBusObj = pSeblApp->GetBusObject("Contact");
   pBusComp = pBusObj->GetBusComp("Contact");

   //Activate necessary fields for customer look-up
   try
   {
      pBusComp->ActivateField("First Name");
      pBusComp->ActivateField("Last Name");
      pBusComp->ActivateField("Personal Street Address");
      pBusComp->ActivateField("Personal City");
      pBusComp->ActivateField("Personal State");
      pBusComp->ActivateField("Personal Postal Code");
   }

   //Create new record
   try
   {
      pBusComp->NewRecord(0);
   }

   //Set field values
   try


   {
      pBusComp->SetFieldValue("First Name", fName);
      pBusComp->SetFieldValue("Last Name", lName);
      pBusComp->SetFieldValue("Personal Street Address", street);
      pBusComp->SetFieldValue("Personal City", city);
      pBusComp->SetFieldValue("Personal State", state);
      pBusComp->SetFieldValue("Personal Postal Code", zip);
   }

   //Commit record
   try
   {
      pBusComp->WriteRecord();
   }

   return 1;
```

```
}
```

Business Component Events

The following topics describe business component events.

- “BusComp_Associate” on page 267
- “BusComp_ChangeRecord” on page 267
- “BusComp_CopyRecord” on page 269
- “BusComp_DeleteRecord” on page 269
- “BusComp_InvokeMethod” on page 270
- “BusComp_NewRecord” on page 271
- “BusComp_PreAssociate” on page 271
- “BusComp_PreCopyRecord” on page 272
- “BusComp_PreDeleteRecord” on page 272
- “BusComp_PreGetFieldValue” on page 274
- “BusComp_PreInvokeMethod” on page 275
- “BusComp_PreNewRecord” on page 275
- “BusComp_PreQuery” on page 276
- “BusComp_PreSetFieldValue” on page 277
- “BusComp_PreWriteRecord” on page 279
- “BusComp_Query” on page 280
- “BusComp_SetFieldValue” on page 281
- “BusComp_WriteRecord” on page 282

BusComp_Associate

The Associate event is called after a record is added to a business component to create an association.

Syntax BusComp_Associate

Argument	Description
Not applicable	

Returns Not applicable

Usage The semantics are the same as for BusComp_NewRecord.

Used With Server Script

See Also "BusComp_NewRecord" on page 271

BusComp_ChangeRecord

The ChangeRecord event is called after a record becomes the current row in the business component.

Syntax BusComp_ChangeRecord

Argument	Description
Not applicable	

Returns Not applicable

Usage Code in the ChangeRecord event handler is executed each time that the focus changes to another record. Avoid lengthy operations in this event handler to enable smooth scrolling in list applets.

Used With Server Script

Example This Siebel VB example uses two subprograms in the (general) (declarations) section to set up an audit trail for service requests. The ChangeRecord event handler is used to initialize the values from the service record so that they can be compared with current values.

```
(general) (declarations)
Option Explicit
Dim OldClosedDate, OldCreated, OldOwner, OldOwnerGroup
Dim OldSeverity, OldSource, OldStatus
Declare Sub CreateAuditRecord
Declare Sub InitializeOldValues

Sub CreateAuditRecord (FieldName As String, NewValue As String,
OldValue As String, ChangedText As String)

   Dim ActionBC As BusComp
   Dim CurrentBO As BusObject
   Dim theSRNumber

   Set CurrentBO = TheApplication.GetBusObject("Service Request")
   Set ActionBC = CurrentBO.GetBusComp("Action")
   theSRNumber = GetFieldValue("SR Number")

   With ActionBC
      .ActivateField "Activity SR Id"
      .ActivateField "Description"
      .ActivateField "Private"
      .ActivateField "Service request id"
      .ActivateField "Type"
      .NewRecord NewAfter

      .SetFieldValue "Activity SR Id",      theSRNumber
      .SetFieldValue "Description",         ChangedText
      .SetFieldValue "Private",               "Y"
      .SetFieldValue "Type",                "Administration"
      .WriteRecord
   End With
End Sub

Sub InitializeOldValues
   OldClosedDate = GetFieldValue("Closed Date")
   OldOwner = GetFieldValue("Owner")
   OldSeverity = GetFieldValue("Severity")
   If GetFieldValue("Severity") <> OldSeverity Then
      NewValue = GetFieldValue("Severity")
      ChangedText = "Changed Priority from " + OldSeverity + _
         " to " + NewValue
```

```
      CreateAuditRecord "Severity", NewValue, OldSeverity, _
         ChangedText
   End If
End Sub

Sub BusComp_ChangeRecord
   InitializeOldValues
End Sub
```

BusComp_CopyRecord

The CopyRecord event is called after a row has been copied in the business component and that row has been made active.

Syntax BusComp_CopyRecord

Argument	Description
Not applicable	

Returns Not applicable

Usage BusComp_CopyRecord is called instead of BusComp_NewRecord when a new record is created:

- Through BusComp.NewRecord NewAfterCopy|NewBeforeCopy
- Through any UI copy record mechanism (Edit > Copy Record; CTRL + B)

Used With Server Script

BusComp_DeleteRecord

The DeleteRecord event is called after a row is deleted. The current context is a different row (the Fields of the just-deleted row are no longer available).

Syntax BusComp_DeleteRecord

Argument	Description
Not applicable	

Usage When a user reads and deletes an existing record or creates and undoes a new record, this invokes DeleteRecord. This invocation causes any associated scripts to be executed.

Returns Not applicable

Used With Server Script

BusComp_InvokeMethod

The InvokeMethod event is called when the InvokeMethod method is called on a business component.

Syntax BusComp_InvokeMethod(*methodName*)

Argument	Description
methodName	String containing the name of the method that was invoked

Returns Not applicable

Usage The InvokeMethod event is called when a specialized method is called on a business component, or when the InvokeMethod method has been explicitly called on a business component.

Used With Server Script

BusComp_NewRecord

The NewRecord event is called after a new row has been created in the business component and that row has been made active. The event may be used to set up default values for Fields.

Syntax BusComp_NewRecord

Argument	Description
Not applicable	

Returns Not applicable

Usage BusComp_NewRecord is called when a new record is created unless the new record was created:

- Through BusComp.NewRecord NewAfterCopy|NewBeforeCopy
- Through any UI copy record mechanism (Edit > Copy Record; CTRL + B)

In these cases, BusComp_CopyRecord is called instead of BusComp_NewRecord.

Used With Server Script

Example For an example, read "Pick" on page 238.

BusComp_PreAssociate

The PreAssociate event is called before a record is added to a business component to create an association. The semantics are the same as for BusComp_PreNewRecord.

Syntax BusComp_PreAssociate

Argument	Description
Not applicable	

Returns ContinueOperation or CancelOperation

Usage CancelOperation stops the execution of the underlying Siebel code associated with the event. However, if there is code in the same script following CancelOperation, that code runs regardless of the CancelOperation.

Used With Server Script

BusComp_PreCopyRecord

The PreCopyRecord event is called before a new row is copied in the business component. The event may be used to perform pre-copy validation.

Syntax BusComp_PreNewRecord

Argument	Description
Not applicable	

Returns ContinueOperation or CancelOperation

Usage CancelOperation stops the execution of the underlying Siebel code associated with the event. However, if there is code in the same script following CancelOperation, that code runs regardless of the CancelOperation.

Used With Server Script

BusComp_PreDeleteRecord

The PreDeleteRecord event is called before a row is deleted in the business component. The event may be used to prevent the deletion or to perform any actions in which you need access to the record that is to be deleted.

Syntax BusComp_PreDeleteRecord

Argument	Description
Not applicable	

Returns ContinueOperation or CancelOperation

Usage This event is called after the user has confirmed the deletion of the record, but before the record is deleted from the database.

CancelOperation stops the execution of the underlying Siebel code associated with the event. However, if there is code in the same script following CancelOperation, that code runs regardless of the CancelOperation.

Used With Server Script

Example This Siebel VB example prevents the deletion of an account that has associated opportunities.

```
(general) (declarations)
Option Explicit

Function BusComp_PreDeleteRecord As Integer
   Dim iReturn as integer
   Dim oBC as BusComp
   Dim oBO as BusObject
   Dim sAcctRowId as string

   iReturn = ContinueOperation
   sAcctRowId = me.GetFieldValue("Id")
   set oBO = TheApplication.GetBusObject("Opportunity")
   set oBC = oBO.GetBusComp("Opportunity")

   With oBC
      .SetViewMode AllView
      .ActivateField "Account Id"
      .ClearToQuery
      .SetSearchSpec "Account Id", sAcctRowId
      .ExecuteQuery ForwardOnly
      If (.FirstRecord) = 1 Then
         RaiseErrorText("Opportunities exist for the Account - _
```

```
            Delete is not allowed")
         iReturn = CancelOperation
      End If
   End With

   BusComp_PreDeleteRecord = iReturn

   Set oBC = Nothing
   Set oBO = Nothing

End Function
```

BusComp_PreGetFieldValue

The PreGetFieldValue event is called when the value of a business component field is accessed.

Syntax BusComp_PreGetFieldValue(*FieldName*, *FieldValue*)

Argument	Description
FieldName	String containing the name of the field accessed
FieldValue	String containing the value accessed

Returns ContinueOperation or CancelOperation

Usage PreGetFieldValue is called at least once for each user interface element that displays the BusComp field value, and it may also be called as a result of other internal uses.

NOTE: PreGetFieldValue is called every time the user interface is updated to repaint fields on the screen. Therefore, a script attached to this event runs very frequently, which may cause the computer to appear to be unresponsive.

CancelOperation stops the execution of the underlying Siebel code associated with the event. However, if there is code in the same script following CancelOperation, that code runs regardless of the CancelOperation.

Used With Server Script

BusComp_PreInvokeMethod

The PreInvokeMethod event is called before a specialized method is invoked on the business component.

Syntax BusComp_PreInvokeMethod(*methodName*)

Argument	Description
methodName	String containing the name of the method invoked

Returns ContinueOperation or CancelOperation

Usage The PreInvokeMethod event is called just before a specialized method is invoked on the business component. Specialized methods are methods based on applet or business component classes other than CSSFrame and CSSBusComp, respectively, that is, specialized classes.

CancelOperation stops the execution of the underlying Siebel code associated with the event. However, if there is code in the same script following CancelOperation, that code runs regardless of the CancelOperation.

Used With Server Script

BusComp_PreNewRecord

The PreNewRecord event is called before a new row is created in the business component. The event may be used to perform preinsert validation.

Syntax BusComp_PreNewRecord

Argument	Description
Not applicable	

Returns ContinueOperation or CancelOperation

Usage CancelOperation stops the execution of the underlying Siebel code associated with the event. However, if there is code in the same script following CancelOperation, that code runs regardless of the CancelOperation.

Used With Server Script

BusComp_PreQuery

The PreQuery event is called before query execution.

Syntax BusComp_PreQuery

Argument	Description
Not applicable	

Returns ContinueOperation or CancelOperation

Usage This event may be used to modify the search criteria or to restrict the execution of certain queries.

CancelOperation stops the execution of the underlying Siebel code associated with the event. However, if there is code in the same script following CancelOperation, that code runs regardless of the CancelOperation.

Used With Server Script

Example

```
Function BusComp_PreQuery() As Integer
   Dim strPosition As String
   Dim strSearchSpec As String
   Dim intReturn As Integer
   intReturn = ContinueOperation
   strPosition = theApplication.PositionName
   strSearchSpec = Me.GetSearchSpec("Owned By")
   If strPosition <> "System Administrator" Then
      if Len(strSearchSpec) = 0 or InStr(strSearchSpec,
         strPosition) = 0 Then
         Me.SetSearchSpec "Owned By", strPosition
      end if
```

```
   End if
   BusComp_PreQuery = intReturn
End Function
```

BusComp_PreSetFieldValue

The PreSetFieldValue event is called before a value is pushed down into the business component from the user interface or through a call to SetFieldValue.

Syntax

BusComp_PreSetFieldValue(*FieldName*, *FieldValue*)

Argument	Description
FieldName	String containing the name of the changed field
FieldValue	String containing the changed value

Returns

ContinueOperation or CancelOperation

Usage

The PreSetFieldValue event is called each time a field is to be changed or populated for a given business component.

When using a picklist to populate multiple fields, PreSetFieldValue is fired for each field that is populated. For example, the Contact Pick Applet is used to populate Last Name, First Name and Contact ID. Therefore, PreSetFieldValue is fired three times, once for each field.

CancelOperation stops the execution of the underlying Siebel code associated with the event. However, if there is code in the same script following CancelOperation, that code runs regardless of the CancelOperation. In the preceding example, if your script returns CancelOperation for a field, that field is not populated. However, PreSetVieldValue still fires for the other two fields populated by the picklist.

NOTE: The PreSetFieldValue event has code that prevents it from running if it is already running on the same BusComp instance (even for other fields). This is to prevent the writing of scripts that might cause infinite recursions.

NOTE: The PreSetFieldValue event in Browser Script is not invoked if the field's Immediate Post Change property is set to TRUE.

Used With

Browser Script, Server Script

Example

This Siebel VB example uses the PreSetFieldValue event to check if a quote discount is greater than 20 percent, and to take appropriate action if it is. For other examples of BusComp_PreSetFieldValue, read "LoginId" on page 169, and "ExecuteQuery" on page 206.

```
Function BusComp_PreSetFieldValue (FieldName As String,
               FieldValue As String) As Integer
'Routine to check if a quote discount>20%
'if it is, notify user and cancel operation
Dim value as Integer
Dim msgtext as String
   If FieldName = "Discount" then
      value = Val(FieldValue)
      If value > 20 then
          msgtext = "Discounts greater than 20% must be approved"
         RaiseError msgtext
         BusComp_PreSetFieldValue = CancelOperation
      Else
         BusComp_PreSetFieldValue = ContinueOperation
       End if
End If
End Function
```

Here is the equivalent example in Siebel eScript.

```
function BusComp_PreSetFieldValue (FieldName, FieldValue)
{
   var msgtext = "Discounts greater than 20% must be approved";

   var iReturn = ContinueOperation
   if (FieldName = "Discount")
   {
      if (FieldValue > 20)
      {
         TheApplication().RaiseErrorText(msgtext);
         return (CancelOperation);
      } else
         return (ContinueOperation)
      }
   }
   else
      return (ContinueOperation);
```

BusComp_PreWriteRecord

The PreWriteRecord event is called before a row is written out to the database. The event may perform any final validation necessary before the actual save occurs.

Syntax

BusComp_PreWriteRecord

Argument	Description
Not applicable	

Returns

ContinueOperation or CancelOperation

Usage

CancelOperation stops the execution of the underlying Siebel code associated with the event. However, if there is code in the same script following CancelOperation, that code runs regardless of the CancelOperation.

The PreWriteRecord event triggers only if a field value was modified or inserted.

Used With

Server Script

Example

```
Function BusComp_PreWriteRecord As Integer
```

```
   ' This code resets the probability before the write
   ' if necessary

   if Me.GetFieldValue("Sales Stage") LIKE "07*" then
      ' Resets the Probability to 75 if less than 75
      if Val(Me.GetFieldValue("Rep %")) < 75 then
         Me.SetFieldValue "Rep %", "75"
      end If
   end if

   BusComp_PreWriteRecord = ContinueOperation
End Function
```

BusComp_Query

The Query event is called just after the query is complete and the rows have been retrieved, but before the rows are actually displayed.

Syntax BusComp_Query

Argument	Description
Not applicable	

Returns Not applicable

Used With Server Script

Example In this Siebel VB example, important information is defined using the Action business component with a special activity type. If the user starts an account query, the code checks whether important information is available. If so, the information is displayed in a message box.

```
Sub BusComp_Query

   Dim oBusObj As BusObject, oCurrFinAct As BusComp,
   Dim oActivities as BusComp, oAppl as Applet
   Dim sName as String, sDescription as String

   On error goto leave
```

```
   set oBusObj = theApplication.ActiveBusObject
   Set oCurrFinAct = theApplication.ActiveBusComp

   If oCurrFinAct.FirstRecord <> 0 then
      sName = oCurrFinAct.GetFieldValue("Name")
      Set oActivities = oBusObj.GetBusComp("Finance _
         Important Info Activity")
      With oActivities
         .ActivateField("Description")
         .ClearToQuery
         .SetSearchSpec "Account Name", sName
         .SetSearchSpec "Type", "Important Info"
         .ExecuteQuery ForwardOnly
         If .FirstRecord <> 0 then
            sDescription = .GetFieldValue("Description")
            theApplication.Trace("Important Information: " +
sDescription)
            do while .NextRecord <> 0
               sDescription = .GetFieldValue("Description")
               theApplication.Trace("Important Information: " +
sDescription)
            loop
         End If
      End With
   End If

leave:

   Set oCurrFinAct = Nothing
   set oBusObj = Nothing

End Sub
```

BusComp_SetFieldValue

The SetFieldValue event is called when a value is pushed down into the business component from the user interface or through a call to SetFieldValue. This event is not triggered for any predefaulted or calculated fields in Siebel Tools.

Syntax BusComp_SetFieldValue(*FieldName*)

Argument	Description
FieldName	String containing the name of the affected field

Returns Not applicable

Used With Server Script

Example This Siebel VB example shows how to invoke methods on an existing BusComp when the SetFieldValue event is triggered.

```
Sub BusComp_SetFieldValue (FieldName As String)
Dim desc As String
Dim newDesc As String
If FieldName = "Type" Then
   newDesc = [can be any valid string containing the new description]
   desc = GetFieldValue("Description")
   SetFieldValue "Description", newDesc
End If
End Sub
```

Here is the equivalent example in Siebel eScript.

```
function BusComp_SetFieldValue (FieldName)
{
   if (FieldName = "Type")
   {
      var newDesc = "Any valid string which contains " +
         "the new description.";
      var desc = GetFieldValue("Description");
      SetFieldValue("Description", newDesc);
   }
}
```

BusComp_WriteRecord

The WriteRecord event is called after a row is written out to the database.

The WriteRecord event triggers after the record has been committed to the database. Do not use SetFieldValue in a WriteRecord event. If you need to use SetFieldValue, put it a PreWriteRecord event (explained in "BusComp_PreWriteRecord" on page 279).

Syntax BusComp_WriteRecord

Argument	Description
Not applicable	

Returns Not applicable

Used With Server Script

Business Object Methods

In the method descriptions, the term *oBusObj* indicates a variable containing a BusObject.

- "GetBusObject" on page 157
- "GetLastErrCode" on page 284
- "GetLastErrText" on page 285
- "Name" on page 285
- "Release" on page 286

GetBusComp

The GetBusComp method returns the specified business component.

Syntax *oBusObj*.GetBusComp (*BusCompName*)

Argument	Description
BusCompName	String containing the desired business component in the business object

Returns The requested business component

Usage *BusCompName* is case-sensitive, and must match in case the form of the name as it appears in Siebel Tools. If an instance of *BusCompName* already exists, that instance is returned. The interpreter instantiates and returns a new instance of a business component using *BusCompName* if one does not already exist.

The Nothing function should be used to destroy the instantiated business component when it is no longer necessary.

NOTE: In Browser Script, the GetBusComp() method can only access business components in the current view; in Server Script, the GetBusComp() method can access every business component that has been instantiated in the active business object.

Used With Browser Script, Server Script, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

GetLastErrCode

The GetLastErrCode method returns the last error code.

Syntax *oBusObj*.GetLastErrCode

Argument	Description
Not applicable	

Returns The last error code as a short integer; 0 indicates no error.

Usage After execution of a method, the GetLastErrCode can be invoked to check if any error was returned from the previous operation. The GetLastErrText method can be invoked to retrieve the text of the error message.

Used With Mobile/Dedicated Web Client Automation Server, COM Data Control

See Also "GetLastErrText" on page 285

GetLastErrText

The GetLastErrText method returns the last error text.

Syntax *oBusObj*.GetLastErrText

Argument	Description
Not applicable	

Returns A string containing the last error text message.

Usage After execution of a method, the GetLastErrCode can be invoked to check if any error was returned from the previous operation. The GetLastErrText method can be invoked to retrieve the text of the error message.

Used With Mobile/Dedicated Web Client Automation Server, COM Data Control

See Also "GetLastErrCode" on page 284

Name

The Name method returns the name of the business object.

Syntax *oBusObj*.Name

Argument	Description
Not applicable	

Returns A string containing the business object name

Used With Browser Script, Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager (implemented as an attribute)

Example For an example, read "Name" on page 234.

Release

The Release()method enables the release of the Business Object and its resources on the Siebel Server.

Syntax *oBusObj.release()*

Argument	Description
Not applicable	

Returns Not applicable

Used With Java Data Bean

Example Here is an example for Java Data Bean:

```
import com.siebel.data.*;

{
...

// create Siebel Data Bean
// login into Siebel Data Bean

...

// Create Siebel Bus Object.
// get the Bus Object from SiebelDataBean

...

// Use the business Object
// Release the business object resources

...

busObj.release();

}
```

Business Service Methods

In the method descriptions, the placeholder *oService* refers to a variable containing a business service.

- "GetFirstProperty"
- "GetLastErrCode" on page 288
- "GetLastErrText" on page 288
- "GetNextProperty" on page 289
- "GetProperty" on page 290
- "InvokeMethod" on page 290
- "Name" on page 292
- "PropertyExists" on page 293
- "Release" on page 293
- "RemoveProperty" on page 294
- "SetProperty" on page 294

GetFirstProperty

This method retrieves the name of the first property of a business service.

Syntax *oService*.GetFirstProperty()

Parameter	Description
Not applicable	

Returns A string containing the name of the first property of the business service

Usage This method retrieves the name of the first property, in order of definition, of a business service. Use GetFirstProperty and GetNextProperty to retrieve the name of a property. You can then use the retrieved name as an argument to GetProperty to retrieve the property value, or with SetProperty to assign property values.

Used With Browser Script, Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Server, Java Data Bean, CORBA Object Manager

See Also "GetNextProperty" on page 289
"GetProperty" on page 290
"SetProperty" on page 294

GetLastErrCode

The GetLastErrCode method returns the most recent error code.

Syntax BusComp.GetLastErrCode

Argument	Description
Not applicable	

Returns The last error code as a short integer; 0 indicates no error.

Usage After execution of a method, the GetLastErrCode can be invoked to check if any error was returned from the previous operation. The GetLastErrText method can be invoked to retrieve the text of the error message.

Used With Mobile/Dedicated Web Client Automation Server

See Also "GetLastErrText"

GetLastErrText

The GetLastErrText method returns the last error text message.

Syntax BusComp.GetLastErrText

Argument	Description
Not applicable	

Returns The most recent error text message as a string

Usage After execution of a method, the GetLastErrCode can be invoked to check if any error was returned from the previous operation. The GetLastErrText method can be invoked to retrieve the text of the error message.

Used With Mobile/Dedicated Web Client Automation Server

See Also "GetLastErrCode" on page 288

GetNextProperty

When the name of the first property has been retrieved, this method retrieves the name of the next property of a business service.

Syntax *oService*.GetNextProperty()

Parameter	Description
Not applicable	

Returns A string containing the name of the next property of a business service, or a null string (`""`) if no more properties exist.

Usage After retrieving the name of the first property with GetFirstProperty, the GetNextProperty method should be used in a loop, to be terminated when a null string (`""`) is returned. When property names have been retrieved, they can be used as parameters to GetProperty to retrieve the property value, or with SetProperty to assign property values.

Used With Browser Script, Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Server, Java Data Bean, CORBA Object Manager

See Also "GetFirstProperty" on page 315
"GetProperty"
"SetProperty" on page 294

GetProperty

The GetProperty method returns the value of the property whose name is specified in its argument.

Syntax *oService*.GetProperty(*propName*)

Parameter	Description
propName	The name of the property whose value is to be returned

Returns A string containing the value of the property indicated by propName or NULL if the property does not exist.

Usage You must know the name of a property to retrieve its value. To retrieve property names, use the GetFirstProperty and GetNextProperty methods.

Used With Browser Script, Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Server, Java Data Bean, CORBA Object Manager

See Also "GetFirstProperty" on page 315
"GetNextProperty" on page 289
"SetProperty" on page 294

InvokeMethod

The InvokeMethod method calls a method on the business service. This can be a documented specialized method or a user-created method.

eScript Syntax

oService.InvokeMethod(*methodName, InputArguments, OutputArguments*)

Siebel VB Syntax

oService.InvokeMethod *methodName, InputArguments, OutputArguments*

Parameter	Description
methodName	A string representing the name of the method to execute
InputArguments	A property set containing the arguments required by the method
OutputArguments	A property set containing the arguments to be returned by the method (passed by reference)

Browser Script Syntax

outputPropSet = *Service*.InvokeMethod(*MethodName, inputPropSet*)

Parameter	Description
methodName	The name of the method
inputPropSet	A property set containing the method arguments
outputPropSet	A property set containing the output arguments of the Property Set

Returns

Not applicable

Usage

Built-in business services work the same way as business component invoke methods. That is, you can call specialized methods on the service that are not exposed directly through the object interface.

Run-time business services can hold user-defined methods, which must be implemented in scripts written in Siebel VB or Siebel eScript. The scripts must be written in these languages within Siebel Tools; however, they can be called through external interfaces.

Although the InvokeMethod function does not return a value, the properties in the *OutputArguments* property set may have been modified.

NOTE: The InvokeMethod method should be used only with documented specialized methods. Siebel Systems does not support calling specialized methods with InvokeMethod, unless they are listed in this book.

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager

See Also "Service_InvokeMethod" on page 295
"Service_PreInvokeMethod" on page 298

Name

The Name property contains the name of the service.

Syntax *oService*.Name

Argument	Description
Not applicable	

Returns A string containing the service name

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean, CORBA Object Manager (implemented as an attribute)

Example Here is a Browser Script example:

```
var svc = theApplication().GetService("Data Quality Manager"):
TheApplication().SWEAlert("The active service is " + svc.Name());
```

PropertyExists

This method returns a Boolean value indicating whether a specified property exists.

Syntax *oService*.PropertyExists(*propName*)

Parameter	Description
propName	A string representing the name of a property of the specified service

Returns In Siebel VB, an integer (0 for false, 1 for true); in other interfaces, a Boolean

Usage Because GetProperty returns a null string ("") for nonexistent properties, you should use PropertyExists() in an if statement to determine whether a specific property has been set.

Used With Browser Script, Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Server, Java Data Bean, CORBA Object Manager

Release

The Release method()enables the release of the Business Service and its resources on the Siebel Server.

Syntax *oBusSvc.release()*

Argument	Description
not applicable	

Returns Not applicable

Used With Java Data Bean

Example For an example, read "Name" on page 234.

RemoveProperty

This method removes a property from a business service.

Syntax *oService*.RemoveProperty(*propName*)

Parameter	Description
propName	A string indicating the name of the property to be removed

Returns Not applicable

Usage This method removes the property *propName* from the business service *oService*. As a result, subsequent calls to PropertyExists for that property returns FALSE.

Used With Browser Script, Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Server, Java Data Bean, CORBA Object Manager

See Also "PropertyExists" on page 293

SetProperty

This method assigns a value to a property of a business service.

Syntax *oService*.SetProperty(*propName, propValue*)

Parameter	Description
propName	A string indicating the name of the property whose value is to be set
propValue	A string containing the value to assign to the property indicated by *propName*

Returns Not applicable

Usage SetProperty is used to set the value of a property of the business service from one of the methods of the service or from an external object.

Used With Browser Script, Server Script, Mobile/Dedicated Web Client Automation Server, COM Data Server, Java Data Bean, CORBA Object Manager

Example For an example, read "Service_PreInvokeMethod" on page 298.

See Also "GetProperty" on page 290

Business Service Events

The following topics describe business service events.

- "Service_InvokeMethod"
- "Service_PreCanInvokeMethod" on page 297
- "Service_PreInvokeMethod" on page 298

Service_InvokeMethod

The InvokeMethod event is called after the InvokeMethod method is called on a business service.

Server Script Syntax

Service_InvokeMethod(*methodName, InputArguments, OutputArguments*)

Parameter	Description
methodName	A string representing the name of the method to execute
InputArguments	A property set containing the arguments required by the method
OutputArguments	A property set containing the arguments to be returned by the method

Browser Script Syntax

OutputArguments = *oService*.InvokeMethod(*methodName, InputArguments*)

Parameter	Description
methodName	A string representing the name of the method to execute
InputArguments	A property set containing the arguments required by the method
OutputArguments	A property set containing the arguments to be returned by the method

Returns

Not applicable

Usage

Although this event does not return a value, it may add properties to, or alter the values of the properties in, the property set *OutputArguments*.

When you invoke business service methods through Browser Script, the business service may be implemented as a browser-based business service (written in JavaScript) or a server-based business service. Initially, the High Interactivity mode framework checks if the business service resides in the browser, and if it does not find it, it sends the request to the server for execution.

NOTE: Browser Script may invoke a browser-based or server-based business service, but Server Script can only invoke a server-based business service.

NOTE: Although the InvokeMethod function does not return a value in Server Script, it may modify the properties in the *OutputArguments* property set.

Used With

Browser Script, Server Script

Example

This Browser Script example invokes the Shipping Engine business service created in "Service_PreInvokeMethod" on page 298 in response to a button click. The InvokeMethod property on the Button is set to "CalcShipping". It gets values from the keyboard through prompts (JavaScript method), passes a property set to the service, and gets return values by means of another property set.

```
function Applet_PreInvokeMethod (name, inputPropSet)
```

```
{
   if (name == "CalcShipping") {
      var svc = theApplication().GetService("Shipping Engine");
      var inputs = theApplication().NewPropertySet();
      var outputs = theApplication().NewPropertySet();

      var size = prompt("Enter the sum of H+W+D in inches");
      var shipper = prompt("Enter the shipping company");
      var weight = prompt("Enter the shipping weight in pounds");

      with (inputs) {
         SetProperty ("Size", size);
         SetProperty ("Shipping Company", shipper);
         SetProperty ("Ship Method", shipMethod);
         SetProperty ("Weight", weight);
         }

      outputs = svc.InvokeMethod("CalculateShipping", inputs);
      var cost = outputs.GetProperty("Cost");
      var delDate = outputs.GetProperty("Delivery Date");

      TheApplication().SWEAlert ("Shipping by " + shipper + ":\n
Shipping Cost is " +
            cost + ",\n Estimated delivery date is " +
            delDate);

      return (CancelOperation);

   }

   else

      return (ContinueOperation);

}
```

See Also "Service_PreInvokeMethod" on page 298

Service_PreCanInvokeMethod

The PreCanInvokeMethod event is called before the PreInvokeMethod, allowing the developer to determine whether or not the user has the authority to invoke the business service method.

Server Syntax Service_PreCanInvokeMethod(*MethodName, &CanInvoke*)

Parameter	Description
MethodName	A string representing the name of the method to be executed
&CanInvoke	A string representing whether or not the business service method can be invoked. Valid values are TRUE and FALSE.

Browser Syntax Service_PreCanInvokeMethod(*MethodName)*

Parameter	Description
MethodName	A string representing the name of the method to be executed

Returns CancelOperation or ContinueOperation

Used With Browser Script, Server Script

Service_PreInvokeMethod

The PreInvokeMethod event is called before a specialized method on the business service is invoked.

Syntax Service_PreInvokeMethod(*methodName, InputArguments, OutputArguments*)

Parameter	Description
methodName	A string representing the name of the method to execute
InputArguments	A property set containing the arguments required by the method
OutputArguments	A property set containing the arguments to be returned by the method

Returns “ContinueOperation” or “CancelOperation”

Usage If implementing a new method, or overriding the behavior of a method implemented in a specialized business service, the script should return CancelOperation to avoid invoking an “Unknown method name” error. As Figure 14 illustrates, this error is predictable if the PreInvokeMethod event is scripted. This occurs because there is no native code to execute in the InvokeMethod event. CancelOperation tells the Siebel application to cancel the remaining operations associated with the event.

NOTE: The example in Figure 14 applies only to new and user-defined methods. For existing standard Siebel methods, it is not necessary to use CancelOperation.

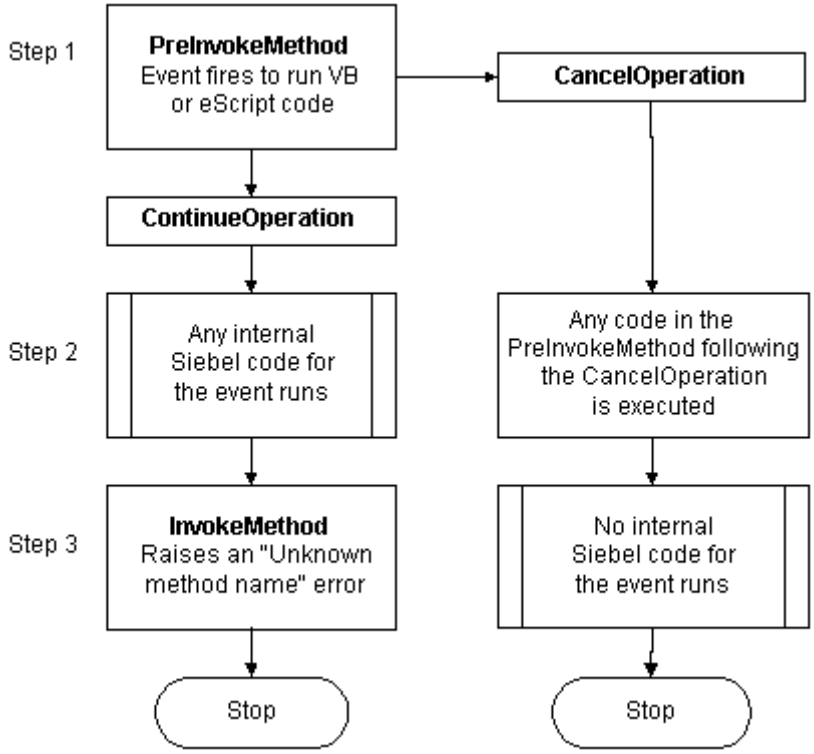


Figure 14. The Effects of CancelOperation and ContinueOperation

Service_InvokeMethod is rarely scripted, but can be used for such post-operation events as posting a notice to a log when the event completes successfully.

Used With Browser Script, Server Script

Example This Siebel VB example creates the new service "Shipping Engine."

```
Function Service_PreInvokeMethod (MethodName As String, Inputs As
PropertySet, Outputs As PropertySet) As Integer

   If MethodName = "CalculateShipping" Then

      Dim sShipper As String, sShipMethod As String
      Dim dWeight As Double, dSize As Double, dCost As Double
      Dim sZone As String, DelDate As Variant
      Dim sCost As String, iReturn As Integer

      iReturn = ContinueOperation
      sShipper = Inputs.GetProperty("Shipping Company")
      sShipMethod = Inputs.GetProperty("Ship Method")
      dWeight = Val(Inputs.GetProperty("Weight"))
      dSize = Val(Inputs.GetProperty("Total Dimensions"))
      iZone = Val(Inputs.GetProperty("Zone"))
      DelDate = DateValue(Now)

      Select Case sShipper
         Case "GlobalEx"
            Select Case sShipMethod
               Case "Next-Day Air"
                  dCost = 14 + dWeight
                  DelDate = DelDate + 1
               Case "Second-Day Air"
                  dCost = 11 + (dWeight * .54)
                  DelDate = DelDate + 2
            End Select

         Case "Airline"
            Select Case sShipMethod
               Case "Next-Day Air"
                  dCost = 5 + (dWeight * .3) + (dSize * .33) + _
                     (Val(sZone) * .5)
                  DelDate = DelDate + 1
               Case "Second-Day Air"
                   dCost = 4 + (dWeight * .3) + (dSize * .2) + _
```

```
                        (Val(sZone) * .3)
                     DelDate = DelDate + 2


                  Case "Ground"
                     dCost = 3 + (dWeight * .18) + (dSize * .1) + _
                        (Val(sZone) * .1)
                     DelDate = DelDate + 2 + Int(Val(sZone) * .8)
               End Select
         End Select

         sCost = Format(dCost, "Currency")
         Outputs.SetProperty "Cost", sCost
         Outputs.SetProperty "Delivery Date", DelDate
         iReturn = CancelOperation

      End If

      Service_PreInvokeMethod = iReturn

   End Function
```

See Also "Service_InvokeMethod" on page 295

Control Methods

In the method descriptions, the placeholder *controlVar* stands for the name of the control on which the method is invoked; for example, `Button1_Click`.

NOTE: Control Methods do not work with ActiveX controls.

- "Applet" on page 302
- "BusComp" on page 302
- "GetProperty" on page 303
- "GetValue" on page 303
- "Name" on page 304
- "SetProperty" on page 307

■ "SetValue" on page 309

Applet

The Applet method returns the parent applet object for a control.

Syntax *controlVar*.Applet

Argument	Description
Not applicable	

Returns The parent applet of the control

Usage Obtaining the parent applet allows you to perform operations on the applet object, not just the control.

Used With Browser Script

BusComp

The BusComp method returns the corresponding business component for the control.

Syntax *controlVar*.BusComp

Argument	Description
Not applicable	

Returns The business component associated with the control's parent applet.

Used With Browser Script

For an example, read "Name" on page 234.

GetProperty

The GetProperty method returns the value of the property of a control.

Syntax *controlVar*.GetProperty(*propName*)

Argument	Description
propName	The name of the property to be set

Returns The value of the property of a control.

Usage GetProperty can be used with the following controls: CheckBox, ComboBox, TextBox, TextArea, and Label.

Use GetProperty to call the following properties: Background Color, Enabled, FontType, FontColor, FontSize, FontStyle, Height, Width, Shown, Read Only, Visible.

If more than one property is to be set, each must be set in a separate statement.

Used With Browser Script

Example This code sample uses GetProperty to return values for FontSize, Background Color, Width, and Height.

```
TheApplication().SWEAlert("checkbox.FontSize : " +
objCheckBox.GetProperty("FontSize"));
TheApplication().SWEAlert("checkbox.BgColor : " +
objCheckBox.GetProperty("BgColor"));
TheApplication().SWEAlert("checkbox.Width : " +
objCheckBox.GetProperty("Width"));
TheApplication().SWEAlert("checkbox.Height : " +
objCheckBox.GetProperty("Height"));
```

GetValue

The GetValue method returns the value of the control. The type of the return value depends on the specific control object.

Syntax *controlVar*.GetValue

Argument	Description
Not applicable	

Returns The value displayed by the control for the data type of the underlying field.

NOTE: GetValue cannot return a literal value input into a control by a user. The method instead returns the value that the user's entry has been stored as, based on the data type of the underlying field.

Usage The GetValue and SetValue methods work only for controls that are associated with business component fields. Therefore, these methods are not applicable to labels.

Used With Browser Script

Name

The Name method returns the name of the object.

Syntax *controlVar*.Name

Argument	Description
Not applicable	

Returns A string containing the object name

Used With Browser Script

Example For an example, read "Name" on page 234.

SetLabelProperty

The SetLabelProperty method sets visual properties of a label.

Syntax *controlVar*.SetLabelProperty(*propName*, *propValue*)

Argument	Description
propName	The name of the property to be set, as described in the following table
propValue	The value to assign to the property, as described in the following table

Returns Not applicable

Usage If more than one property is to be set, each must be set in a separate statement.

The following table lists the properties that can be set for a label, and the values that can be assigned to them.

Property	Value	Description
BgColor	string	Determines Background Color for a label. For example, “Green” or “f0f0f0”
FontColor	string	Determines FontColor for a label. For example, “Green” or “f0f0f0”
FontType	string	Determines FontType for a label. For example, “Times Roman”
FontSize	string	Determines FontSize for a label. For example, “12 pt”
FontStyle	string	Determines FontStyle for a label. For example, “Italic”
FontWeight	string	Determines FontWeight for a label. Acceptable values are normal, bold, bolder, lighter, 100, 200, 300, 400 (equivalent to normal), 500, 600, 700 (equivalent to bold), 800, and 900. Default is normal. For example, “800”
Height	string	Determines Height for a label, in pixels. For example, “5”

Property	Value	Description
Visible	`visible` or `hidden`	Determines whether the label is visible. Defaults to repository definition unless explicitly modified by using SetLabelProperty.
Width	string	Determines Width for a label, in pixels. For example, “80”

The SetLabelProperty method is not enabled by default. You must enable it in Siebel Tools before using it in a script. To enable the SetLabelProperty, expand the Control node in the Tools Object Explorer and select the Control User Prop node. Then add a new Control User Prop named “useLabelID” with a value of “TRUE”.

Used With Browser Script

Example The following code shows the use of SetLabelProperty:

```
function Applet_PreInvokeMethod (name, inputPropSet){
   // example of changing the Font Size of the Location label
   if (name == "fontsize") {
      var ctl = this.FindControl("Location");
      var fSize = prompt("Please specify the desired label font
size (numeric value only).");
      ctl.SetLabelProperty("FontSize", fSize);
      return ("CancelOperation");
   }

   // example of changing the Background Color of the Location
label
   else if (name == "bgcolor") {
      var ctl = this.FindControl("Location");
      var bgColor = prompt("Specify the background color of the
label. Please enter a valid RGB value");
      ctl.SetLabelProperty("BgColor", bgColor);
      return ("CancelOperation");
   }

   // example of changing the Font Type of the Location label
   else if (name == "fonttype") {
      var ctl = this.FindControl("Location");
      var fontType = prompt("Please specify the font type for the
label");
```

```
      ctl.SetLabelProperty("FontType", fontType);
      return ("CancelOperation");
   }

   // example of changing the Font Color of the Location label
   else if (name == "fontcolor") {
      var ctl = this.FindControl("Location");
      var fontColor = prompt("Specify the font color of the label.
Please enter a valid RGB value");
      ctl.SetLabelProperty("FontColor", fontColor);
      return ("CancelOperation");
   }
   else
      return ("ContinueOperation");
   }
```

SetProperty

The SetProperty method sets visual properties of a control.

Syntax *controlVar*.SetProperty(*propName*, *propValue*)

Argument	Description
propName	The name of the property to be set, as described in the following table
propValue	The value to assign to the property, as described in the following table

Returns Not applicable

Usage SetProperty can be used with the following controls: CheckBox, ComboBox, TextBox, and TextArea.

If more than one property is to be set, each must be set in a separate statement.

The following table lists the properties that can be set for a control, and the values that can be assigned to them.

Property	Value	Description
BgColor	string	Determines Background Color for a control. For example, “Green” or “f0f0f0”
Enabled	TRUE or FALSE	Is the button active? (Unless explicitly modified by using SetProperty, default is TRUE.)
FontColor	string	Determines FontColor for a control. For example, “Green” or “f0f0f0”
FontType	string	Determines FontType for a control. For example, “Times Roman”
FontSize	string	Determines FontSize for a control. For example, “12 pt”
FontStyle	string	Determines FontStyle for a control. For example, “Bold”
Height	string	Determines Height for a control, in pixels. For example, “5”
Shown	TRUE or FALSE	Is the control shown? (Unless explicitly modified by using SetProperty, default is as defined in the repository.)
ReadOnly	TRUE or FALSE	Determines whether the control is read-only. Defaults to repository definition unless explicitly modified by using SetProperty.
Visible	TRUE or FALSE	Determines whether the control is visible. Defaults to repository definition unless explicitly modified by using SetProperty.
Width	string	Determines Width for a control, in pixels. For example, “80”

Used With Browser Script

Example The following code shows the use of SetProperty:

```
objCheckBox.SetProperty("FontColor", "f0f0f0");
objCheckBox.SetProperty("FontStyle", "italic");
objCheckBox.SetProperty("FontType", "Verdana");
objCheckBox.SetProperty("FontSize", "25pt");
objCheckBox.SetProperty("BgColor", "00f000");
objCheckBox.SetProperty("Width", "100");
objCheckBox.SetProperty("Height", "100");
```

SetValue

The SetValue method sets the contents of the specified control to the value indicated.

Syntax *controlVar*.SetValue (*controlValue*)

Argument	Description
controlValue	String containing the value to which to set the control

Returns Not applicable

Usage The GetValue and SetValue methods work only for controls that are associated with business component fields. Therefore, these methods are not applicable to labels. SetValue sets the contents of a control. The user can still change those contents before they are committed to the BusComp field.

Used With Browser Script

Property Set Methods

In the method descriptions, the placeholder *oPropSet* refers to a variable containing a property set.

- "AddChild" on page 310
- "Copy" on page 311
- "GetChild" on page 313

- "GetChildCount" on page 314
- "GetFirstProperty" on page 315
- "GetNextProperty" on page 316
- "GetProperty" on page 317
- "GetPropertyCount" on page 318
- "GetType" on page 319
- "GetValue" on page 319
- "InsertChildAt" on page 320
- "PropertyExists" on page 321
- "RemoveChild" on page 321
- "RemoveProperty" on page 322
- "Reset" on page 323
- "SetProperty" on page 323
- "SetType" on page 324
- "SetValue" on page 325

AddChild

The AddChild method is used to add subsidiary property sets to a property set, so as to form hierarchical (tree-structured) data structures.

Syntax *oPropSet*.AddChild(*childPropSet as PropertySet)*

Parameter	Description
childObject	A property set to be made subsidiary to the property set indicated by *oPropSet*

Returns An integer indicating the index of the child property set.

Usage Property sets can be used to create tree-structured data structures. Any number of arbitrarily structured child property sets can be added to a property set. You may use child property sets to structure a property set in a manner similar to the data model. For example, the parent property set might be Account, with child property sets for opportunities, contacts, activities, and so on. At the same time, you could construct an independent property set called Opportunity, to which accounts, contacts, and activities might be children.

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

Example The following fragment of eScript code shows how child property sets may be added to a parent property set.

```
var Account = TheApplication().NewPropertySet();
var Opportunity = TheApplication().NewPropertySet();
var Contact = TheApplication().NewPropertySet();
var Activity = TheApplication().NewPropertySet();

Account.AddChild(Opportunity);
Account.AddChild(Contact);
Account.AddChild(Activity);
```

See Also "GetChild" on page 313
"InsertChildAt" on page 320
"RemoveChild" on page 321

Copy

This method returns a copy of a property set.

Syntax *oPropSet*.Copy()

Parameter	Description
Not applicable	

Returns A copy of the property set indicated by *oPropSet*

Usage This method creates a copy of a property set, including any properties and children it may have. Because property sets are generally passed by reference, making a copy allows the method to manipulate the property set without affecting the original definition.

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

Example This Siebel VB example uses a copy of a property set to store the original values of its properties, and displays both the original and Pig-Latin forms of the properties.

```
(general) (declarations)
Option Explicit

Function PigLatin (Name1 As String) As String
   Dim Name2 As String, FirstLetter As String
   Name2 = Right$(Name1, len(Name1) - 1)
   FirstLetter = Left$(Name1, 1)
   Name2 = UCase(Mid$(Name1, 2, 1)) & _
      Right$(Name2, Len(Name2) - 1)
   Name2 = Name2 & LCase(FirstLetter) & "ay"
   PigLatin = Name2
End Function

(Sub ClickMe_Click)

   Dim Inputs As PropertySet, Outputs As PropertySet
   Dim message As String, propName, propVal, newPropVal
   set Inputs = theApplication.NewPropertySet

   Inputs.SetProperty "Name", "Harold"
   Inputs.SetProperty "Assistant", "Kathryn"
   Inputs.SetProperty "Driver", "Merton"

   set Outputs = Inputs.Copy()

   propName = Outputs.GetFirstProperty()
   do while propName <> ""
      propVal = Outputs.GetProperty(propName)
      newPropVal = PigLatin(propVal)
      Outputs.SetProperty propName, newPropVal
      message = message & propVal & " has become " & _
         newPropVal & Chr$(13)
```

```
        propName = Outputs.GetNextProperty()
    loop
    TheApplication.RaiseErrorText message

End Sub
```

GetChild

Syntax

GetChild returns a specified child property set of a property set.

oPropSet.GetChild(*index*)

Parameter	Description
index	An integer representing the index number of the child property set to be retrieved

Returns

The property set at index *index* of the parent property set

Usage

When child property sets are created, each is given an index number within the parent property set, starting at 0. Property sets added using AddChild get the next available index number. However, a property set added using InsertChildAt inserts a new property set at a specified index. The property set previously at that index, and every property set after it, have their indexes increased by 1. Similarly, a property set removed using RemoveChild decreases the indexes of following child property sets by 1.

NOTE: This method returns the number of direct descendants only. That is, if the child property sets have children of their own, these grandchildren are not included in the computation of the return value.

Used With

Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

Example

This Siebel eScript example sets the Name property of child property sets to the same value.

```
function Test1_Click ()
{
   var Account = TheApplication().NewPropertySet();
   var Opportunity = TheApplication().NewPropertySet();
   var Contact = TheApplication().NewPropertySet();
   var Activity = TheApplication().NewPropertySet();
   var j;

   Account.AddChild(Opportunity);
   Account.AddChild(Contact);
   Account.AddChild(Activity);

   for (var i = 0; i < Account.GetChildCount(); i++)
   {
      j = Account.GetChild(i);
      j.SetProperty('Name', 'Allied Handbooks');
   }
}
```

See Also "AddChild" on page 310
"InsertChildAt" on page 320

GetChildCount

This method returns the number of child property sets attached to a parent property set.

Syntax *oPropSet*.GetChildCount()

Parameter	Description
Not applicable	

Returns The number of child property sets subordinate to *oPropSet*

Usage This method returns the actual number of child property sets of *oPropSet*. Because index numbers for child property sets start at 0, a child count of 3 indicates that there are child property sets at indexes 0, 1, and 2.

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

Example For an example, read "GetChild" on page 313.

GetFirstProperty

This method returns the name of the first property in a property set.

Syntax *oPropSet*.GetFirstProperty()

Parameter	Description
Not applicable	

Returns A string representing the name of the first property in a property set

Usage GetFirstProperty() retrieves the name of the first property, in order of definition, of a business service. Use GetFirstProperty and GetNextProperty to retrieve the name of a property. You can then use the retrieved name as an argument to GetProperty to retrieve the property value, or with SetProperty to assign property values.

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

Example This example uses GetFirstProperty to get the first property, then retrieves all subsequent properties using GetNextProperty. The loop terminates when GetNextProperty retrieves a null.

```
function Service_PreInvokeMethod (MethodName, Inputs, Outputs)
{
   var propName = "";
   var propVal = "";

   propName = Inputs.GetFirstProperty();
```

```
      // stay in loop if the property name is not null
      // or a null string
      while ((propName != "") && (propName != null)) {
         propVal = Inputs.GetProperty(propName);

         // if a property with the same name does not exist
         // add the name value pair to the output
         if (!Outputs.PropertyExists(propName)) {
            Outputs.SetProperty(propName, propVal);
         }

         propName = Inputs.GetNextProperty();

      }
      return (CancelOperation);
   }
```

See Also "GetNextProperty"
"GetProperty" on page 317

GetNextProperty

This method returns the next property in a property set.

Syntax *oPropSet*.GetNextProperty()

Parameter	Description
Not applicable	

Returns A string representing the name of the next property in a property set

Usage After retrieving the name of the first property with the GetFirstProperty method, GetNextProperty should be used in a loop, to be terminated when a null string ("") is returned. When property names have been retrieved, they may be used as parameters to GetProperty to retrieve the property value, or with SetProperty to assign property values.

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

Example This example uses GetFirstProperty to get the first property, then retrieves all subsequent properties using GetNextProperty. The loop terminates when GetNextProperty retrieves a null.

```
function Service_PreInvokeMethod (MethodName, Inputs, Outputs)
{
   var propName = "";
   var propVal = "";

   propName = Inputs.GetFirstProperty();

   // stay in loop if the property name is not null
   // or a null string
   while ((propName != "") && (propName != null)) {
      propVal = Inputs.GetProperty(propName);

      // if a property with the same name does not exist
      // add the name value pair to the output
      if (!Outputs.PropertyExists(propName)) {
         Outputs.SetProperty(propName, propVal);
      }

      propName = Inputs.GetNextProperty();

   }
   return (CancelOperation);
}
```

See Also "GetFirstProperty" on page 315
"GetProperty"

GetProperty

This method returns the value of a property when given the property name.

Syntax *oPropSet*.GetProperty(*propName*)

Parameter	Description
propName	A string representing the name of a property as returned by GetFirstProperty or GetNextProperty

Returns A string representing the value stored in the property indicated by *propName*, or an empty string (`""`) if the property does not exist

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

Example The following fragment of Siebel eScript code receives a set of input properties used with the Shipping Engine service described in "Service_PreInvokeMethod" on page 298.

```
var Inputs = TheApplication().NewPropertySet();

var sShipper = Inputs.GetProperty("Shipping Company");
var dWeight = Val(Inputs.GetProperty("Weight"));
var dSize = Val(Inputs.GetProperty("Total Dimensions"));
var iZone = Val(Inputs.GetProperty("Zone"));
```

Here is an example for C++:

```
CORBA::string_var inputType = inputPS.type;
```

or

```
char typeBuffer[40];

strcpy(typeBuffer, inputPS.type);
```

See Also
"GetFirstProperty" on page 315
"GetNextProperty" on page 316
"SetProperty" on page 323

GetPropertyCount

This method returns the number of properties attached to a property set.

Syntax *oPropSet*.GetPropertyCount

Parameter	Description
Not applicable	

Returns The number of properties contained within a property set

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server

GetType

This method retrieves the data value stored in the type attribute of a property set.

Syntax *oPropSet*.GetType

Parameter	Description
Not applicable	

Returns A string representing the value stored in the type attribute of the property set

Usage Type, like value, is a special storage location for a data value.

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

See Also "GetValue"
"SetType" on page 324

GetValue

This method retrieves the data value stored in the value attribute of a property set.

Syntax *oPropSet*.GetValue

Parameter	Description
Not applicable	

Returns A string representing the data value stored in the value attribute of a property set

Usage Value, like type, is a special storage location for a data value.

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

See Also "GetProperty" on page 317
"GetType" on page 319
"SetValue" on page 325

InsertChildAt

This method inserts a child property set into a parent property set at a specific location.

Syntax *oPropSet*.InsertChildAt *childObject*, *index*

Parameter	Description
childObject	A property set to be made subsidiary to the property set indicated by *oPropSet*
index	An integer representing the position at which *childObject* is to be inserted

Returns Not applicable

Usage This method inserts the property set *childObject* at the location *index*. Index numbers start at 0. When a child property set is inserted, the property set previously at the location *index* has its index increased by 1, as do subsequent child property sets.

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

See Also "AddChild" on page 310

PropertyExists

This method returns a Boolean value indicating whether a specified property exists in a property set.

Syntax *oPropSet*.PropertyExists(*propName*)

Parameter	Description
propName	A string representing the name of the property to be found

Returns In Siebel VB, an integer (0 for false, 1 for true); in other interfaces, a Boolean

Usage Because GetProperty returns a null string ("") for every nonexistent property, use PropertyExists() in an if statement to determine whether a specific property has been set.

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

Example For an example, read "GetFirstProperty" on page 315.

RemoveChild

This method removes a child property set from a parent property set.

Syntax *oPropSet*.RemoveChild *index*

Parameter	Description
index	An integer representing the index number of the child property set to be removed

Returns Not applicable

Usage When a child property set is removed, every child property set with an index higher than that of the removed set has its index decremented by 1.

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

Example The following Siebel VB code fragment removes every child property set of a property set.

```
Dim i As Integer
for i = 0 to outputs.GetChildCount()
   outputs.RemoveChild(0)
Next i
```

See Also "AddChild" on page 310
"InsertChildAt" on page 320

RemoveProperty

This method removes a property from a property set.

Syntax *oPropSet*.RemoveProperty *propName*

Parameter	Description
propName	The name of the property to be removed

Returns Not applicable

Usage This method removes the property *propName* from the property set *oPropSet*.

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

Reset

This method removes every properties and child property set from a property set.

Syntax *oPropSet*.Reset()

Parameter	Description
Not applicable	

Returns Not applicable

Usage This method removes every property and children from a property set, allowing the property set to be reused with new properties.

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

SetProperty

This method assigns a data value to a property in a property set.

Syntax *oPropSet*.SetProperty *propName, propValue*

Parameter	Description
propName	A string representing the name of a property
propValue	A string representing the value to be assigned to *propName*

Returns Not applicable

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

Example This Siebel VB fragment makes use of the business service "Shipping Engine," which is illustrated in "Service_PreInvokeMethod" on page 298.

```
Dim Svc As Service
Dim Inputs As PropertySet, Outputs As PropertySet
Set Svc = theApplication.GetService("Shipping Engine")
Set Inputs = theApplication.NewPropertySet()

With Inputs
   .SetProperty "Shipping Company", "Airline"
   .SetProperty "Weight", "12"
   .SetProperty "Total Dimensions", "48"
   .SetProperty "Shipping Method", "Second-Day Air"
End With
```

See Also "GetProperty" on page 317

SetType

This method assigns a data value to the type attribute of a property set.

Syntax *oPropSet*.SetType *type*

Parameter	Description
type	A string representing data to be stored in the type attribute

Returns Not applicable

Usage Type, like value, is a special storage location for a data value.

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

See Also "GetType" on page 319
"SetValue" on page 325

SetValue

This method assigns a data value to the value attribute of a property set.

Syntax *oPropSet*.SetValue *value*

Parameter	Description
value	A string representing data to be stored in the value attribute

Returns Not applicable

Usage Values, like properties and types, are storage locations for a data value.

Used With Browser Script, Server Script, Web Client Automation Server, Mobile/Dedicated Web Client Automation Server, COM Data Control, COM Data Server, Java Data Bean

See Also "GetValue" on page 319
"SetProperty" on page 323
"SetValue"

SiebelAppFactory Methods

The following methods are for SiebelApp.

- "CreateSiebelAppObject"
- "Release" on page 327

CreateSiebelAppObject

CreateSiebelAppObject returns a CORBA object reference for an Application object. An object reference for the SiebelAppFactory is available when the Object Request Broker is initialized. You can then use SiebelAppFactory.CreateSiebelAppObject to create Application object references.

Syntax factoryRef- > CreateSiebelAppObject()

Argument	Description
Not applicable	

Returns A CORBA object reference for a Siebel Application Object.

Usage Obtain the SiebelAppFactory object reference as shown in the following example. Then invoke SiebelAppFactory.CreateSiebelAppObject to obtain an object reference for a Siebel Application object.

Used With CORBA Object Manager

Example This C++ example illustrates how to get a reference for the SiebelAppFactory.

```
char* strServerName = new char[SERVER_NAME_LEN];
char* strHostName = "SCOMSERVER";
// append the host name to the server name
strcpy(strServerName,"SiebelObjectFactory:SiebelCorbaServer");
try {
      // bind to the Factory object in the server.
      appFact = SiebelAppFactory::_bind(strServerName,
strHostName);
   } catch(const CORBA::SystemException& excep) {
      cout << "Error binding to SiebelObjectFactory" <<
      endl;
      return;
   }
```

CreateSiebelAppObject returns the reference to the Siebel Application Object.

```
SiebelApplication_var appObj;

try {
      appObj = appFact->CreateSiebelAppObject();
      cout << "Created Siebel object" << endl;
      }
   catch (const SiebelException& seblExcep) {
      cerr << "Error id : " << seblExcep.id << endl;
      cerr << "Error text : " << endl;
```

```
      cerr << seblExcep.desc << endl;
      return;
      }

try {
   // call the login method to access the Siebel application
   appObj->Login("CCONWAY", "CCONWAY");
   // Get the reference to the Accounts business object
   pBusObj = appObj->GetBusObject("Account");
} catch (SiebelException& excep) {

   cerr << "Error code : " << excep.id << endl;
   cerr << "Error text : " << excep.desc << endl;
   return;
}
```

Release

The Release method releases resources on the server for a particular client application. AppFactory is the only valid object reference held by this method. After running this method, you must terminate the client or obtain a new Application object through SiebelAppFactory.CreateSiebelAppObject.

Syntax *AppFactory.Release(appObj)*

Parameter	Description
appObj	Siebel Application object

Used With CORBA Object Manager

Miscellaneous Methods

The following methods do not belong to any other category.

- "GetErrorCode" on page 328
- "GetErrorMessage();" on page 328
- "TheApplication" on page 329

GetErrorCode

This method is used with the Java Data Bean to display numeric error codes.

Syntax public int getErrorCode()

Parameter	Description
Not applicable	

Returns A numeric error code

Used With Java Data Bean

See Also "GetErrorMessage();"

GetErrorMessage();

This method is used with the Java Data Bean to display error messages.

Syntax public string getErrorMessage()

Parameter	Description
Not applicable	

Returns A string containing an error message

Used With Java Data Bean

See Also "GetErrorCode"

TheApplication

TheApplication is a global method that returns the unique object of type Application. This is the root of objects within the Siebel Applications object hierarchy. Use this method to determine the object reference of the application, which is later used to find other objects or to invoke methods on the application object.

Browser Script Syntax

theApplication()

VB Syntax

theApplication

eScript Syntax

TheApplication()

Argument	Description
Not applicable	

Returns

Application, an object for use in finding other objects or invoking methods

Usage

For convenience, the Siebel applications provide the shortcut constant theApplication.

To determine if you are logged in to a server database or local database, use TheApplication.invokemethod("GetDataSource").

Used With

Browser Script, Server Script

Example

Here is an example in VB Script to retrieve the login name from the application object and create the Employee business object:

```
Dim oEmpBusObj as BusObject
Dim sLoginName as String

sLoginName = theApplication.LoginName
Set oEmpBusObj = theApplication.GetBusObject("Employee")

Set oEmpBusObj = Nothing
```

Accessing Siebel COM Data Server with C++ 4

This chapter presents a series of steps to build a simple COM client in Visual C++ and the Microsoft Foundation Class (MFC) library, which accesses the Siebel Data Server. Use this to build real-time interfaces to Siebel using C++ for integration purposes.

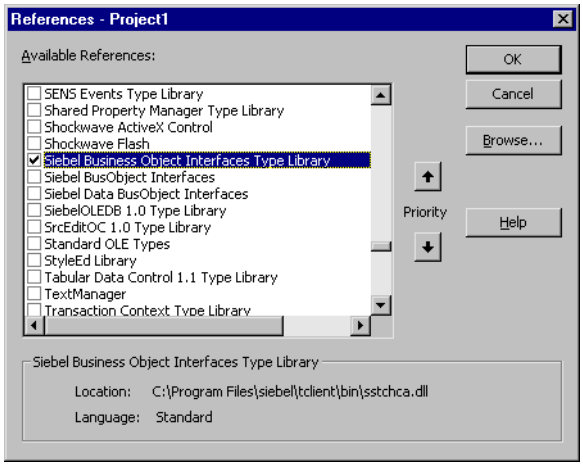
Building the Siebel COM Client in C++

To build the Siebel COM client in C++

1. In Microsoft Visual C++, choose File > New > Project.
2. Select the `MFC AppWizard (exe)` project type.
3. In the Project name field, enter `SiebelCOM`, and then click OK.

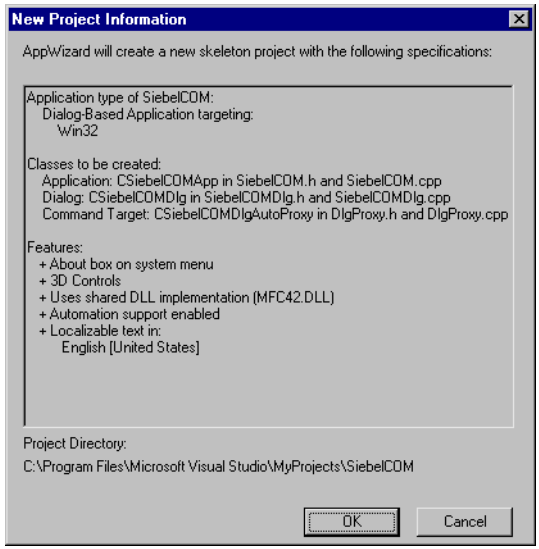
   The MFC AppWizard starts.
4. Check the Dialog-based option and then click Next.

5 In the “What other support would you like to include?” frame, check Automation and clear ActiveX Controls, and then click Next. Click Next again.



6 Click Finish.

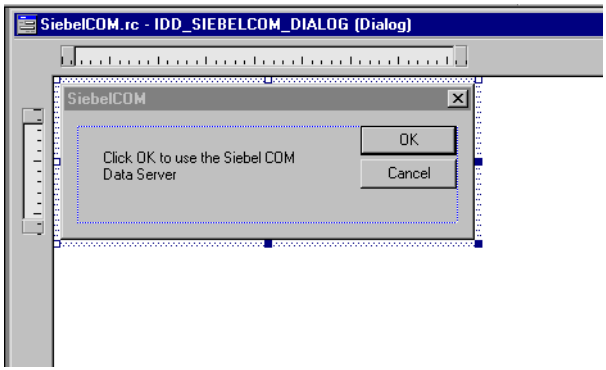
Microsoft Visual C++ displays the project information.



7 Click OK.

The Application Wizard generates the standard MFC code that serves as the skeleton for this project. Headers and libraries necessary to support COM automation are included. Refer to the Microsoft Visual Studio [MSDN] documentation for a detailed description of the MFC libraries.

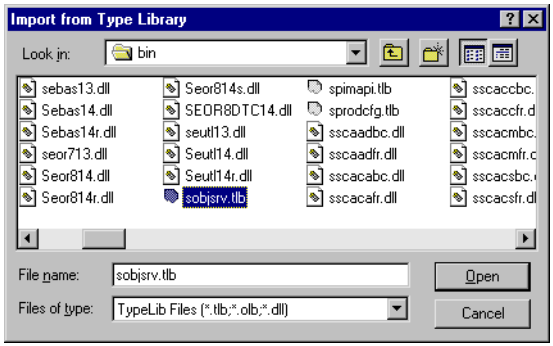
8 The newly created dialog box appears in the workspace. You can resize the box and change the text in the label by editing its properties. Right-click the label in the dialog box to edit its properties. Modify the dialog box so that it looks something like this:



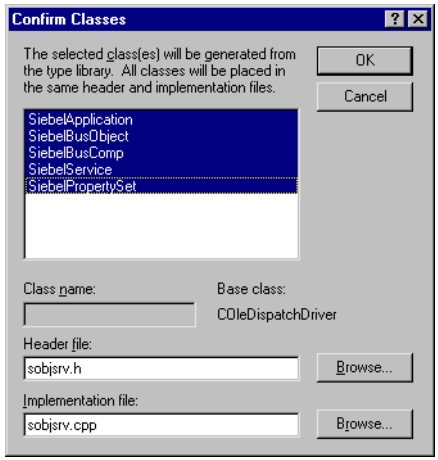
9 Choose View > ClassWizard > Automation.

10 Click Add Class > From a type library.

11 Navigate to the `C:\Sea750\client\bin` folder. Choose `sobjsrv.tlb`.



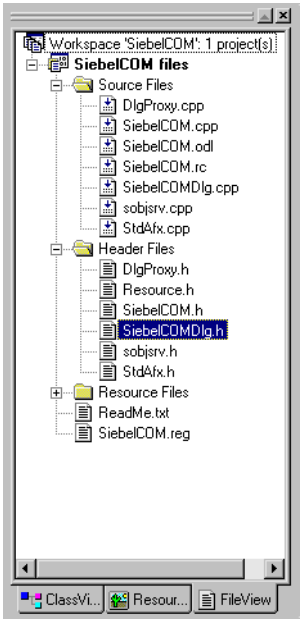
12 In the Confirm Classes dialog box, make sure all five Siebel classes are selected, and then click OK. Click OK again to close the Class Wizard.



13 Add code to communicate with the Siebel COM Server.

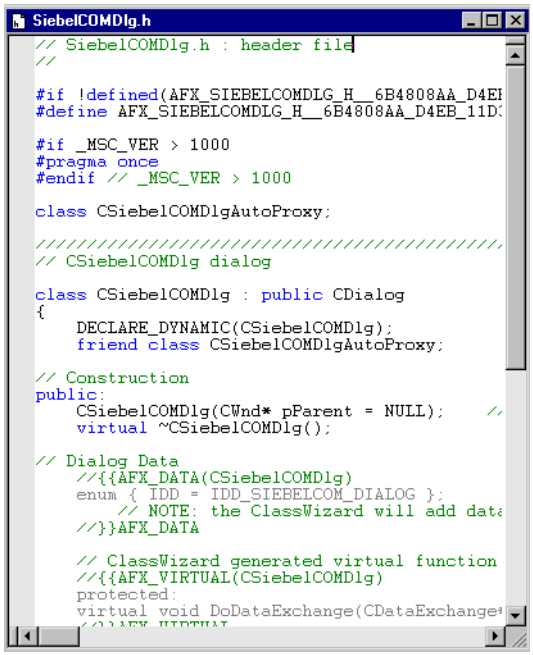
a In the workspace window, click the FileView tab.

b Expand the Source Files and Header Files folders, as shown.



c Double-click the `SiebelCOMDlg.h` file.

The code window opens.



d Enter the code that is highlighted in boldface in Figure 15 into the `SiebelCOMDlg.h` file.

```
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "sobjsrv.h"   //include Siebel wrapper classes

class CSiebelCOMDlgAutoProxy;

/////////////////////////////////////////////////////////////////////////////
// CSiebelCOMDlg dialog

class CSiebelCOMDlg : public CDialog{
     DECLARE_DYNAMIC(CSiebelCOMDlg);
     friend class CSiebelCOMDlgAutoProxy;
     SiebelApplication sApp;         //declare Siebel object

// Construction
public:
     CSiebelCOMDlg(CWnd* pParent = NULL);// standard constructor
     virtual ~CSiebelCOMDlg();
```

Figure 15. Code for SiebelCOMDlg.h

e Choose File > Open and select the `SiebelCOMDlg.cpp` file. Add the code that is highlighted in boldface in Figure 16 to the `OnInitDialog` procedure.

```
BOOL CSiebelCOMDlg::OnInitDialog()
{
   CDialog::OnInitDialog();

   // Add “About...” menu item to system menu

   // IDM_ABOUTBOX must be in the system command range.
   ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
   ASSERT(IDM_ABOUTBOX < 0xF000);

   CMenu* pSysMenu = GetSystemMenu(FALSE);
   if (pSysMenu != NULL)
   {
      CString strAboutMenu;
      strAboutMenu.LoadString(IDS_ABOUTBOX);
      if (!strAboutMenu.IsEmpty())
      {
         pSysMenu->AppendMenu(MF_SEPARATOR);
         pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
      }
   }

   // Set the icon for this dialog. The framework does this
   // automatically when the application’s main window
   // is not a dialog
   SetIcon(m_hIcon, TRUE); // Set big icon
   SetIcon(m_hIcon, FALSE); // Set small icon

   // TODO: Add extra initialization here
   // Start the Siebel Data Server
   if (!sApp.CreateDispatch(_T("SiebelDataServer.ApplicationObject")))
   {
      AfxMessageBox("Cannot start Siebel Data Server.");
      EndDialog(-1); //fail
   } else
   {
      AfxMessageBox("Siebel Data Server initialized.");
   }

   return TRUE; // return TRUE unless you set the focus to a control
}
```

Figure 16. Code to Be Added to OnInitDialog Routine in SiebelCOMDlg.cpp

In the same file, add the code that is highlighted in boldface in Figure 17 and Figure 18 to the `OnOKDialog` procedure. Make sure that the line beginning with `sApp.LoadObjects` points to the location of the CFG file you intend to use. In the line beginning with `sApp.Login`, make sure that you have entered a valid logon name and password.

```
void CSiebelCOMDlg::OnOK()
{

    short sErr;

    //Load Configuration File
        // Make sure that The  following line points to the configuration
        // file you intend to use!
    sApp.LoadObjects("C:\\siebel\\bin\\siebel.cfg", &sErr);
    if(sErr)
    {
        AfxMessageBox("LoadObjects failed.");
        return;
    } else
    {
        AfxMessageBox("CFG file loaded.");
    }

    //Login as Sadmin
    sApp.Login("SADMIN", "SADMIN", &sErr);
    if (sErr)
    {
        AfxMessageBox("Login failed.");
        return;
    } else
    {
        AfxMessageBox("Logged into Siebel database.");
    }

    //Get Account BusObject
    LPDISPATCH lpdBo;
    lpdBo = sApp.GetBusObject("Account", &sErr);
    if (sErr)
    {
        AfxMessageBox("GetBusObject failed.");
        return;
    } else
    {
        AfxMessageBox("Account BusObject retrieved.");
```

Figure 17. Code to be Added to OnOKDialog Routine in SiebelCOMDlg.cpp

```
//Get Account BusComp
    LPDISPATCH lpdBc;
    lpdBc = Bo.GetBusComp("Account", &sErr);
    if (sErr)
    {
        AfxMessageBox("GetBusComp failed.");
        return;
    } else
    {
        AfxMessageBox("Account BusComp retrieved.");
    }
    SiebelBusComp Bc(lpdBc);

    //Get the name of the first account
    Bc.ActivateField("Name", &sErr);
    if (sErr) return;
    Bc.ClearToQuery(&sErr);
    if (sErr) return;
    Bc.SetSearchSpec("Name","*",&sErr);
    if (sErr) return;
    Bc.ExecuteQuery(0,&sErr);
    if (sErr) return;
    Bc.FirstRecord(&sErr);
    if (sErr) return;

    //Display the account name in a message box
    CString csAcctName;
    csAcctName = Bc.GetFieldValue("Name", &sErr);
    AfxMessageBox(csAcctName);

    return;

    if (CanExit())
        CDialog::OnOK();
}
```

Figure 18. Code to Be Added to OnOKDialog Routine in SiebelCOMDlg.cpp

When you have finished creating your program, test it to make sure it works properly.

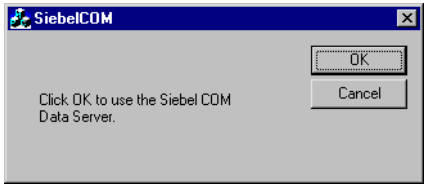
Testing Your Program

To test your program

1. Start your Siebel client application using the same CFG file and login parameters you specified in the code.
2. Choose Screens > Accounts > All Accounts. Verify that there is at least one account visible in the Account list applet. If there is not, create one. Exit the Siebel client.
3. Open the CFG file you specified in the code and make sure that the `DataSource` key indicates the database source you specified at logon in Step 2.
4. In Microsoft Visual C++, choose Build > Build SiebelCOM.exe, or press F7. If there are any errors or warnings reported in the output window, correct the errors and repeat this step.
5. Choose Build > Execute SiebelCOM.exe, or press F5.

   A message box displays the message "Siebel Data Server initialized."
6. Click OK.

   The customized dialog box opens.



7. The application displays a series of message boxes, with the following messages:

   "CFG file loaded."
   "Logged into Siebel database."
   "Account BusObject retrieved."
   "Account BusComp retrieved."

   The application displays the name of the first account in the All Accounts view.

COM Data Control Quick Reference 5

This quick reference has the following topics:

- "Application"
- "Business Component" on page 348
- "Business Object" on page 352
- "Business Service" on page 352
- "Property Set" on page 353

Application

Table 25 lists a summary of the Application Methods syntax.

Table 25. Application Methods Syntax Summary

Method	Description	Syntax

Attach	Allows an external application to reconnect to an existing Siebel session.	`Dim application as SiebelDataControl`
		`Dim status as Boolean`
		`status = applicaton.Attach(sessionID As String)`

CurrencyCode	Returns the three-letter operating currency code.	`Dim application as SiebelApplication`
		`Dim sCur as String`
		`sCur = Application.CurrencyCode`

Detach	Returns a string containing the Siebel session ID.	`Dim application as SiebelDataControl`
		`Dim sessionId as String`
		`sessionId = applicaton.Detach()`

EnableExceptions	Enables/disables native COM error handling.	`Dim application as SiebelApplication`
		`Dim bEnable as Boolean`
		`bEnable = application.EnableExceptions(`*`bEnable`*`)`

Table 25. Application Methods Syntax Summary

Method	Description	Syntax

GetBusObject	Instantiates and returns a new instance of the business object specified in the argument.	`Dim application as SiebelApplication`
		`Dim busObject as SiebelBusObject`
		`set busObject = application.GetBusObject(`*`busobjName`* `as String)`

GetLastErrCode	Returns the last error code.	`Dim application as SiebelApplication`
		`Dim iErr as Integer`
		`iErr = GetLastErrCode`

GetLastErrText	Returns the last error text message.	`Dim application as SiebelApplication`
		`Dim sText as String`
		`sText = application.GetLastErrText`

GetProfileAttr	Returns the value of an attribute in a user profile.	

GetService	Instantiates and returns a new instance of the argument-specified service.	`Dim oApplication as SiebelApplication`
		`Dim oService as SiebelService`
		`set Service = Application.GetService(`*`serviceName`* `as String)`

GetSharedGlobal	Returns the shared user-defined global variables.	

InvokeMethod	Calls the named specialized method.	`Dim application as SiebelApplication`
		`Dim sReturn as String`
		`sReturn = application.InvokeMethod(`*`methodName`* `as String,` *`methodArgs`* `as String or StringArray)`

Login	Allows external applications to log in to the COM Data Server.	`Dim application as SiebelApplication`
		`Dim sErr as String`
		`sErr = application.Login(`*`connectString`* `as String,` *`userName`* `as String,` *`password`* `as String)`

LoginId	Returns the login ID of the user who started the Siebel application.	`Dim application as SiebelApplication`
		`Dim sID as String`
		`sID = application.LoginId`

LoginName	Returns the login name of the user who started the Siebel application.	`Dim application as SiebelApplication`
		`Dim sUser as String`
		`sUser = application.LoginName`

Logoff	Disconnects the client from the server.	`Dim application as SiebelApplication`
		`Dim SiebApp as SiebelDataControl`
		`boolVal=siebApp.LogOff()`

NewPropertySet	Constructs and returns a new property set object.	`Dim application as SiebelApplication`
		`Dim PropSet as ProperySet`
		`PropSet = oApplication.NewPropertySet()`

Table 25. Application Methods Syntax Summary

Method	Description	Syntax

PositionId	Returns the position ID that describes the user's current position.	`Dim application as SiebelApplication`
		`Dim sRow as String`
		`sRow = application.PositionId`

PositionName	Returns the position name of the user's current position.	`Dim application as SiebelApplication`
		`Dim sPosition as String`
		`sPosition = application.PositionName`

SetPositionId	Sets the active position to the Position ID specified in the argument.	`Dim application as SiebelDataControl`
		`Dim status as Boolean`
		`status = application.SetPositionId(sPosId)`

SetPositionName	Sets the active position to the position name specified in the argument. Returns a Boolean value indicating whether or not method succeeded.	`Dim application as SiebelDataControl`
		`Dim status as Boolean`
		`status = application.SetPositionName(sPosName)`

SetProfileAttr	Used in personalization to assign values to attributes in a user profile.	`Dim application as SiebelApplication` `application.SetProfileAttr(`*`name`* `as String,` *`value`* `as String)`

SetSharedGlobal	Sets a shared user-defined global variable, which may be accessed using GetSharedGlobal.	`Dim application as SiebelApplication`
		`Dim SiebApp as SiebelDataControl`
		`boolVal=SetSharedGlobal(`*`varName`* `As String,` *`value`* `As String)`

Trace	Appends a message to the trace file.	`Dim application as SiebelApplication`
		`Dim SiebApp as SiebelDataControl`
		`boolVal=siebApp.TraceOn(`*`msg`* `As String) As Boolean`

TraceOff	Turns off the tracing started by the TraceOn method.	`Dim application as SiebelApplication`
		`Dim SiebApp as SiebelDataControl`
		`boolVal=siebApp.TraceOff as Boolean`

TraceOn	Turns on the tracking of allocations and deallocations of Siebel objects, and SQL statements generated by the Siebel application.	`Dim application as SiebelApplication`
		`Dim SiebApp as SiebelDataControl`
		`boolVal=siebApp.TraceOn(`*`fileName`* `As String,` *`category`* `As String,` *`src`* `As String) As Boolean`

Business Component

Table 26 lists a summary of the Business Component Methods syntax.

Table 26. Business Component Methods Syntax Summary

Method	Description	Syntax
ActivateField	Allows queries to retrieve data for the specified field.	`Dim busComp as SiebelBusComp` `BusComp.ActivateField(`*`fieldName`*` as String)`
ActivateMultipleFields	Allows queries to retrieve data for the fields specified in the property set.	`Dim busComp as SiebelBusComp` `busComp.ActivateMultipleFields(oPropSet as SiebelPropertySet)`
Associate	Creates a new many-to-many relationship for the parent object through an association business component.	`Dim busComp as SiebelBusComp` `busComp.Associate(`*`whereIndicator`*` as Integer)`

BusObject	Returns the business object that contains the business component.	`Dim busComp as SiebelBusComp`
		`Dim busObject as SiebelBusObject`
		`Set busObject = busComp.BusObject`

ClearToQuery	Clears the current query and sort specifications on the business component.	`Dim busComp as SiebelBusComp` `busComp.ClearToQuery`
DeactivateFields	Deactivates every currently activated field.	`Dim busComp as SiebelBusComp` `busComp.DeactivateFields`
DeleteRecord	Removes the current record from the business component.	`Dim busComp as SiebelBusComp` `busComp.DeleteRecord`
ExecuteQuery	Retrieves a set of BusComp records.	`ExecuteQuery(cursorMode As Integer) As Boolean`
ExecuteQuery2	Retrieves a set of BusComp records.	`ExecuteQuery(cursorMode As Integer,ignoreMaxCursorSize As Integer) As Boolean`

FirstRecord	Moves to the first record in the business component.	`Dim busComp as SiebelBusComp`
		`Dim bIsRecord as Boolean`
		`bIsRecord = busComp.FirstRecord`

FirstSelected	Returns the association business component.	`Dim busComp as SiebelBusComp`
		`Dim AssocBusComp as SiebelBusComp`
		`Set AssocBusComp = busComp.GetAssocBusComp`

Table 26. Business Component Methods Syntax Summary

Method	Description	Syntax

GetFieldValue	Returns a value for the field specified in the argument.	`Dim busComp as SiebelBusComp`
		`Dim sValue as String`
		`sValue = busComp.GetFieldValue(`*`FieldName`*` as String)`

GetFormattedFieldValue	Returns a formatted value for the field specified in the argument.	`Dim busComp as SiebelBusComp`
		`Dim sValue as String`
		`sValue = busComp.GetFormattedFieldValue(`*`FieldName`*` as String)`

GetLastErrCode	Returns the most recent error code.	`Dim errCode As Integer`
		`Dim SiebApp as SiebelDataControl`
		`errCode=siebApp.GetLastErrCode`

GetLastErrText	Returns the most recent error text message.	`Dim busComp as SiebelBusComp`
		`Dim sErr as String`
		`busComp.GetLastErrText`

GetMultipleFieldValues	Returns a value for the fields specified in the property set.	`Dim busComp as SiebelBusComp` `busComp.GetMultipleFieldValues(oFieldNames as SiebelPropertySet, oFieldValues as SiebelPropertySet)`

GetMVGBusComp	Returns the MVG business component associated with the field specified in the argument.	`Dim busComp as SiebelBusComp`
		`Dim mVGBusComp as SiebelBusComp`
		`set mVGBusComp = busComp.GetMVGBusComp(`*`FieldName`*` as String)`

GetNamedSearch	Returns the argument-named search specification.	`Dim busComp as SiebelBusComp`
		`Dim sValue as String`
		`sValue = busComp.GetNamedSearch(`*`SearchName`*` as String)`

GetPicklistBusComp	Returns the pick business component associated with the field specified in the argument.	`Dim busComp as SiebelBusComp`
		`Dim pickBusComp as SiebelBusComp`
		`Set pickBusComp = busComp.GetPicklistBusComp(FieldName as String)`

GetSearchExpr	Returns the current search expression.	`Dim busComp as SiebelBusComp`
		`Dim sExpr as String`
		`sExpr = busComp.GetSearchExpr`

GetSearchSpec	Returns the current search specification for the field specified in the argument.	`Dim busComp as SiebelBusComp`
		`Dim sSpec as String`
		`sSpec = busComp.GetSearchSpec(`*`FieldName`*` as String)`

GetUserProperty	Returns the value of a named user property.	`Dim application as SiebelApplication`
		`Dim SiebApp as SiebelDataControl`
		`retStr=GetUserProp(prop As String) As String`

Table 26. Business Component Methods Syntax Summary

Method	Description	Syntax

GetViewMode	Returns the visibility mode for the business component.	`Dim busComp as SiebelBusComp`
		`Dim iMode as Integer`
		`iMode = busComp.GetViewMode`

InvokeMethod	Calls the specialized method named in the argument.	`Dim busComp as SiebelBusComp`
		`Dim sReturn as String`
		`sReturn = busComp.InvokeMethod(`*`methodName`* `as String,` *`methodArgs`* `as String or StringArray)`

LastRecord	Moves to the last record in the business component.	`Dim busComp as SiebelBusComp`
		`Dim bReturn as Boolean`
		`bReturn = busComp.LastRecord`

Name	Returns the name of the business component.	`Dim busComp as SiebelBusComp`
		`Dim sName as String`
		`sName = busComp.Name`

NewRecord	Adds a new record to the business component.	`Dim busComp as SiebelBusComp` `busComp.NewRecord(`*`whereIndicator`* `as Integer)`

NextRecord	Moves to the next record in the business component.	`Dim busComp as SiebelBusComp`
		`bReturn as Boolean`
		`bReturn = busComp.NextRecord`

ParentBusComp	Returns the parent business component.	`Dim busComp as SiebelBusComp`
		`Dim parentBusComp as SiebelBusComp`
		`Set parentBusComp = busComp.ParentBusComp`

Pick	Places the currently selected record in a picklist business component into the appropriate fields of the parent business component.	`Dim busComp as SiebelBusComp` `busComp.Pick`

PreviousRecord	Moves to the previous record in the business component.	`Dim busComp as SiebelBusComp`
		`Dim bReturn as Boolean`
		`bReturn = busComp.PreviousRecord`

RefineQuery	Refines a query after a query has been executed.	`Dim busComp as SiebelBusComp` `busComp.RefineQuery`

SetFieldValue	Assigns a new value to the named field for the current row of the business component.	`Dim busComp as SiebelBusComp`
		`busComp.SetFieldValue(`*`FieldName`* `as String,`
		`FieldValue` `as String)`

Table 26. Business Component Methods Syntax Summary

Method	Description	Syntax

SetFormattedFieldValue	Accepts the field value in the current local format and assigns the new value to the named field for the current row of the business component.	`Dim busComp as SiebelBusComp`
		`busComp.SetFormattedFieldValue(`*`FieldName`* `as String,`
		`FieldValue` `as String)`

SetMultipleFieldValues	Assigns a new value to the fields specified in the property set for the current row of the business component.	`Dim busComp as SiebelBusComp` `BusComp.SetMultipleFieldValues(oPropSet as SiebelPropertySet)`

SetNamedSearch	Sets a named search specification on the business component.	`Dim busComp as SiebelBusComp`
		`busComp.SetNameSearch(`*`searchName`* `as String,`
		`searchSpec` `as String)`

SetSearchExpr	Sets the search specification for the business component.	`Dim busComp as SiebelBusComp` `busComp.SetSearchExpr(`*`searchSpec`* `as String)`

SetSearchSpec	Sets the search specification for the specified field.	`Dim busComp as SiebelBusComp`
		`busComp.SetSearchSpec(`*`FieldName`* `as String,`
		`searchSpec as String)`

SetSortSpec	Sets the sort specification for a query.	`Dim busComp as SiebelBusComp` `busComp.SetSortSpec(`*`sortSpec`* `as String)`
SetUserProperty	Not supported for COM Data Control.	

SetViewMode	Sets the visibility type for the business component.	`Dim application as SiebelApplication`
		`Dim SiebApp as SiebelDataControl`
		`boolVal=siebBC.SetViewMode(mode As Integer) As Boolean`

UndoRecord	Deletes an active record created by NewRecord.	`Dim busComp as SiebelBusComp` `busComp.UndoRecord`
WriteRecord	Commits to the database any changes made to the current record.	`Dim busComp as SiebelBusComp` `busComp.WriteRecord`

Business Object

Table 27 lists a summary of the Business Object Methods syntax.

Table 27. Business Object Methods Syntax Summary

Method	Description	Syntax

GetBusComp	Returns the specified business component.	`Dim busObject as SiebelBusObject`
		`Dim busComp as SiebelBusComp`
		`set busComp = BusObject.GetBusComp(`*`BusCompName`* `as String)`

GetLastErrCode	Returns the most recent error code.	`Dim busObject as SiebelBusObject`
		`Dim iErr as Integer`
		`busObject.GetLastErrCode`

GetLastErrText	Returns the most recent error text.	`Dim busObject as SiebelBusObject`
		`Dim sErr as String`
		`busObject.GetLastErrText`

Name	Returns the name of the control.	`Dim busObject as SiebelBusObject`
		`Dim sName as String`
		`sName = busObject.Name`

Business Service

Table 28 lists a summary of the Business Service Methods syntax.

Table 28. Business Service Methods Syntax Summary

Method	Description	Syntax

GetFirstProperty	Retrieves the name of the first property of a business service.	`Dim oService as SiebelService`
		`Dim sName as String`
		`sName = oService.GetFirstProperty()`

GetNextProperty	Once the name of the first property has been retrieved, retrieves the name of the next property of a business service.	`Dim oService as SiebelService`
		`Dim sName as String`
		`sName = oService.GetNextProperty()`

GetProperty	Retrieves the value stored in the specified property.	`Dim oService as SiebelService`
		`Dim sValue as String`
		`sValue = oService.GetProperty(`*`propName`* `as String)`

Table 28. Business Service Methods Syntax Summary

Method	Description	Syntax

Name	Returns the name of the business service.	`Dim oService as SiebelService`
		`Dim sName as String`
		`sName = oService.Name`

InvokeMethod	Calls a specialized method or a user-created method on the business service.	`Dim oService as SiebelService`
		`Dim Return`
		`Return = oService.InvokeMethod(methodName as String, InputArguments as SiebelPropertySet, OutputArguments as SiebelPropertySet)`

PropertyExists	Returns a Boolean value indicating whether the property specified in the argument exists.	`Dim oService as SiebelService` `oService.PropertyExists(`*`propName`*` as String)`
RemoveProperty	Removes a property from a business service.	`Dim oService as SiebelService` `oService.RemoveProperty(`*`propName`*` as String)`
SetProperty	Assigns a value to a property of a business service.	`Dim oService as SiebelService` `oService.SetProperty(`*`propName`*` as String, `*`propValue`*` as String)`

Property Set

Table 29 lists a summary of the Property Set Methods syntax.

Table 29. Property Set Methods Syntax Summary

Method	Description	Syntax

AddChild	Adds subsidiary property sets to a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim iIndex as Integer`
		`oPropSet.AddChild(`*`childObject`*` as Property Set)`

Copy	Returns a copy of a property set.	`Dim oPropSet1 as SiebelPropertySet`
		`Dim oPropSet2 as SiebelPropertySet`
		`oPropSet2 = oPropSet1.Copy()`

GetChild	Returns a specified child property set of a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim sPropVal as String`
		`sPropVal = oPropSet.GetChild(`*`index`*` as Integer)`

Table 29. Property Set Methods Syntax Summary

Method	Description	Syntax

GetChildCount	Returns the number of child property sets attached to a parent property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim iCount as Integer`
		`iCount = oPropSet.GetChildCount()`

GetFirstProperty	Returns the name of the first property in a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim sPropName as String`
		`sPropName = oPropSet.GetFirstProperty()`

GetNextProperty	Returns the name of the next property in a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim sPropName as String`
		`sPropName = oPropSet.GetNextProperty()`

GetProperty	Returns the value of a property when given the property name.	`Dim oPropSet as SiebelPropertySet`
		`Dim sPropVal as String`
		`sPropVal = oPropSet.GetProperty(`*`propName`*` as String)`

GetPropertyCount	Returns the number of properties attached to a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim count as Long`
		`count = oPropSet .GetPropertyCount`

GetType	Returns the value stored in a type in a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim sTypeVal as String`
		`sTypeVal = oPropSet.GetType(`*`value`*` as String)`

GetValue	Returns a value stored as part of a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim sValVal as String`
		`sValVal = oPropSet.GetValue(`*`value`*` as String)`

InsertChildAt	Inserts a child property set into a parent property set at a specific location.	`Dim oPropSet as SiebelPropertySet` `oPropSet.InsertChildAt(childObject as SiebelPropertySet, index as Long)`
PropertyExists	Returns a Boolean value indicating whether the property specified in the argument exists.	`Dim oPropSet as Property Set` `oPropSet.PropertyExists(`*`propName`*` as String)`
RemoveChild	Removes a child property set as a specified index from a parent property set.	`Dim oPropSet as SiebelPropertySet` `oPropSet.RemoveChild(index as Long)`
RemoveProperty	Removes the property specified in its argument from a property set.	`Dim oPropSet as SiebelPropertySet` `oPropSet.RemoveProperty(`*`propName`*` as String)`
Reset	Removes every property and child property set from a property set.	`Dim oPropSet as SiebelPropertySet` `oPropSet.Reset()`

Table 29. Property Set Methods Syntax Summary

Method	Description	Syntax
SetProperty	Assigns a value to the property of a property set specified in its argument.	`Dim oPropSet as SiebelPropertySet` `oPropSet.SetProperty(`*`propName`* `as String,` *`propValue`* `as String)`
SetType	Assigns a data value to a type member of a property set.	`Dim oPropSet as SiebelPropertySet` `oPropSet.SetType(`*`value`* `as String)`
SetValue	Assigns a data value to a value member of a property set.	`Dim oPropSet as SiebelPropertySet` `oPropSet.SetValue(`*`value`* `as String)`

COM Data Server Quick Reference 6

This quick reference has the following topics:

- "Application"
- "Business Component" on page 360
- "Business Object" on page 364
- "Business Service" on page 364
- "Property Set" on page 365

Application

Table 30 lists a summary of the Applications Methods syntax.

Table 30. Application Methods Syntax Summary

Method	Description	Syntax

CurrencyCode	Returns the three-letter operating currency code.	`Dim application as SiebelApplication`
		`Dim sCur as String`
		`sCur = Application.CurrencyCode(ErrCode as Integer)`

GetBusObject	Instantiates and returns a new instance of the business object specified in the argument.	`Dim application as SiebelApplication`
		`Dim busObject as SiebelBusObject`
		`set busObject = application.GetBusObject(`*`busobjName`*` as String, ErrCode as Integer)`

GetLastErrCode	Returns the last Siebel error number.	`Dim application as SiebelApplication`
		`Dim iErrNum as Integer`
		`iErrNum = application.GetLastErrCode(ErrCode as Integer)`

GetLastErrText	Returns the last error text message.	`Dim application as SiebelApplication`
		`Dim sText as String`
		`sText = application.GetLastErrText(ErrCode as Integer)`

Table 30. Application Methods Syntax Summary

Method	Description	Syntax
GetProfileAttr	Returns the value of an attribute in a user profile.	`GetProfileAttr(name As String, errCode As Integer) As String`

GetService	Instantiates and returns a new instance of the argument-specified service.	`Dim Application as SiebelApplication`
		`Dim Service as SiebelService`
		`set Service = Application.GetService(`*`serviceName`* `as String, ErrCode as Integer)`

GetSharedGlobal	Gets the shared user-defined global variables.	`Dim application as SiebelApplication`
		`Dim sName as String`
		`sName = application.GetSharedGlobal(`*`varName`* `as String, ErrCode as Integer)`

LoadObjects	Starts the COM Data Server object and returns a reference to the Application object.	`Dim application as SiebelApplication`
		`Dim returned as SiebelApplication`
		`set returned = application.LoadObjects(`*`pathName\cfgFileName`* `as String, ErrCode as Integer)`

Login	Allows external applications to log in to the COM Data Server.	`Dim application as SiebelApplication`
		`Dim sErr as String`
		`sErr = application.Login(`*`userName`* `as String,` *`password`* `as String, ErrCode as Integer)`

LoginId	Returns the login ID of the user who started the Siebel application.	`Dim application as SiebelApplication`
		`Dim sID as String`
		`sID = application.LoginId(ErrCode as Integer)`

LoginName	Returns the login name of the user who started the Siebel application.	`Dim application as SiebelApplication`
		`Dim sUser as String`
		`sUser = application.LoginName(ErrCode as Integer)`

NewPropertySet	Constructs and returns a new property set object.	`Dim oApplication as SiebelApplication`
		`Dim oPropSet as ProperySet`
		`oPropSet = oApplication.NewPropertySet()`

PositionId	Returns the position ID that describes the user's current position.	`Dim application as SiebelApplication`
		`Dim sRow as String`
		`sRow = application.PositionId(ErrCode as Integer)`

PositionName	Returns the position name of the user's current position.	`Dim application as SiebelApplication`
		`Dim sPosition as String`
		`sPosition = application.PositionName(ErrCode as Integer)`

Table 30. Application Methods Syntax Summary

Method	Description	Syntax

SetPositionId	Sets the active position to the position ID specified in the argument. Returns a Boolean value indicating if the method succeeded.	`Dim application as SiebelWebApplication`
		`Dim posId as String`
		`Dim status as Boolean`
		`status = application.SetPositionId(posId as String, ErrCode as Integer)`

SetPositionName	Sets the active position to the position name specified in the argument. Returns a Boolean value indicating if the method succeeded.	`Dim application as SiebelWebApplication`
		`Dim posName as String`
		`Dim status as Boolean`
		`status = application.SetPositionName(posName as String, ErrCode as Integer)`

SetProfileAttr	Used in personalization to assign values to attributes in a user profile.	`Dim application as SiebelApplication` `application.SetProfileAttr(`*`name`*` as String, `*`value`*` as String, ErrCode as Integer)`
SetSharedGlobal	Sets a shared user-defined global variable.	`Dim application as SiebelApplication` `application.SetSharedGlobal(`*`varName`*` as String, value as String, ErrCode as Integer)`
Trace	Appends a message to the trace file.	`Dim application as SiebelApplication` `application.Trace(`*`message`*` as String, ErrCode as Integer)`
TraceOff	Turns off the tracing started by TraceOn.	`Dim application as SiebelApplication` `application.TraceOff(ErrCode as Integer)`
TraceOn	Turns tracing on	`Dim application as SiebelApplication` `application.TraceOn(`*`filename`*` as String, `*`type`*` as Integer, Selection as String, ErrCode as Integer)`

Business Component

Table 31 lists a summary of the Business Component Methods syntax.

Table 31. Business Component Methods Syntax Summary

Method	Description	Syntax
ActivateField	Allows queries to retrieve data for the specified field.	`Dim busComp as SiebelBusComp` `busComp.ActivateField(`*`fieldName`*` as String, ErrCode as Integer)`
ActivateMultipleFields	Allows queries to retrieve data for the fields specified in the property set.	`Dim buscomp as SiebelBusComp` `buscomp.ActivateMultipleFields(oPropSet as SiebelPropertySet, ErrCode as Integer)`
Associate	Creates a new many-to-many relationship for the parent object through an association business component.	`Dim busComp as SiebelBusComp` `busComp.Associate(`*`whereIndicator`*` as Integer, ErrCode as Integer)`

BusObject	Returns the business object that contains the business component.	`Dim busComp as SiebelBusComp`
		`Dim busObject as BusObject`
		`Set busObject = busComp.BusObject(ErrCode as Integer)`

ClearToQuery	Clears the current query and sort specifications on the business component.	`Dim busComp as SiebelBusComp` `busComp.ClearToQuery(ErrCode as Integer)`
DeactivateFields	Deactivates every currently activated field.	`Dim busComp as SiebelBusComp` `busComp.DeactivateFields(ErrCode as Integer)`
DeleteRecord	Removes the current record from the business component.	`Dim busComp as SiebelBusComp` `busComp.DeleteRecord(ErrCode as Integer)`
ExecuteQuery	Retrieves a set of BusComp records.	`Dim busComp as SiebelBusComp` `busComp.ExecuteQuery(`*`cursorMode`*` as Boolean, ErrCode as Integer)`
ExecuteQuery2	Retrieves a set of BusComp records.	`Dim busComp as SiebelBusComp` `busComp.ExecuteQuery2(`*`cursorMode`*` as Boolean, `*`ignoreMaxCursorSize`*` as Boolean, ErrCode as Integer)`

FirstRecord	Moves to the first record in the business component.	`Dim busComp as SiebelBusComp`
		`Dim bIsRecord as Boolean`
		`bIsRecord = busComp.FirstRecord(ErrCode as Integer)`

Table 31. Business Component Methods Syntax Summary

Method	Description	Syntax

FirstSelected	Returns the association business component.	`Dim busComp as SiebelBusComp`
		`Dim AssocBusComp as BusComp`
		`Set AssocBusComp = busComp.GetAssocBusComp(ErrCode as Integer)`

GetFieldValue	Returns a value for the field specified in the argument.	`Dim busComp as SiebelBusComp`
		`Dim sValue as String`
		`sValue = busComp.GetFieldValue(`*`FieldName`*` as String, ErrCode as Integer)`

GetFormattedFieldValue	Returns a formatted value for the field specified in the argument.	`Dim busComp as SiebelBusComp`
		`Dim sValue as String`
		`sValue = busComp.GetFormattedFieldValue(`*`FieldName`*` as String, ErrCode as Integer)`

GetMultipleFieldValues	Returns a value for the fields specified in the property set.	`Dim buscomp as SiebelBusComp` `buscomp.GetMultipleFieldValues(oPropSet as SiebelPropertySet, ErrCode as Integer)`

GetMVGBusComp	Returns the MVG business component associated with the field specified in the argument.	`Dim busComp as SiebelBusComp`
		`Dim mVGBusComp as SiebelBusComp`
		`set mVGBusComp = busComp.GetMVGBusComp(`*`FieldName`*` as String, ErrCode as Integer)`

GetNamedSearch	Returns the argument-named search specification.	`Dim busComp as SiebelBusComp`
		`Dim sValue as String`
		`sValue = busComp.GetNamedSearch(`*`SearchName`*` as String, ErrCode as Integer)`

GetPicklistBusComp	Returns the pick business component associated with the field specified in the argument.	`Dim busComp as SiebelBusComp`
		`Dim pickBusComp as SiebelBusComp`
		`Set pickBusComp = busComp.GetPicklistBusComp(`*`FieldName`*` as String, ErrCode as Integer)`

GetSearchExpr	Returns the current search expression.	`Dim busComp as SiebelBusComp`
		`Dim sExpr as String`
		`sExpr = busComp.GetSearchExpr(ErrCode as Integer)`

GetSearchSpec	Returns the current search specification for the field specified in the argument.	`Dim busComp as BusComp`
		`Dim sSpec as String`
		`sSpec = busComp.GetSearchSpec(`*`FieldName`*` as String, ErrCode as Integer)`

GetUserProperty	Returns the value for the property name whose name is specified in the argument.	`Dim busComp as SiebelBusComp`
		`Dim sValue as String`
		`sValue = busComp.GetUserProperty(`*`propertyName`*` as String, ErrCode as Integer)`

Table 31. Business Component Methods Syntax Summary

Method	Description	Syntax

GetViewMode	Returns the visibility mode for the business component.	`Dim busComp as SiebelBusComp`
		`Dim iMode as Integer`
		`iMode = busComp.GetViewMode(ErrCode as Integer)`

LastRecord	Moves to the last record in the business component.	`Dim busComp as SiebelBusComp`
		`Dim bReturn as Boolean`
		`bReturn = busComp.LastRecord(ErrCode as Integer)`

Name	Returns the name of the business component.	`Dim busComp as SiebelBusComp`
		`Dim sName as String`
		`sName = busComp.Name(ErrCode as Integer)`

NewRecord	Adds a new record to the business component.	`Dim busComp as SiebelBusComp` `busComp.NewRecord(`*`whereIndicator`*` as Integer, ErrCode as Integer)`

NextRecord	Moves to the next record in the business component.	`Dim busComp as SiebelBusComp`
		`Dim bReturn as Boolean`
		`bReturn = busComp.NextRecord(ErrCode as Integer)`

ParentBusComp	Returns the parent business component.	`Dim busComp as SiebelBusComp`
		`Dim parentBusComp as SiebelBusComp`
		`Set parentBusComp = busComp.ParentBusComp(ErrCode as Integer)`

Pick	Places the currently selected record in a picklist business component into the appropriate fields of the parent business component.	`Dim busComp as SiebelBusComp` `busComp.Pick(ErrCode as Integer)`

PreviousRecord	Moves to the previous record in the business component.	`Dim busComp as SiebelBusComp`
		`Dim bReturn as Boolean`
		`bReturn = busComp.PreviousRecord(ErrCode as Integer)`

RefineQuery	Refines a query after a query has been executed.	`Dim busComp as SiebelBusComp` `busComp.RefineQuery(ErrCode as Integer)`

SetFieldValue	Assigns a new value to the named field for the current row of the business component.	`Dim busComp as SiebelBusComp`
		
		`SetFieldValue(fieldname As String, fieldValue As string,errCode as Integer)`

Table 31. Business Component Methods Syntax Summary

Method	Description	Syntax

SetFormattedFieldValue	Accepts the field value in the current local format and assigns the new value to the named field for the current row of the business component.	`Dim busComp as SiebelBusComp`
		`busComp.SetFormattedFieldValue(`*`FieldName`* `as String,`
		`FieldValue` `as String, ErrCode as Integer)`

SetMultipleFieldValues	Assigns a new value to the fields specified in the property set for the current row of the business component.	`Dim buscomp as SiebelBusComp` `buscomp.SetMultipleFieldValues(oPropSet as SiebelPropertySet, ErrCode as Integer)`
SetNamedSearch	Sets a named search specification on the business component.	`Dim busComp as SiebelBusComp` `busComp.SetNamedSearch(`*`searchName`* `as String, searchSpec as String, ErrCode as Integer)`
SetSearchExpr	Sets the search specification for the business component.	`Dim busComp as SiebelBusComp` `busComp.SetSearchExpr(`*`searchSpec`* `as String, ErrCode as Integer)`

SetSearchSpec	Sets the search specification for the specified field.	`Dim busComp as SiebelBusComp`
		`busComp.SetSearchSpec(`*`FieldName`* `as String,`
		`searchSpec` `as String, ErrCode as Integer)`

SetSortSpec	Sets the sort specification for a query.	`Dim busComp as SiebelBusComp` `busComp.SetSortSpec(`*`sortSpec`* `as String, ErrCode as Integer)`

SetUserProperty	Sets the value of the specified User Property.	`Dim busComp as SiebelBusComp`
		`busComp.SetUserProperty(`*`propertyName`* `as String,`
		`newValue` `as String, ErrCode as Integer)`

SetViewMode	Sets the visibility type for the business component.	`SetViewMode(mode As Integer, errCode As Integer)`
UndoRecord	Deletes an active record created by NewRecord.	`Dim busComp as SiebelBusComp` `busComp.UndoRecord(ErrCode as Integer)`
WriteRecord	Commits to the database any changes made to the current record	`Dim busComp as SiebelBusComp` `busComp.WriteRecord(ErrCode as Integer)`

Business Object

Table 32 lists a summary of the Business Object Methods syntax.

Table 32. Business Object Methods Syntax Summary

Method	Description	Syntax

GetBusComp	Returns the specified business component.	`Dim busObject as SiebelBusObject`
		`Dim busComp as SiebelBusComp`
		`set busComp = busObject.GetBusComp(`*`BusCompName`* `as String, ErrCode as Integer)`

Name	Returns the name of the control.	`Dim busObject as SiebelBusObject`
		`Dim sName as String`
		`sName = busObject.Name(ErrCode as Integer)`

Business Service

Table 33 lists a summary of the Business Service Methods syntax.

Table 33. Business Service Methods Syntax Summary

Method	Description	Syntax

GetFirstProperty	Retrieves the name of the first property of a business service.	`Dim oService as SiebelService`
		`Dim sName as String`
		`sName = oService.GetFirstProperty(ErrCode as Integer)`

GetNextProperty	Once the name of the first property has been retrieved, retrieves the name of the next property of a business service.	`Dim oService as SiebelService`
		`Dim sName as String`
		`sName = oService.GetNextProperty(ErrCode as Integer)`

GetProperty	Retrieves the value stored in the specified property.	`Dim oService as SiebelService`
		`Dim sValue as String`
		`sValue = oService.GetProperty(`*`propName`* `as String, ErrCode as Integer)`

Name	Returns the name of the business service.	`Dim oService as SiebelService`
		`Dim sName as String`
		`sName = oService.Name`

Table 33. Business Service Methods Syntax Summary

Method	Description	Syntax

InvokeMethod	Calls a specialized method or a user-created method on the business service.	`Dim oService as SiebelService`
		`Dim Return`
		`Return = oService.InvokeMethod(methodName as String, InputArguments as SiebelPropertySet, OutputArguments as SiebelPropertySet, ErrCode as Integer)`

PropertyExists	Returns a Boolean value indicating whether the property specified in the argument exists.	`Dim oService as SiebelService` `oService.PropertyExists(`*`propName`*` as String, ErrCode as Integer)`
RemoveProperty	Removes a property from a business service.	`Dim oService as SiebelService` `oService.RemoveProperty(`*`propName`*` as String, ErrCode as Integer)`
SetProperty	Assigns a value to a property of a business service.	`Dim oService as SiebelService` `oService.SetProperty(`*`propName`*` as String, `*`propValue`*` as String, ErrCode as Integer)`

Property Set

Table 34 lists a summary of the Property Set Methods syntax.

Table 34. Property Set Methods Syntax Summary

Method	Description	Syntax

AddChild	Adds subsidiary property sets to a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim iIndex as Integer`
		`OPropSet.AddChild(childObject as SiebelPropertySet, errCode as Integer)`

Copy	Returns a copy of a property set.	`Dim oPropSet1 as SiebelPropertySet`
		`Dim oPropSet2 as SiebelPropertySet`
		`oPropSet2 = oPropSet1.Copy(ErrCode as Integer)`

GetChild	Returns a specified child property set of a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim sPropVal as String`
		`sPropVal = oPropSet.GetChild(`*`index`*` as Integer, ErrCode as Integer)`

GetChildCount	Returns the number of child property sets attached to a parent property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim iCount as Integer`
		`iCount = oPropSet.GetChildCount(ErrCode as Integer)`

Table 34. Property Set Methods Syntax Summary

Method	Description	Syntax

GetFirstProperty	Returns the name of the first property in a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim sPropName as String`
		`sPropName = oPropSet.GetFirstProperty(ErrCode as Integer)`

GetNextProperty	Returns the name of the next property in a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim sPropName as String`
		`sPropName = oPropSet.GetNextProperty(ErrCode as Integer)`

GetProperty	Returns the value of a property when given the property name.	`Dim oPropSet as SiebelPropertySet`
		`Dim sPropVal as String`
		`sPropVal = oPropSet.GetProperty(`*`propName`*` as String, ErrCode as Integer)`

GetPropertyCount	Returns the number of properties contained within the property set.	`Dim oPropSet as SiebelPropertySet` `oPropSet.GetPropertyCount (ErrCode as Integer)`

GetType	Returns the value stored in a type in a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim sTypeVal as String`
		`sTypeVal = oPropSet.GetType(`*`value`*` as String)`

GetValue	Returns a value stored as part of a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim sValVal as String`
		`sValVal = oPropSet.GetValue(`*`value`*` as String, ErrCode as Integer)`

InsertChildAt	Inserts a child property set into a parent property set at a specific location.	`Dim oPropSet as SiebelPropertySet` `oPropSet.InsertChildAt(`*`childObject`*` as String, `*`index`*` as Integer, ErrCode as Integer)`
PropertyExists	Returns a Boolean value indicating whether the property specified in the argument exists.	`Dim oPropSet as Property Set` `oPropSet.PropertyExists(`*`propName`*` as String, ErrCode as Integer)`
RemoveChild	Removes a child property set as a specified index from a parent property set.	`Dim oPropSet as SiebelPropertySet` `oPropSet.RemoveChild `*`index`*` as Integer`
RemoveProperty	Removes the property specified in its argument from a property set.	`Dim oPropSet as SiebelPropertySet` `oPropSet.RemoveProperty(`*`propName`*` as String, ErrCode as Integer)`
Reset	Removes every property and child property set from a property set.	`Dim oPropSet as SiebelPropertySet` `oPropSet.Reset(ErrCode as Integer)`

Table 34. Property Set Methods Syntax Summary

Method	Description	Syntax
SetProperty	Assigns a value to the property of a property set specified in its argument.	`Dim oPropSet as SiebelPropertySet` `oPropSet.SetProperty(`*`propName`* `as String,` *`propValue`* `as String, ErrCode as Integer)`
SetType	Assigns a data value to a type member of a property set.	`Dim oPropSet as SiebelPropertySet` `oPropSet.SetType(`*`value`* `as String, ErrCode as Integer)`
SetValue	Assigns a data value to a value member of a property set.	`Dim oPropSet as SiebelPropertySet` `oPropSet.SetValue(`*`value`* `as String)`

Mobile/Dedicated Web Client Automation Server Quick Reference 7

This quick reference has the following topics:

- "Application"
- "Business Component" on page 372
- "Business Object" on page 376
- "Business Service" on page 377
- "Property Set" on page 378

Application

Table 35 lists a summary of the Application Methods syntax.

Table 35. Application Methods Syntax Summary

Method	Description	Syntax

ActiveBusObject	Returns the business object for the business component for the active applet.	`Dim application as SiebelWebApplication`
		`Dim busObject as SiebelBusObject`
		`Set busObject = application.ActiveBusObject`

ActiveViewName	Returns the name of the active view.	`Dim application as SiebelWebApplication`
		`Dim sView as String`
		`sView = application.ActiveViewName`

CurrencyCode	Returns the three-letter operating currency code.	`Dim application as SiebelWebApplication`
		`Dim sCur as String`
		`sCur = Application.CurrencyCode`

EnableExceptions	Enables or disables native COM error handling.	`Dim application as SiebelWebApplication`
		`Dim bEnable as Integer`
		`Call application.EnableExceptions(bEnable as Integer)`

Table 35. Application Methods Syntax Summary

Method	Description	Syntax

GetBusObject	Instantiates and returns a new instance of the business object specified in the argument.	`Dim application as SiebelWebApplication`
		`Dim busObject as SiebelBusObject`
		`set busObject = application.GetBusObject(`*`busobjName`* `as String)`

GetLastErrCode	Gets the shared user-defined global variables.	`Dim application as SiebelWebApplication`
		`Dim sName as String`
		`sName = application.GetSharedGlobal(`*`varName`* `as String)`

GetLastErrText	Returns the last error text message.	`Dim application as SiebelWebApplication`
		`Dim sText as String`
		`sText = application.GetLastErrText`

GetProfileAttr	Returns the value of an attribute in a user profile.	`Dim application as SiebelWebApplication`
		`Dim profName as String`
		`Dim profValue as String`
		`profValue = application.GetProfileAttr(profName as String)`

GetService	Instantiates and returns a new instance of the argument-specified service.	`Dim application as SiebelWebApplication`
		`Dim oService as SiebelService`
		`set oService = Application.GetService(`*`serviceName`* `as String)`

GetSharedGlobal	Returns the shared user-defined global variables.	`Dim application as SiebelWebApplication`
		`Dim name as String`
		`name = application.GetSharedGlobal (sName as String)`

InvokeMethod	Calls the named specialized method.	`Dim application as SiebelWebApplication`
		`Dim sReturn as String`
		`sReturn = application.InvokeMethod(methodName as String,`
		`methodArgs` `as String or StringArray)`

LoginId	Returns the login ID of the user who started the Siebel application.	`Dim application as SiebelWebApplication`
		`Dim sID as string`
		`sID = application.LoginId`

LoginName	Returns the login name of the user who started the Siebel application.	`Dim application as SiebelWebApplication`
		`Dim sUser as String`
		`sUser = application.LoginName`

Logoff	Terminates the Mobile Web Client session.	`Dim application as SiebelWebApplication`
		`Dim status as Boolean`
		`Status = application.Logoff`

Table 35. Application Methods Syntax Summary

Method	Description	Syntax

NewPropertySet	Constructs a new property set object.	`Dim application as SiebelWebApplication`
		`Dim propset As SiebelPropertySet`
		`propset = application.NewPropertySet`

PositionId	Returns the position ID that describes the user's current position.	`Dim application as SiebelWebApplication`
		`Dim sRow as String`
		`sRow = application.PositionId`

PositionName	Returns the position name of the user's current position.	`Dim application as SiebelWebApplication`
		`Dim sPosition as String`
		`sPosition = application.PositionName`

SetPositionId	Sets the active position to the Position ID specified in the argument.	`Dim application as SiebelWebApplication`
		`Dim posId as String`
		`Dim status as Boolean`
		`status = application.SetPositionId(posId)`

SetPositionName	Sets the active position to the position name specified in the argument.	`Dim application as SiebelWebApplication`
		`Dim posName as String`
		`Dim status as Boolean`
		`status = application.SetPositionName(posName)`

SetProfileAttr	Used in personalization to assign values to attributes in a user profile.	`Dim oApplication as SiebelWebApplication`
		`Dim bool as Boolean`
		`bool = oApplication.SetProfileAttr(`*`name`*` as String, `*`value`*` as String)`

SetSharedGlobal	Sets a shared user-defined global variable.	`Dim application as SiebelWebApplication`
		`Dim bool as Boolean`
		`bool = application.SetSharedGlobal(varName as String, `*`value`*` as String)`

Trace	Appends a message to the trace file.	`Dim application as SiebelWebApplication` `application.Trace(message as String)`

TraceOff	Turns off the tracing started by TraceOn.	`Dim application as SiebelWebApplication`
		`Dim bool as Boolean`
		`bool = application.TraceOff`

TraceOn	Turns tracing on.	`Dim application as SiebelWebApplication`
		`Dim bool as Boolean`
		`bool = application.TraceOn(`*`filename`*` as String, `*`type`*` as Integer, Selection as String)`

Business Component

Table 36 lists a summary of the Business Component Methods syntax.

Table 36. Business Component Methods Syntax Summary

Method	Description	Syntax

ActivateField	Allows queries to retrieve data for the specified field.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = BusComp.ActivateField(`*`fieldName`*` as String)`

ActivateMultipleFields	Allows queries to retrieve data for the fields specified in the property set.	`Dim buscomp as SiebelBusComp` `buscomp.ActivateMultipleFields(oPropSet as SiebelPropertySet)`

Associate	Creates a new many-to-many relationship for the parent object through an association business component.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.Associate(`*`whereIndicator`*` as Integer)`

BusObject	Returns the business object that contains the business component.	`Dim busComp as SiebelBusComp`
		`Dim busObject as SiebelBusObject`
		`Set BusObject = busComp.BusObject`

ClearToQuery	Clears the current query and sort specifications on the business component.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.ClearToQuery`

DeactivateFields	Deactivates every currently activated field.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.DeactivateFields`

DeleteRecord	Removes the current record from the business component.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.DeleteRecord`

ExecuteQuery	Retrieves a set of BusComp records.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.ExecuteQuery(`*`cursorMode`*` as Integer)`

ExecuteQuery2	Retrieves a set of BusComp records.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.ExecuteQuery2(`*`cursorMode`*` as Integer,`
		`ignoreMaxCursorSize`` as Integer)`

FirstRecord	Moves to the first record in the business component.	`Dim busComp as SiebelBusComp`
		`Dim bIsRecord as Boolean`
		`bIsRecord = busComp.FirstRecord`

Table 36. Business Component Methods Syntax Summary

Method	Description	Syntax

GetAssocBusComp	Returns the association business component.	`Dim busComp as SiebelBusComp`
		`Dim AssocBusComp as SiebelBusComp`
		`Set AssocBusComp = busComp.GetAssocBusComp`

GetFieldValue	Returns a value for the field specified in the argument.	`Dim busComp as SiebelBusComp`
		`Dim sValue as String`
		`sValue = busComp.GetFieldValue(`*`FieldName`*` as String)`

GetFormattedFieldValue	Returns a formatted value for the field specified in the argument.	`Dim busComp as SiebelBusComp`
		`Dim sValue as String`
		`sValue = busComp.GetFormattedFieldValue(`*`FieldName`*` as String)`

GetLastErrCode	Returns the last Siebel error number.	`Dim buscomp as SiebelBusComp`
		`Dim bool as Integer`
		`bool = buscomp.GetLastErrCode`

GetLastErrText	Returns the last error text message.	`Dim busComp as SiebelBusComp`
		`Dim sErr as String`
		`sErr = busComp.GetLastErrText`

GetMultipleFieldValues	Returns a value for the fields specified in the property set.	`Dim buscomp as SiebelBusComp` `buscomp.GetMultipleFieldValues(oPropSet as SiebelPropertySet, PValues as SiebelPropertySet)`

GetMVGBusComp	Returns the MVG business component associated with the field specified in the argument.	`Dim busComp as SiebelBusComp`
		`Dim mVGBusComp as SiebelBusComp`
		`set mVGBusComp = busComp.GetMVGBusComp(`*`FieldName`*` as String)`

GetNamedSearch	Returns the argument-named search specification.	`Dim busComp as SiebelBusComp`
		`Dim sValue as String`
		`sValue = busComp.GetNamedSearch(`*`SearchName`*` as String)`

GetPicklistBusComp	Returns the pick business component associated with the field specified in the argument.	`Dim busComp as SiebelBusComp`
		`Dim pickBusComp as SiebelBusComp`
		`Set pickBusComp = busComp.GetPicklistBusComp(`*`FieldName`*` as String)`

GetSearchExpr	Returns the current search expression.	`Dim busComp as SiebelBusComp`
		`Dim sExpr as String`
		`sExpr = busComp.GetSearchExpr`

GetSearchSpec	Returns the current search specification for the field specified in the argument.	`Dim busComp as SiebelBusComp`
		`Dim sSpec as String`
		`sSpec = busComp.GetSearchSpec(`*`FieldName`*` as String)`

Table 36. Business Component Methods Syntax Summary

Method	Description	Syntax

GetUserProperty	Returns the value for the property name specified in the argument.	`Dim busComp as SiebelBusComp`
		`Dim sValue as String`
		`sValue = busComp.GetUserProperty(`*`propertyName`* `as String)`

GetViewMode	Returns the visibility mode for the business component.	`Dim busComp as SiebelBusComp`
		`Dim iMode as Integer`
		`iMode = busComp.GetViewMode`

InvokeMethod	Calls the specialized method named in the argument.	`Dim busComp as SiebelBusComp`
		`Dim sReturn as String`
		`sReturn = busComp.InvokeMethod(`*`methodName`* `as String or StringArray,` *`methodArgs`* `as String)`

LastRecord	Moves to the last record in the business component.	`Dim busComp as SiebelBusComp`
		`Dim bReturn as Boolean`
		`bReturn = busComp.LastRecord`

Name	Returns the name of the business component.	`Dim busComp as SiebelBusComp`
		`Dim sName as String`
		`sName = busComp.Name`

NewRecord	Adds a new record to the business component.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.NewRecord(`*`whereIndicator`* `as Integer)`

NextRecord	Moves to the next record in the business component.	`Dim busComp as SiebelBusComp`
		`Dim bReturn as Boolean`
		`bReturn = busComp.NextRecord`

ParentBusComp	Returns the parent business component.	`Dim busComp as SiebelBusComp`
		`Dim parentBusComp as SiebelBusComp`
		`Set parentBusComp = busComp.ParentBusComp`

Pick	Places the currently selected record in a picklist business component into the appropriate fields of the parent business component.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.Pick`

PreviousRecord	Moves to the previous record in the business component.	`Dim busComp as SiebelBusComp`
		`Dim bReturn as Boolean`
		`bReturn = busComp.PreviousRecord`

RefineQuery	Refines a query after a query has been executed.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.RefineQuery`

Table 36. Business Component Methods Syntax Summary

Method	Description	Syntax

SetFieldValue	Assigns a new value to the named field for the current row of the business component.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.SetFieldValue(`*`FieldName`* `as String,`
		`FieldValue` `as String)`

SetFormattedFieldValue	Accepts the field value in the current local format and assigns the new value to the named field for the current row of the business component.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.SetFormattedFieldValue(`*`FieldName`* `as String,` *`FieldValue`* `as String)`

SetMultipleFieldValues	Assigns a new value to the fields specified in the property set for the current row of the business component.	`Dim buscomp as SiebelBusComp` `buscomp.SetMultipleFieldValues(oPropSet as SiebelPropertySet)`

SetNamedSearch	Sets a named search specification on the business component.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.SetNamedSearch(`*`searchName`* `as String,`
		`searchSpec` `as String)`

SetSearchExpr	Sets the search specification for the business component.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.SetSearchExpr(`*`searchSpec`* `as String)`

SetSearchSpec	Sets the search specification for the specified field.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.SetSearchSpec(`*`FieldName`* `as String,`
		`searchSpec` `as String)`

SetSortSpec	Sets the sort specification for a query.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.SetSortSpec(`*`sortSpec`* `as String)`

SetUserProperty	Sets the value of the specified User Property.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.SetUserProperty(`*`propertyName`* `as String,` *`newValue`* `as String)`

SetViewMode	Sets the visibility type for the business component.	`Dim buscomp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = buscomp.SetViewMode(mode As Integer)`

Table 36. Business Component Methods Syntax Summary

Method	Description	Syntax

UndoRecord	Deletes an active record created by NewRecord.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.UndoRecord`

WriteRecord	Commits to the database any changes made to the current record.	`Dim busComp as SiebelBusComp`
		`Dim bool as Boolean`
		`bool = busComp.WriteRecord`

Business Object

Table 37 lists a summary of the Business Object Methods syntax.

Table 37. Business Object Methods Syntax Summary

Method	Description	Syntax

GetBusComp	Returns the specified business component.	`Dim busObject as SiebelBusObject`
		`Dim busComp as SiebelBusComp`
		`set busComp = busObject.GetBusComp(`*`BusCompName`*` as String)`

GetLastErrCode	Returns the last Siebel error number.	`Dim busobject as SiebelBusObject`
		`Dim bool as Integer`
		`bool = busobject.GetLastErrCode`

GetLastErrText	Returns the last error text message.	`Dim busobject as SiebelBusObject`
		`Dim bool as String`
		`bool = busobject.GetLastErrText`

Name	Returns the name of the business object.	`Dim busObject as SiebelBusObject`
		`Dim sName as String`
		`sName = busObject.Name`

Business Service

Table 38 lists a summary of the Business Service Methods syntax.

Table 38. Business Service Methods Syntax Summary

Method	Description	Syntax

GetFirstProperty	Retrieves the name of the first property of a business service.	`Dim oService as SiebelService`
		`Dim sName as String`
		`sName = oService.GetFirstProperty`

GetNextProperty	Once the name of the first property has been retrieved, retrieves the name of the next property of a business service.	`Dim oService as SiebelService`
		`Dim sName as String`
		`sName = oService.GetNextProperty`

GetProperty	Retrieves the value stored in the specified property.	`Dim oService as SiebelService`
		`Dim sValue as String`
		`sValue = oService.GetProperty(`*`propName`*` as String)`

InvokeMethod	Calls a specialized method or a user-created method on the business service.	`Dim oService as SiebelService`
		`Dim Return as Boolean`
		`Return = oService.InvokeMethod(methodName as String, InputArguments as SiebelPropertySet, OutputArguments as SiebelPropertySet)`

Name	Returns the name of the business service.	`Dim oService as SiebelService`
		`Dim sName as String`
		`sName = oService.Name`

PropertyExists	Returns a Boolean value indicating whether the property specified in the argument exists.	`Dim oService as SiebelService`
		`Dim bool as Boolean`
		`bool = oService.PropertyExists(`*`propName`*` as String)`

RemoveProperty	Removes a property from a business service.	`Dim oService as SiebelService`
		`Dim bool as Boolean`
		`bool = oService.RemoveProperty(`*`propName`*` as String)`

SetProperty	Assigns a value to a property of a business service.	`Dim oService as SiebelService` `oService.SetProperty(`*`propName`*` as String, `*`propValue`*` as String)`

Property Set

Table 39 lists a summary of the Property Set Methods syntax.

Table 39. Property Set Methods Syntax Summary

Method	Description	Syntax
AddChild	Adds subsidiary property sets to a property set.	`Dim oPropSet as SiebelPropertyset` `oPropSet.AddChild(`*`childObject`*` as SiebelPropertySet)`

Copy	Returns a copy of a property set.	`Dim oPropSet1 as SiebelPropertyset`
		`Dim oPropSet2 as SiebelPropertyset`
		`oPropSet2 = oPropSet1.Copy`

GetChild	Returns a specified child property set of a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim childPropSet as SiebelPropertySet`
		`childPropSet = oPropSet.GetChild(index as Long)`

GetChildCount	Returns the number of child property sets attached to a parent property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim iCount as Long`
		`iCount = oPropSet.GetChildCount`

GetFirstProperty	Returns the name of the first property in a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim sPropName as String`
		`sPropName = oPropSet.GetFirstProperty`

GetLastErrCode	Returns the last Siebel error number.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as Integer`
		`bool = oPropSet.GetLastErrCode`

GetLastErrText	Returns the last error text message.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as String`
		`bool = oPropSet.GetLastErrText`

GetNextProperty	Returns the name of the next property in a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim sPropName as String`
		`sPropName = oPropSet.GetNextProperty`

GetProperty	Returns the value of a property when given the property name.	`Dim oPropSet as SiebelPropertySet`
		`Dim sPropVal as String`
		`sPropVal = oPropSet.GetProperty(`*`propName`*` as String)`

GetPropertyCount	Returns the number of properties contained within the property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as Long`
		`bool = oPropSet.GetPropertyCount`

Table 39. Property Set Methods Syntax Summary

Method	Description	Syntax

GetType	Returns the value stored in a type in a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim sTypeVal as String`
		`sTypeVal = oPropSet.GetType(value as String)`

GetValue	Returns a value stored as part of a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim sValVal as String`
		`sValVal = oPropSet.GetValue(value as String)`

InsertChildAt	Inserts a child property set into a parent property set at a specific location.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as Boolean`
		`bool = oPropSet.InsertChildAt(childObject as SiebelPropertySet, index as Long)`

PropertyExists	Returns a Boolean value indicating whether the property specified in the argument exists.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as Boolean`
		`bool = oPropSet.PropertyExists(propName as String)`

RemoveChild	Removes a child property set as a specified index from a parent property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as Boolean`
		`bool = oPropSet.RemoveChild(index as Long)`

RemoveProperty	Removes the property specified in its argument from a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as Boolean`
		`bool = oPropSet.RemoveProperty(propName as String)`

Reset	Removes every property and child property set from a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as Boolean`
		`bool = oPropSet.Reset`

SetProperty	Assigns a value to the property of a property set specified in its argument.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as Boolean`
		`bool = oPropSet.SetProperty(propName as String, propValue as String)`

SetType	Assigns a data value to a type member of a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as Boolean`
		`bool = oPropSet.SetType(value as String)`

SetValue	Assigns a data value to a value member of a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as Boolean`
		`bool = oPropSet.SetValue(value as String)`

Siebel Web Client Automation Server Quick Reference

8

This quick reference has the following topics:

- "SiebelHTMLApplication"
- "SiebelService" on page 382
- "PropertySet" on page 382

SiebelHTMLApplication

Table 40 lists a summary of the Siebel HTMLApplication Methods syntax.

Table 40. SiebelHTMLApplication Methods Syntax Summary

Method	Description	Syntax

GetLastErrCode	Returns the last error code.	`Dim siebelApp As SiebelHTMLApplication`
		`Dim iErr as Long`
		`iErr = siebelApp.GetLastErrCode`

GetLastErrText	Returns the last error text message.	`Dim siebelApp As SiebelHTMLApplication`
		`Dim sText as String`
		`sText = siebelApp.GetLastErrText`

GetService	Instantiates and returns a new instance of the service specified in the argument.	`Dim siebelApp As SiebelHTMLApplication`
		`Dim svc As SiebelService`
		`Set svc = siebelApp.GetService(`*`ServiceName`* `as String)`

Name	Returns the name of the current application as defined in the repository.	`Dim siebelApp As SiebelHTMLApplication`
		`Dim name as String`
		`name = siebelApp.Name`

NewPropertySet	Constructs and returns a new property set object.	`Dim siebelApp As SiebelHTMLApplication`
		`Dim propSet as SiebelPropertySet`
		`Set propSet = siebelApp.NewPropertySet`

SiebelService

Table 41 lists a summary of the SiebelService Methods syntax.

Table 41. SiebelService Methods Syntax Summary

Method	Description	Syntax

GetLastErrCode	Returns the last error code.	`Dim svc As SiebelServic`
		`Dim iErr as Long`
		`iErr = svc.GetLastErrCode`

GetLastErrText	Returns the last error text message.	`Dim svc As SiebelService`
		`Dim sText as String`
		`sText = svc.GetLastErrText`

InvokeMethod	Calls a specialized method or a user-created method on the business service.	`Dim svc As SiebelService`
		`Dim status as Boolean`
		`Status = svc.InvokeMethod(`*`MethodName as String, inputPropSet as SiebelPropertySet, outputPropSet as SiebelPropertySet`*`)`

Name	Returns the name of the business service.	`Dim svc As SiebelService`
		`Dim name as String`
		`name = svc.Name`

PropertySet

Table 42 lists a summary of the PropertySet Methods syntax.

Table 42. PropertySet Methods Syntax Summary

Method	Description	Syntax

AddChild	Adds subsidiary property sets to a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as Long`
		`bool = oPropSet.AddChild(`*`childObject`* `as SiebelPropertySet)`

Copy	Returns a copy of a property set.	`Dim oPropSet1 as SiebelPropertySet`
		`Dim oPropSet2 as SiebelPropertySet`
		`Set oPropSet2 = oPropSet1.Copy`

GetChild	Returns a specified child property set of a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim oChildPropSet as SiebelPropertySet`
		`Set oChildPropSet = oPropSet.GetChild(`*`index`* `as Long)`

Table 42. PropertySet Methods Syntax Summary

Method	Description	Syntax

GetChildCount	Returns the number of child property sets attached to a parent property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim iCount as Long`
		`iCount = oPropSet.GetChildCount`

GetFirstProperty	Returns the name of the first property in a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim sPropName as String`
		`sPropName = oPropSet.GetFirstProperty`

GetLastErrCode	Returns the last error code.	`Dim oPropSet as SiebelPropertySet`
		`Dim iErr as Long`
		`iErr = oPropSet.GetLastErrCode`

GetLastErrText	Returns the last error text message.	`Dim oPropSet as SiebelPropertySet`
		`Dim sText as String`
		`sText = oPropSet.GetLastErrText`

GetNextProperty	Returns the name of the next property in a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim sPropName as String`
		`sPropName = oPropSet.GetNextProperty`

GetProperty	Returns the value of a property when given the property name.	`Dim oPropSet as SiebelPropertySet`
		`Dim sValue as String`
		`sValue = oPropSet.GetProperty(`*`propName`* `as String)`

GetPropertyCount	Returns the number of properties attached to a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim iCount as Long`
		`iCount = oPropSet.GetPropertyCount`

GetType	Returns the value stored in a type in a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim type as String`
		`type = oPropSet.GetType`

GetValue	Returns a value stored as part of a property set.	`Dim oPropSet as SiebelProperty`
		`SetDim sValue as String`
		`sValue = oPropSet.GetValue`

InsertChildAt	Inserts a child property set into a parent property set at a specific location.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as Boolean`
		`bool = oPropSet.InsertChildAt(`*`childObject`* `as SiebelPropertySet,` *`index`* `as Long)`

PropertyExists	Returns a Boolean value indicating whether the property specified in the argument exists.	`Dim oPropSet as SiebelProperty`
		`Dim bool as Boolean`
		`bool = oPropSet.PropertyExists(`*`propName`* `as String)`

RemoveChild	Removes a child property set as a specified index from a parent property set.	`Dim oPropSet as SiebelPropertySet` `oPropSet.RemoveChild(`*`index`* `as Long)`

Table 42. PropertySet Methods Syntax Summary

Method	Description	Syntax

RemoveProperty	Removes the property specified in its argument from a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as Boolean`
		`bool = oPropSet.RemoveProperty(`*`propName`* `as String)`

Reset	Removes every property and child property set from a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as Boolean`
		`bool = oPropSet.Reset`

SetProperty	Assigns a value to the property of a property set specified in its argument.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as Boolean`
		`bool = oPropSet.SetProperty(`*`propName`* `as String,` *`propValue`* `as String)`

SetType	Assigns a data value to a type member of a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as Boolean`
		`bool = oPropSet.SetType(`*`value`* `as String)`

SetValue	Assigns a data value to a value member of a property set.	`Dim oPropSet as SiebelPropertySet`
		`Dim bool as Boolean`
		`bool = oPropSet.SetValue(`*`value`* `as String)`

Java Data Bean Quick Reference 9

This quick reference has the following topics:

- "Data Bean"
- "Business Component" on page 387
- "Business Object" on page 391
- "Business Service" on page 391
- "Property Set" on page 392
- "SiebelException" on page 394

Data Bean

Table 43 lists a summary of the SiebelDataBean Methods syntax.

Table 43. SiebelDataBean Methods Syntax Summary

Method	Description	Syntax
Attach	Allows an external application to reconnect to an existing Siebel session.	`boolean attach(String sessionID) throws SiebelException`
CurrencyCode	Returns the three-letter operating currency code.	`String currencyCode ()`
Detach	Returns a string containing the Siebel session ID.	`String detach() throws SiebelException`
GetBusObject	Instantiates and returns a new instance of the business object specified in the argument.	`SiebelBusObject getBusObject (String boName) throws SiebelException`

Table 43. SiebelDataBean Methods Syntax Summary

Method	Description	Syntax
GetProfileAttr	Returns the value of an attribute in a user profile.	`String getProfileAttr(String attrName) throws SiebelException`
GetService	Returns a specified service. If the service is not already running, it is constructed.	`SiebelService getService (string serviceName) throws SiebelException`
InvokeMethod	Calls the named specialized method.	`String invokeMethod (String name, String[] args) throws SiebelException`
Login	Allows external applications to log in to the Data Bean.	`boolean login (String connString, String userName, String passWord) throws SiebelException`
LoginId	Returns the login ID of the user who started the Siebel application.	`String loginId ()`
LoginName	Returns the login name of the user who started the Siebel application.	`String loginName ()`
Logoff	Disconnects the client from the server.	`boolean logoff() throws SiebelException`
NewPropertySet	Constructs and returns a new property set object.	`SiebelPropertySet newPropertySet()`
PositionId	Returns the position ID that describes the user's current position.	`String positionId ()`
PositionName	Returns the position name of the user's current position.	`String positionName ()`
SetPositionId	Sets the active position to the Position ID specified in the argument.	`boolean setPositionId(String posId) throws SiebelException`
SetPositionName	Sets the active position to the position name specified in the argument. Returns a Boolean value indicating if the method succeeded.	`boolean setPositionName(String posName) throws SiebelException`
SetProfileAttr	Personalization uses to assign values to attributes in a user profile.	`boolean setProfileAttr(String attrName,.String attrValue) throws SiebelException`

Table 43. SiebelDataBean Methods Syntax Summary

Method	Description	Syntax
Trace	The Trace method appends a message to the trace file. Trace is useful for debugging SQL query execution.	`boolean trace(String message) throws SiebelException`
TraceOff	TraceOff turns off the tracing started by the TraceOn method.	`boolean traceOff() throws SiebelException`
TraceOn	TraceOn turns on the tracking of allocations and deallocations of Siebel objects, and SQL statements generated by the Siebel application.	`boolean traceOn(String filename, String Category, String selection)throws SiebelException`

Business Component

Table 44 lists a summary of the Siebel BusComp Methods syntax.

Table 44. SiebelBusComp Methods Syntax Summary

Method	Description	Syntax
ActivateField	Allows queries to retrieve data for the specified field.	`boolean activateField (String fieldName) throws SiebelException`
ActivateMultipleFields	Allows queries to retrieve data for the fields specified in the property set.	`boolean activateMultipleFields(SiebelPropertySet psFields) throws SiebelException`
Associate	Creates a new many-to-many relationship for the parent object through an association business component.	`boolean associate (boolean isInsertBefore) throws SiebelException`
BusObject	Returns the business object that contains the business component.	`SiebelBusObject busObject () throws SiebelException`
ClearToQuery	Clears the current query and sort specifications on the business component.	`boolean clearToQuery () throws SiebelException`

Table 44. SiebelBusComp Methods Syntax Summary

Method	Description	Syntax
DeactivateFields	Deactivates every currently activated field.	`boolean deactivateFields()`
DeleteRecord	Removes the current record from the business component.	`boolean deleteRecord () throws SiebelException`
ExecuteQuery	Retrieves a set of BusComp records.	`boolean executeQuery (boolean cursorMode) throws SiebelException`
ExecuteQuery2	Retrieves a set of BusComp records.	`boolean executeQuery2 (boolean cursorMode, boolean ignoreMaxCursorSize) throws SiebelException`
FirstRecord	Moves to the first record in the business component.	`boolean firstRecord () throws SiebelException`
FirstSelected	Returns the association business component.	`SiebelBusComp getAssocBusComp () throws SiebelException`
GetFieldValue	Returns a value for the field specified in the argument.	`String getFieldValue (String fieldName) throws SiebelException`
GetFormattedFieldValue	Returns a formatted value for the field specified in the argument.	`String getFormattedFieldValue (String fieldName) throws SiebelException`
GetMultipleFieldValues	Returns a value for the fields specified in the property set.	`boolean getMultipleFieldValues(SiebelPropertySet Src, SiebelPropertySet result) throws SiebelException`
GetMVGBusComp	Returns the MVG business component associated with the field specified in the argument.	`SiebelBusComp getMVGBusComp (String fieldName) throws SiebelException`
GetNamedSearch	Returns the argument-named search specification.	`String getNamedSearch (String searchName) throws SiebelException`
GetPicklistBusComp	Returns the pick business component associated with the field specified in the argument.	`SiebelBusComp getPicklistBusComp (String fieldName) throws SiebelException`
GetSearchExpr	Returns the current search expression.	`String getSearchExpr () throws SiebelException`
GetSearchSpec	Returns the current search specification for the field specified in the argument.	`String getSearchSpec (String fieldName) throws SiebelException`

Table 44. SiebelBusComp Methods Syntax Summary

Method	Description	Syntax
GetUserProperty	Returns the value for the property whose name is specified in the argument.	`String getUserProperty (String property) throws SiebelException`
GetViewMode	Returns the visibility mode for the business component.	`int getViewMode ()`
InvokeMethod	Calls the specialized method named in the argument.	`String invokeMethod (String methodName, String[] methodArgs) throws SiebelException`
LastRecord	Moves to the last record in the business component.	`boolean lastRecord () throws SiebelException`
Name	Returns the name of the business component.	`String name ()`
NewRecord	Adds a new record to the business component.	`boolean newRecord (boolean isInsertBefore) throws SiebelException`
NextRecord	Moves to the next record in the business component.	`boolean nextRecord () throws SiebelException`
ParentBusComp	Returns the parent business component.	`SiebelBusComp parentBusComp () throws SiebelException`
Pick	Places the currently selected record in a picklist business component into the appropriate fields of the parent business component.	`boolean pick () throws SiebelException`
PreviousRecord	Moves to the previous record in the business component.	`boolean previousRecord () throws SiebelException`
RefineQuery	Refines a query after a query has been executed.	`boolean refineQuery () throws SiebelException`
Release	Enables the release of the business component and its resources on the Siebel Server.	`void release() {}`
SetFieldValue	Assigns a new value to the named field for the current row of the business component.	`boolean setFieldValue (String fieldName, String fieldValue) throws SiebelException`

Table 44. SiebelBusComp Methods Syntax Summary

Method	Description	Syntax
SetFormattedFieldValue	Accepts the field value in the current local format and assigns the new value to the named field for the current row of the business component.	`boolean setFormattedFieldValue (String fieldName, String fieldValue) throws SiebelException`
SetMultipleFieldValues	Assigns a new value to the fields specified in the property set for the current row of the business component.	`boolean setMultipleFieldValues(SiebelPropertySet psFields) throws SiebelException`
SetNamedSearch	Sets a named search specification on the business component.	`boolean setNamedSearch (String searchName, String searchText) throws SiebelException`
SetSearchExpr	Sets the search specification for the business component.	`boolean setSearchExpr (String searchExpr) throws SiebelException`
SetSearchSpec	Sets the search specification for the specified field.	`boolean setSearchSpec (String fieldName, String searchSpec) throws SiebelException`
SetSortSpec	Sets the sort specification for a query.	`boolean setSortSpec (String sortSpec) throws SiebelException`
SetUserProperty	Sets the value of the specified User Property.	`boolean setUserProperty(.String propName, String propVal)`
SetViewMode	Sets the visibility type for the business component.	`boolean setViewMode (int mode) throws SiebelException`
UndoRecord	Deletes an active record created by NewRecord.	`boolean undoRecord () throws SiebelException`
WriteRecord	Commits to the database any changes made to the current record.	`boolean writeRecord () throws SiebelException`

Business Object

Table 45 lists a summary of the Siebel BusObject Methods syntax.

Table 45. SiebelBusObject Methods Syntax Summary

Method	Description	Syntax
GetBusComp	Returns the specified business component.	`SiebelBusComp getBusComp (String busCompName) throws SiebelException`
Name	Returns the name of the business object.	`String name ()`
Release	Enables the release of the business object and its resources on the Siebel Server.	`void release() {}`

Business Service

Table 46 lists a summary of the SiebelService Methods syntax.

Table 46. SiebelService Methods Syntax Summary

Method	Description	Syntax
GetFirstProperty	Retrieves the name of the first property of a business service.	`String getFirstProperty()`
GetNextProperty	Once the name of the first property has been retrieved, retrieves the name of the next property of a business service.	`String getNextProperty()`
GetProperty	Retrieves the value stored in the specified property.	`String getProperty(String propName) throws SiebelException`
InvokeMethod	Calls a specialized method or a user-created method on the business service.	`boolean invokeMethod(String methodName, SiebelPropertySet inputPropertySet, SiebelPropertySet outputPropertySet) throws SiebelException`
Name	Returns the name of the service.	`String getName()`

Table 46. SiebelService Methods Syntax Summary

Method	Description	Syntax
PropertyExists	Returns a Boolean value indicating whether the property specified in the argument exists.	`boolean propertyExists(String propName) throws SiebelException`
Release	Enables the release of the Business Service and its resources on the Siebel Server.	`void release()`
RemoveProperty	Removes a property from a business service.	`void removeProperty(String propName) trows SiebelException`
SetProperty	Assigns a value to a property of a business service.	`void setProperty(String propName, String propValue) throws SiebelException`

Property Set

Table 47 lists a summary of the SiebelPropertySet Methods syntax.

Table 47. SiebelPropertySet Methods Syntax Summary

Method	Description	Syntax
AddChild	Adds subsidiary property sets to a property set.	`int addChild(SiebelPropertySet propertySet)`
Copy	Returns a copy of a property set.	`SiebelPropertySet copy(SiebelPropertySet propertySet)`
GetChild	Returns a specified child property set of a property set.	`SiebelPropertySet getChild(int index)`
GetChildCount	Returns the number of child property sets attached to a parent property set.	`int getChildCount()`
GetFirstProperty	Returns the name of the first property in a property set.	`String getFirstProperty()`
GetNextProperty	Returns the name of the next property in a property set.	`String getNextProperty()`
GetProperty	Returns the value of a property when given the property name.	`String getProperty(String propertyName)`

Table 47. SiebelPropertySet Methods Syntax Summary

Method	Description	Syntax
GetPropertyCount	Returns the number of properties attached to a property set.	`int GetPropertyCount ()`
GetType	Returns the value stored in a type in a property set.	`String getType()`
GetValue	Returns a value stored as part of a property set.	`String getValue()`
InsertChildAt	Inserts a child property set into a parent property set at a specific location.	`boolean insertChildAt(SiebelPropertySet propertySet, int index)`
PropertyExists	Returns a Boolean value indicating whether the property specified in the argument exists.	`boolean propertyExists(String propertyName)`
RemoveChild	Removes a child property set as a specified index from a parent property set.	`boolean removeChild(int index)`
RemoveProperty	Removes the property specified in its argument from a property set.	`boolean removeProperty(String propertyName)`
Reset	Removes every property and child property set from a property set.	`boolean reset()`
SetProperty	Assigns a value to the property of a property set specified in its argument.	`boolean setProperty(String propertyName, String propertyValue)`
SetType	Assigns a data value to a type member of a property set.	`boolean setType(java.lang.String type)`
SetValue	Assigns a data value to a value member of a property set.	`boolean setValue(String value)`

SiebelException

Table 48 lists a summary of the SiebelException Methods syntax.

Table 48. SiebelException Methods Syntax Summary

Method	Description	Syntax
GetErrorCode	Gets a numeric error code.	`int getErrorCode()`
GetErrorMessage();	Gets an error message.	`String getErrorMessage()`

For more information on the Java Data Bean Interface, see the Siebel JAR files included with the installation: `SiebelJI_common.jar` and `SiebelJI_<lang>` located in `sea750\siebsrvr\CLASSES`.

CORBA Quick Reference 10

This quick reference has the following topics:

- "CORBA Interfaces and Unicode"
- "Application" on page 396
- "Business Component" on page 398
- "Business Object" on page 401
- "Business Service" on page 402
- "Property Set" on page 403
- "SiebelAppFactory" on page 405
- "SiebelException" on page 405

CORBA Interfaces and Unicode

Methods available to CORBA also have wide versions available (suffixed with a "W"). The wide versions behave identically to the original interfaces, except that string parameters are passed as "wide" strings, which are UTF-16 encoded. Rather than throwing SiebelException, the wide versions throw SiebelExceptionW, which is the wide version of an exception.

These examples from the Siebel Interface Definition Language file (`scorba.idl`) illustrate the difference between regular and wide syntax:

```
string                    GetFieldValue(in string FieldName)
                                raises (SiebelException);

wstring                   GetFieldValueW(in wstring FieldName)
                                raises (SiebelExceptionW);
```

```
string                    GetFirstProperty()
                                raises (SiebelException);

wstring                   GetFirstPropertyW()
                                raises (SiebelExceptionW);


string                    GetNextProperty()
                                raises (SiebelException);

wstring                   GetNextPropertyW()
                                raises (SiebelExceptionW);


void                     InvokeMethod(in string method,
                                in SiebelPropertySet inputs,
                                out SiebelPropertySet outputs)
                                raises (SiebelException);

void                     InvokeMethodW(in wstring method,
                              in SiebelPropertySetW inputs,
                              out SiebelPropertySetW outputs)
                               raises (SiebelExceptionW);
```

Application

Table 49 lists the CORBA interface SiebelApplication methods.

Table 49. CORBA Interface SiebelApplication

Method	Description	IDL
CurrencyCode	Returns the three-letter operating currency code.	`readonly attribute string CurrencyCode;`
GetBusObject	Instantiates and returns a new instance of the argument-specified business object.	`SiebelBusObject GetBusObject(in string BusObjectName) raises (SiebelException);`
GetLastErrText	Returns the last error text message.	`readonly attribute string GetLastErrText;`

Table 49. CORBA Interface SiebelApplication

Method	Description	IDL
GetProfileAttr	Returns the value of an attribute in a user profile.	`string GetProfileAttr(in string name) raises (SiebelException);`
GetSharedGlobal	Gets the shared user-defined global variables.	`string GetSharedGlobal(in string varName) raises (SiebelException);`
GetService	Instantiates and returns a new instance of the argument-specified service.	`SiebelService GetService(in string ServiceName) raises (SiebelException);`
InvokeMethod	Calls the named specialized method.	`string InvokeMethod(in string methodName, in anySeq methodArgs) raises (SiebelException);`
Login	Allows external applications to log in to the CORBA object manager.	`void Login(in string UserName, in string Password) raises (SiebelException);`
LoginId	Returns the login ID of the user who started the Siebel application.	`readonly attribute string LoginId;`
LoginName	Returns the login name of the user who started the Siebel application.	`readonly attribute string LoginName;`
Logoff	Disconnects the client from the server.	`void Logoff() raises (SiebelException);`
PositionId	Returns the position ID that describes the user's current position.	`readonly attribute string PositionId;`
PositionName	Returns the position name of the user's current position.	`readonly attribute string PositionName;`
Release	Destroys every subordinate object and returns the resources.	`void Release() raises (SiebelException);`
SetPositionId	Sets the active position to the Position ID specified in the argument.	`boolean SetPositionID (in string value) raises (SiebelException);`
SetPositionName	Sets the active position to the position name specified in the argument. Returns a Boolean value indicating if the method succeeded.	`boolean SetPositionName (in string value) raises (SiebelException);`

Table 49. CORBA Interface SiebelApplication

Method	Description	IDL
SetProfileAttr	Used in personalization to assign values to attributes in a user profile.	`void SetProfileAttr(in string name, in string value) raises (SiebelException);`
SetSharedGlobal	Sets a shared user-defined global variable.	`void SetSharedGlobal(in string varName, in string value) raises (SiebelException);`
Trace	Appends a message to the trace file.	`void Trace(in string Message) raises (SiebelException);`
TraceOff	Turns off the tracing started by TraceOn.	`void TraceOff() raises (SiebelException);`
TraceOn	Turns tracing on	`void TraceOn(in string FileName, in string Category, in string Source) raises (SiebelException);`

Business Component

Table 50 lists the CORBA interface SiebelBusComp methods.

Table 50. CORBA Interface SiebelBusComp

Method	Description	IDL
ActivateField	Allows queries to retrieve data for the specified field.	`void ActivateField(in string FieldName) raises (SiebelException);`
ActivateMultipleFields	Allows queries to retrieve data for the fields specified in the property set.	`void ActivateMultipleFields(in Siebel PropertySet fields) raises (SiebelException);`
Associate	Creates a new many-to-many relationship for the parent object through an association business component.	`void Associate(in short where) raises (SiebelException);`
BusObject	Returns the business object that contains the business component.	`readonly attribute SiebelBusObject BusObject;`
ClearToQuery	Clears the current query and sort specifications on the business component.	`void ClearToQuery() raises (SiebelException);`

Table 50. CORBA Interface SiebelBusComp

Method	Description	IDL
DeactivateFields	Deactivates every currently activated field.	`void DeactivateFields() raises (SiebelException);`
DeleteRecord	Removes the current record from the business component.	`void DeleteRecord() raises (SiebelException);`
ExecuteQuery	Retrieves a set of BusComp records.	`void ExecuteQuery(in boolean cursorMode) raises (SiebelException);`
ExecuteQuery2	Retrieves a set of BusComp records.	`void ExecuteQuery2(in boolean cursorMode, in boolean ignoreMaxCursorSize) raises (SiebelException);`
FirstRecord	Moves to the first record in the business component.	`boolean FirstRecord() raises (SiebelException);`
FirstSelected	Returns the association business component.	`SiebelBusComp GetAssocBusComp() raises (SiebelException);`
GetFieldValue	Returns a value for the argument-specified field.	`string GetFieldValue(in string FieldName) raises (SiebelException);`
GetFormattedFieldValue	Returns a formatted value for the argument-specified field.	`string GetFormattedFieldValue(in string FieldName) raises (SiebelException);`
GetMultipleFieldValues	Returns a value for the fields specified in the property set.	`void GetMultipleFieldValues(in SiebelPropertySet fields, out SiebelPropertySet values) raises (SiebelException);`
GetMVGBusComp	Returns the MVG business component associated with the argument-specified field.	`SiebelBusComp GetMVGBusComp(in string FieldName) raises (SiebelException);`
GetNamedSearch	Returns the argument-named search specification.	`string GetNamedSearch(in string searchName) raises (SiebelException);`
GetPicklistBusComp	Returns the pick business component associated with the argument-specified field.	`SiebelBusComp GetPicklistBusComp(in string FieldName) raises (SiebelException);`
GetSearchSpec	Returns the current search specification for the argument-specified field.	`string GetSearchSpec(in string FieldName) raises (SiebelException);`
GetUserProperty	Returns the value for an argument-specified property name.	`string GetUserProperty(in string Property) raises (SiebelException);`

Table 50. CORBA Interface SiebelBusComp

Method	Description	IDL
InvokeMethod	Calls the specialized method named in the argument.	`string InvokeMethod(in string methodName, in anySeq methodArgs) raises (SiebelException);`
LastRecord	Moves to the last record in the business component.	`boolean LastRecord() raises (SiebelException);`
Name	Returns the name of the business component.	`readonly attribute string Name;`
NewRecord	Adds a new record to the business component.	`void NewRecord(in short where) raises (SiebelException);`
NextRecord	Moves to the next record in the business component.	`boolean NextRecord() raises (SiebelException);`
ParentBusComp	Returns the parent business component.	`SiebelBusComp ParentBusComp() raises (SiebelException);`
Pick	Places the currently selected record in a picklist business component into the appropriate fields of the parent business component.	`void Pick() raises (SiebelException);`
PreviousRecord	Moves to the previous record in the business component.	`boolean PreviousRecord() raises (SiebelException);`
RefineQuery	Refines a query after a query has been executed.	`void RefineQuery() raises (SiebelException);`
SearchExpr	Sets the search specification for the business component.	`attribute string SearchExpr;`
SetFieldValue	Assigns a new value to the named field for the current row of the business component.	`void SetFieldValue(in string FieldName, in string FieldValue) raises (SiebelException);`
SetFormattedFieldValue	Accepts the field value in the current local format and assigns the new value to the named field for the current row of the business component.	`void SetFormattedFieldValue(in string FieldName, in string FieldValue) raises (SiebelException);`
SetMultipleFieldValues	Assigns a new value to the fields specified in the property set for the current row of the business component.	`void SetMultipleFieldValues(in SiebelPropertySet fieldVals) raises (SiebelException);`

Table 50. CORBA Interface SiebelBusComp

Method	Description	IDL
SetNamedSearch	Sets a named search specification on the business component.	`void SetNamedSearch(in string searchName, in string searchText) raises (SiebelException);`
SetSearchSpec	Sets the search specification for the specified field.	`void SetSearchSpec(in string FieldName, in string SearchSpec) raises (SiebelException);`
SetSortSpec	Sets the sort specification for a query.	`void SetSortSpec(in string SortSpec) raises (SiebelException);`
SetUserProperty	Sets the value of the specified User Property.	`void SetUserProperty(in string Property, in string value) raises (SiebelException);`
UndoRecord	Deletes an active record created by NewRecord.	`void UndoRecord() raises (SiebelException);`
ViewMode	Sets and gets the visibility type for the business component.	`attribute short ViewMode;`
WriteRecord	Commits to the database any changes made to the current record.	`void WriteRecord() raises (SiebelException);`

Business Object

Table 51 lists the CORBA interface SiebelBusObject methods.

Table 51. CORBA Interface SiebelBusObject

Method	Description	IDL
GetBusObject	Returns the specified business component.	`SiebelBusComp GetBusComp(in string BusCompName) raises (SiebelException);`
Name	Returns the name of the control.	`readonly attribute string Name;`

Business Service

Table 52 lists the CORBA interface SiebelService methods.

Table 52. CORBA Interface SiebelService

Method	Description	IDL
GetFirstProperty	Retrieves the name of the first property of a business service.	`string GetFirstProperty() raises (SiebelException);`
GetNextProperty	Once the name of the first property has been retrieved, retrieves the name of the next property of a business service.	`string GetNextProperty() raises (SiebelException);`
GetProperty	Retrieves the value stored in the specified property.	`string GetProperty(in string propName) raises (SiebelException);`
Name	Returns the name of the business service.	`readonly attribute string name;`
InvokeMethod	Calls a specialized method or a user-created method on the business service.	`void InvokeMethod(in string method, in SiebelPropertySet inputs, out SiebelPropertySet outputs) raises (SiebelException);`
PropertyExists	Returns a Boolean value indicating whether the property specified in the argument exists.	`boolean PropertyExists(in string propName) raises (SiebelException);`
RemoveProperty	Removes a property from a business service.	`void RemoveProperty(in string propName) raises (SiebelException);`
SetProperty	Assigns a value to a property of a business service	`void SetProperty(in string propName, in string propValue) raises (SiebelException);`

Property Set

In the CORBA Object Manager, the Property Set object is implemented as a structure (and potentially an array of structures) called SiebelPropertySet in the IDL, rather than as an interface. Therefore, the following Property Set methods are not applicable with respect to the CORBA Object Manager: AddChild, Copy, GetChild, GetChildCount, GetFirstProperty, GetNextProperty, GetProperty, GetType, GetValue, InsertChildAt, PropertyExists, Remove Child, Remove Property, Reset, SetProperty, SetType, and SetValue. The client application constructs a SiebelPropertySet in C++ and then passes it to every method that takes property sets as arguments, for example, SiebelService InvokeMethod.

The advantage of this implementation is that the property set can be constructed entirely on the client side instead of incurring network traffic associated with constructing it through an interface. The methods defined for the PropertySet differ greatly from the methods available through COM. However, the same functionality is available through either interface.

A SiebelPropertyPair represents a distinct name/value combination to describe an attribute of an object. The SiebelPropertyPair structure is defined in the IDL as follows:

```
struct SiebelPropertyPair
{
   string name;      //Name of the attribute
   string value;      //Description of the attribute
};
```

A SiebelPropertySet represents a collection of SiebelPropertyPairs (structures) that describe an object and its attributes. The SiebelPropertySet structure is defined in the IDL as follows:

```
struct SiebelPropertySet
{
   string type;
   string value;
   sequenceSiebelPropertyPair properties;
   sequenceSiebelPropertySet children;
};
typedef sequenceany anySeq;
```

The following C++ code fragment from a client application demonstrates the construction of a SiebelPropertySet and passing it to a SiebelService method.

```
// Declarations

SiebelPropertyPair inputPP1, inputPP2, inputPP3;
SiebelPropertySet inputPS;
SiebelPropertySet* outputPS;
SiebelService pService;

// Setting up the Input Property Set
   inputPS.type = "root";
   inputPS.value = "rootValue";

// Specifying the number or SiebelPropertyPairs to be
// added to the PropertySet
   inputPS.properties.length(3);

// Defining the first property
   inputPP1.name = "prop1";
   inputPP1.value = "value1";
   inputPS.properties[0] = inputPP1;

   inputPP2.name = "prop2";
   inputPP2.value = "value2";
   inputPS.properties[1] = inputPP2;

   inputPP3.name = "prop3";
   inputPP3.value = "value3";
   inputPS.properties[2] = inputPP3;

// Adding 2 Child Property Sets
   inputPS.children.length(2);

   inputPS.children[0].type = "child1";
   inputPS.children[0].properties.length(1);
   inputPS.children[0].properties[0].name = "child1:prop1";
   inputPS.children[0].properties[0].value = "child1:value1";

   inputPS.children[1].type = "child2";
   inputPS.children[1].properties.length(2);
   inputPS.children[1].properties[0].name = "child2:prop1";
   inputPS.children[1].properties[0].value = "child2:value1";
   inputPS.children[1].properties[1].name = "child2:prop2";
   inputPS.children[1].properties[1].value = "child2:value2";

// Passing the Input Property Set and a pointer to the Output
// Property Set to the Service Method
   pService->InvokeMethod ("test1", inputPS, outputPS);
```

SiebelAppFactory

Table 53 lists the CORBA interface SiebelAppFactory methods.

Table 53. CORBA Interface SiebelAppFactory

Method	Description	IDL
CreateSiebelAppObject	Returns a CORBA object reference for an application object.	`SiebelApplication CreatSiebelAppObject() raises (SiebelException);`
Release	Destroys every subordinate object and returns the resources.	`void Release(in SiebelApplication app) raises (SiebelException);`

SiebelException

The following topics describe SiebelException.

- "desc"
- "id" on page 406

desc

A textual description of the error that was raised.

Syntax exObj.desc

Argument	Description
Not applicable	

Used With CORBA Object Manager

Example

```
catch (const SiebelException& e)
{
   CERR << "Unexpected Siebel exception: " << ENDL;
   CERR << "Error code : " << e.id << ENDL;
   CERR << "Error text : " << e.desc << ENDL;
}
```

id

A Long representing the numeric value of the error that was raised.

Syntax

exObj.id

Argument	Description
Not applicable	

Used With

CORBA Object Manager

Example

```
catch (const SiebelException& e)
   {
      CERR << "Unexpected Siebel exception: " << ENDL;
      CERR << "Error code : " << e.id << ENDL;
      CERR << "Error text : " << e.desc << ENDL;
   }
```

Siebel VB Quick Reference 11

This quick reference has the following topics:

- "Applet"
- "Application" on page 409
- "Business Component" on page 412
- "Business Object" on page 418
- "Business Service" on page 418
- "Property Set" on page 420
- "Miscellaneous Interfaces and Events" on page 422

Applet

Table 54 lists a summary of the Applet Methods syntax.

Table 54. Applet Methods Syntax Summary

Method	Description	Syntax

BusComp	Function that returns the business component that is associated with the applet.	`Dim oApplet as Applet`
		`Dim oBusComp as BusComp`
		`Set oBusComp = oApplet.BusComp`

BusObject	Function that returns the business object for the business component for the applet.	`Dim oApplet as Applet`
		`Dim oBusObject as BusObject`
		`Set oBusObject = oApplet.BusObject`

Table 54. Applet Methods Syntax Summary

Method	Description	Syntax

InvokeMethod	Calls an argument-specified specialized method.	`Dim oApplet as Applet`
		`oApplet.InvokeMethod` *`methodName`* `as String,`
		`methodArgs as String or StringArray`

Name	Function that returns the name of the applet.	`Dim oApplet as Applet`
		`Dim sApplet as String`
		`sApplet = oApplet.Name`

Table 55 lists a summary of the WebApplet Events.

Table 55. WebApplet Events Summary

Event	Description	Syntax
WebApplet_InvokeMethod()	Called after a specialized method or a user-defined method on the Web applet has been executed.	`Applet_InvokeMethod(MethodName as String)`
WebApplet_PreCanInvokeMethod	Called before the PreInvokeMethod, allowing the developer to determine whether or not the user has the authority to invoke the business service method.	`Applet_PreCanInvokeMethod(MethodName as String)`
WebApplet_PreInvokeMethod()	Called before a specialized method for the Web applet is invoked or a user-defined method is invoked through oWebApplet.Invoke Method.	`Applet_PreInvokeMethod(MethodName as String)`
WebApplet_Load	Called when the applet loses focus.	`Applet_Load`

Table 55. WebApplet Events Summary

Event	Description	Syntax
WebApplet_ShowControl	Allows scripts to modify the HTML generated by the Siebel Web Engine to render a control on a Web page in a Standard Activity application.	`WebApplet_ShowControl`
WebApplet_ShowListColumn	Allows scripts to modify the HTML generated by the Siebel Web Engine to render a list column on a Web page in a Standard Activity application.	`WebApplet_ShowListColumn`

Application

Table 56 lists a summary of the Application Methods syntax.

Table 56. Application Methods Syntax Summary

Method	Description	Syntax

ActiveBusObject	Returns the business object for the business component of the active applet.	`Dim oApplication as Application`
		`Dim oBusObject as BusObject`
		`Set oBusObject = oApplication.ActiveBusObject`

ActiveViewName	Function that returns the name of the active view.	`Dim oApplication as Application`
		`Dim sView as String`
		`sView = oApplication.ActiveViewName`

CurrencyCode	Returns the three-letter operating currency code.	`Dim oApplication as Application`
		`Dim sCur as String`
		`sCur = oApplication.CurrencyCode`

GetBusObject	Instantiates and returns a new instance of the argument-specified business object.	`Dim oApplication as Application`
		`Dim oBusObject as BusObject`
		`set oBusObject = oApplication.GetBusObject`
		`busobject` `as String`

Table 56. Application Methods Syntax Summary

Method	Description	Syntax

Name	Returns the value of an attribute in a user profile.	`Dim oApplication as Application`
		`Dim sAttr as String`
		`SAttr = oApplication.GetProfileAttr(`*name* `as String)`

GetService	Instantiates and returns a new instance of the argument-specified service.	`Dim oApplication as Application`
		`Dim oService as Service`
		`set oService = oApplication.GetService(`*serviceName* `as String)`

GetSharedGlobal	Gets the shared user-defined global variables.	`Dim oApplication as Application`
		`Dim sName as String`
		`sName = Application.GetSharedGlobal(`*varName* `as String)`

GotoView	Activates the named view and its business object.	`Dim oApplication as Application`
		`oApplication.GotoView` *viewName* `as String,`
		`[`*BusinessObjectName* `as string]`

InvokeMethod	Calls the named specialized method.	`Dim oApplication as Application`
		`Dim sReturn as String`
		`sReturn = oApplication.InvokeMethod(methodName as String,` *methodArgs* `as String or StringArray)`

LoginId	Function that returns the login ID of the user who started the Siebel application.	`Dim oApplication as Application`
		`Dim sID as String`
		`iID = oApplication.LoginId`

LoginName	Function that returns the login name of the user who started the Siebel application.	`Dim oApplication as Application`
		`Dim sUser as String`
		`sUser = oApplication.`*LoginName*

NewPropertySet	Constructs and returns a new property set object.	`Dim oApplication as Application`
		`Dim oPropSet as ProperySet`
		`oPropSet = oApplication.NewPropertySet()`

PositionId	Function that returns the position ID that describes the user's current position.	`Dim oApplication as Application`
		`Dim sRow as String`
		`sRow = oApplication.PositionId`

PositionName	Function that returns the position name of the user's current position.	`Dim oApplication as Application`
		`Dim sPosition as String`
		`sPosition = oApplication.`*PositionName*

RaiseError	Raises a scripting error message to the browser. The error code is a canonical number.	`Dim oApplication as Application` `oApplication.RaiseError keyValue as String, parma1 as String, ...`

Table 56. Application Methods Syntax Summary

Method	Description	Syntax
RaiseErrorText	Raises a scripting error message to the browser. The error text is the specified literal string.	`Dim oApplication as Application` `oApplication.RaiseErrorText message as String`
SetPositionId	Sets the active position to the position ID specified in the argument.	`Dim oApplication as Application` `oApplication.SetPositionIdposId as string`
SetPositionName	Sets the active position to the position name specified in the argument. Returns a Boolean value indicating whether or not method succeeded.	`Dim oApplication as Application` `oApplication.SetPositionNameposName as string`
SetProfileAttr	Used in personalization to assign values to attributes in a user profile.	`Dim oApplication as Application` `oApplication.SetProfileAttr` *`name`* `as String,` *`value`* `as String`
SetSharedGlobal	Sets a shared user-defined global variable.	`Dim oApplication as Application` `oApplication.SetSharedGlobal` *`varName`* `as String,` *`value`* `as String`
Trace	Appends a message to the trace file.	`Dim oApplication as Application` `oApplication.Trace` *`message`* `as String`
TraceOff	Turns off the tracing started by TraceOn.	`Dim oApplication as Application` `oApplication.TraceOff`
TraceOn	Turns tracing on.	`Dim oApplication as Application` `oApplication.TraceOn` *`filename`* `as String,` *`type`* `as Integer,` *`selection`* `as String`

Table 57 lists a summary of the Application Events.

Table 57. Application Events Summary

Event	Description	Syntax
Application_Close	Called before the application exits.	`Application_Close`
Application_Navigate	Called after the client has navigated to a view.	`Application_Navigate (As String)`

Table 57. Application Events Summary

Event	Description	Syntax
Application_InvokeMethod	Called after a specialized method is invoked.	`Application_InvokeMethod(methodName as String)`
Application_PreInvokeMethod	Called before a specialized method is invoked.	`Application_PreInvokeMethod(methodName as String)`
Application_PreNavigate	Called before the client has navigated from one view to the next.	`Application_PreNavigate (DestViewName, DestBusObjName As String) As Integer`
Application_Start	Called when the client starts.	`Application_Start(commandLine as String)`

Business Component

Table 58 lists a summary of the Business Component Methods syntax.

Table 58. Business Component Methods Syntax Summary

Method	Description	Syntax
ActivateField	Allows queries to retrieve data for the specified field.	`Dim oBusComp as BusComp` `oBusComp.ActivateField fieldName as String`
ActivateMultipleFields	Allows queries to retrieve data for the fields specified in the property set.	`Dim oBusComp as BusComp` `oBusComp.ActivateMultipleFields oPropSet as PropertySet`
Associate	Creates a new many-to-many relationship for the parent object through an association business component.	`Dim oBusComp as BusComp` `oBusComp.Associate whereIndicator as Integer`

BusObject	Function that returns the business object that contains the business component.	`Dim oBusComp as BusComp`
		`Dim oBusObject as BusObject`
		`Set oBusObject = oBusComp.BusObject`

ClearToQuery	Clears the current query and sort specifications on the business component.	`Dim oBusComp as BusComp` `oBusComp.ClearToQuery`

Table 58. Business Component Methods Syntax Summary

Method	Description	Syntax
DeactivateFields	Deactivates every currently activated field.	`Dim oBusComp as BusComp` `oBusComp.DeactivateFields`
DeleteRecord	Removes the current record from the business component.	`Dim oBusComp as BusComp` `oBusComp.DeleteRecord`
ExecuteQuery	Retrieves a set of BusComp records.	`Dim oBusComp as BusComp` `oBusComp.ExecuteQuery` *`cursorMode`* `as Integer`

ExecuteQuery2	Retrieves a set of BusComp records.	`Dim oBusComp as BusComp`
		`oBusComp.ExecuteQuery2` *`cursorMode`* `as Integer,`
		`ignoreMaxCursorSize` `as Integer`

FirstRecord	Moves to the first record in the business component.	`Dim oBusComp as BusComp`
		`Dim iIsRecord as Integer`
		`iIsRecord = oBusComp.FirstRecord`

FirstSelected	Moves the focus to the first record of the multiple selection in the business component.	`Dim oBusComp as BusComp`
		`Dim iIsMultipleSection as Integer`
		`iIsMultipleSelection = oBusComp.FirstSelected`

GetAssocBusComp	Function that returns the association business component.	`Dim oBusComp as BusComp`
		`Dim AssocBusComp as BusComp`
		`Set AssocBusComp = oBusComp.GetAssocBusComp`

GetFieldValue	Function that returns a value for the argument-specified field.	`Dim oBusComp as BusComp`
		`Dim sValue as String`
		`sValue = oBusComp.GetFieldValue(`*`FieldName`* `as String)`

GetFormattedFieldValue	Function that returns a formatted value for the argument-specified field.	`Dim oBusComp as BusComp`
		`Dim sValue as String`
		`sValue = oBusComp.GetFormattedFieldValue(`*`FieldName`* `as String)`

GetMultipleFieldValues	Returns a value for the fields specified in the property set.	`Dim oBusComp as BusComp` `oBusComp.GetMultipleFieldValues` *`oFields`* `as PropertySet,` *`oValues`* `as PropertySet`

GetMVGBusComp	Function that returns the MVG business component associated with the argument-specified field.	`Dim oBusComp as BusComp`
		`Dim MvgBusComp as BusComp`
		`set MvgBusComp = oBusComp.GetMVGBusComp(`*`FieldName`* `as String)`

GetNamedSearch	Function that returns the argument-named search specification.	`Dim oBusComp as BusComp`
		`Dim sValue as String`
		`sValue = oBusComp.GetNamedSearch(`*`SearchName`* `as String)`

Table 58. Business Component Methods Syntax Summary

Method	Description	Syntax

GetPicklistBusComp	Function that returns the pick business component associated with the argument-specified field.	`Dim oBusComp as BusComp`
		`Dim pickBusComp as BusComp`
		`Set pickBusComp =`
		`oBusComp.GetPicklistBusComp(`*`FieldName`*` as String)`

GetSearchExpr	Function that returns the current search expression.	`Dim oBusComp as BusComp`
		`Dim sExpr as String`
		`sExpr = oBusComp.GetSearchExpr`

GetSearchSpec	Function that returns the current search specification for the argument-specified field.	`Dim oBusComp as BusComp`
		`Dim sSpec as String`
		`sSpec = oBusComp.GetSearchSpec(`*`FieldName`*` as String)`

GetUserProperty	Function that returns the value for an argument-specified property name.	`Dim oBusComp as BusComp`
		`Dim sValue as String`
		`sValue = oBusComp.GetUserProperty(`*`propertyName`*` as String)`

GetViewMode	Function that returns the visibility mode for the business component.	`Dim oBusComp as BusComp`
		`Dim iMode as Integer`
		`iMode = oBusComp.GetViewMode`

InvokeMethod	Calls the specialized method named in the argument.	`Dim oBusComp as BusComp`
		`Dim Return`
		`Return = oBusComp.InvokeMethod(`*`methodName`*` as String, `*`methodArgs`*` as String or StringArray)`

LastRecord	Moves to the last record in the business component.	`Dim oBusComp as BusComp`
		`Dim iReturn as Integer`
		`iReturn = oBusComp.LastRecord`

Name	Function that returns the name of the business component.	`Dim oBusComp as BusComp`
		`Dim sName as String`
		`sName = oBusComp.Name`

NewRecord	Adds a new record to the business component.	`Dim oBusComp as BusComp` `oBusComp.NewRecord(`*`whereIndicator`*` as Integer)`

NextRecord	Moves to the next record in the business component.	`Dim oBusComp as BusComp`
		`Dim iReturn as Integer`
		`iReturn = oBusComp.NextRecord`

NextSelected	Moves to the next record of the current multiple selection.	`Dim oBusComp as BusComp`
		`Dim iReturn as Integer`
		`iReturn = oBusComp.NextSelected`

ParentBusComp	Function that returns the parent business component.	`Dim oBusComp as BusComp`
		`Dim parentBusComp as BusComp`
		`Set parentBusComp = oBusComp.ParentBusComp`

Table 58. Business Component Methods Syntax Summary

Method	Description	Syntax
Pick	Places the currently selected record in a picklist business component into the appropriate fields of the parent business component.	`Dim oBusComp as BusComp` `oBusComp.Pick`

PreviousRecord	Moves to the previous record in the business component.	`Dim oBusComp as BusComp`
		`Dim iReturn as Integer`
		`iReturn = oBusComp.PreviousRecord`

RefineQuery	Refines a query after a query has been executed.	`Dim oBusComp as BusComp` `oBusComp.RefineQuery`

SetFieldValue	Assigns a new value to the named field for the current row of the business component.	`Dim oBusComp as BusComp`
		`oBusComp.SetFieldValue` *`FieldName`* `as String,`
		`FieldValue` `as String`

SetFormattedFieldValue	Accepts the field value in the current local format and assigns the new value to the named field for the current row of the business component.	`Dim oBusComp as BusComp` `oBusComp.SetFormattedFieldValue` *`FieldName`* `as String,` *`FieldValue`* `as String`
SetMultipleFieldValues	Assigns a new value to the fields specified in the property set for the current row of the business component.	`Dim oBusComp as BusComp` `oBusComp.SetMultipleFieldValues` *`oPropSet`* `as PropertySet`

SetNamedSearch	Sets a named search specification on the business component.	`Dim oBusComp as BusComp`
		`oBusComp.SetNamedSearch` *`searchName`* `as String,`
		`searchSpec` `as String`

SetSearchExpr	Sets the search specification for the business component.	`Dim oBusComp as BusComp` `oBusComp.SetSearchExpr` *`searchSpec`* `as String`

SetSearchSpec	Sets the search specification for the specified field.	`Dim oBusComp as BusComp`
		`oBusComp.SetSearchSpec` *`fieldName`* `as String,`
		`searchSpec` `as String )`

SetSortSpec	Sets the sort specification for a query.	`Dim oBusComp as BusComp` `oBusComp.SetSortSpec` *`sortSpec`* `as String`

Table 58. Business Component Methods Syntax Summary

Method	Description	Syntax
SetUserProperty	Sets the value of the specified User Property.	`Dim oBusComp as BusComp` `oBusComp.SetUserProperty` *`propertyName`* `as String,` *`newValue`* `as String`
SetViewMode	Sets the visibility type for the business component.	`Dim oBusComp as BusComp` `oBusComp.SetViewMode` *`viewMode`* `as Integer`
UndoRecord	Deletes an active record created by NewRecord.	`Dim oBusComp as BusComp` `oBusComp.UndoRecord`
WriteRecord	Commits to the database any changes made to the current record.	`Dim oBusComp as BusComp` `oBusComp.WriteRecord`

Table 59 lists a summary of the Business Components Events.

Table 59. Business Component Events Summary

Event	Description	Syntax
BusComp_Associate	Called to create an association after a record is added to a business component.	`BusComp_Associate`
BusComp_ChangeRecord	Called after the current row changes in the business component.	`BusComp_ChangeRecord`
BusComp_CopyRecord	Called after a new row is copied in the business component.	`BusComp_CopyRecord`
BusComp_DeleteRecord	Called after a row is deleted in the business component.	`BusComp_DeleteRecord`
BusComp_InvokeMethod	Calls a specialized method whose name is specified in its argument.	`BusComp_InvokeMethod(`*`methodName`* `as String)`
BusComp_NewRecord	Called after a new row has been created and made active in the business component.	`BusComp_NewRecord`
BusComp_PreAssociate	Called before a record is added to a business component to create an association.	`BusComp_PreAssociate`

Table 59. Business Component Events Summary

Event	Description	Syntax
BusComp_PreCopyRecord	Called before a new row is copied in the business component.	`BusComp_PreCopyRecord`
BusComp_PreDeleteRecord	Called before a row is deleted in the business component.	`BusComp_PreDeleteRecord`
BusComp_PreGetFieldValue	Called when the value of a business component field is accessed.	`BusComp_PreGetFieldValue(`*`FieldName`* `as String,` *`FieldValue`* `as String)`
BusComp_PreInvokeMethod	Called before a specialized method is invoked on a business component.	`BusComp_PreInvokeMethod(methodName as String, Arguments as String or StringArray)`
BusComp_PreNewRecord	Called before a new row is created in the business component.	`BusComp_PreNewRecord`
BusComp_PreQuery	Called before query execution.	`BusComp_PreQuery`
BusComp_PreSetFieldValue	Called when a value is pushed down into the business component from the user interface.	`BusComp_PreSetFieldValue(`*`FieldName`* `as String,`*`FieldValue`* `as String)`
BusComp_PreWriteRecord	Called before a row is written out to the database.	`BusComp_PreWriteRecord`
BusComp_Query	Called after the query is complete and every row has been retrieved, but before they have been displayed.	`BusComp_Query`
BusComp_SetFieldValue	Called after a value has been pushed down into the business component from the user interface.	`BusComp_SetFieldValue(`*`fieldName`* `as String)`
BusComp_WriteRecord	Called after a row is written to the database.	`BusComp_WriteRecord`

Business Object

Table 60 lists a summary of the Business Object Methods syntax.

Table 60. Business Object Methods Syntax Summary

Method	Description	Syntax

GetBusComp	Function that returns the specified business component.	`Dim oBusObject as BusObject`
		`Dim oBusComp as BusComp`
		`set oBusComp = BusObject.GetBusComp(`*`BusCompName`*` as String)`

Name	Function that returns the name of the control.	`Dim oBusObject as BusObject`
		`Dim sName as String`
		`sName = oBusObject.Name`

Business Service

Table 61 lists a summary of the Business Service Methods syntax.

Table 61. Business Service Methods Syntax Summary

Method	Description	Syntax

GetFirstProperty	Retrieves the name of the first property of a business service.	`Dim oService as Service`
		`Dim sName as String`
		`sName = oService.GetFirstProperty()`

GetNextProperty	Once the name of the first property has been retrieved, retrieves the name of the next property of a business service.	`Dim oService as Service`
		`Dim sName as String`
		`sName = oService.GetNextProperty()`

GetProperty	Retrieves the value stored in the specified property.	`Dim oService as Service`
		`Dim sValue as String`
		`sValue = oService.GetProperty(`*`propName`*` as String)`

InvokeMethod	Calls a specialized method or a user-created method on the business service.	`Dim oService as Service`
		`Dim Return`
		`Return = oService.InvokeMethod(methodName as String, InputArguments as PropertySet, OutputArguments as PropertySet)`

Name	Returns the name of the business service.	`Dim oService as Service`
		`Dim sName as String`
		`sName = oService.Name`

Table 61. Business Service Methods Syntax Summary

Method	Description	Syntax
PropertyExists	Returns a Boolean value indicating whether the property specified in the argument exists.	`Dim oService as Service` `oService.PropertyExists(`*`propName`*` as String)`
RemoveProperty	Removes a property from a business service.	`Dim oService as Service` `oService.RemoveProperty` *`propName`* `as String`
SetProperty	Assigns a value to a property of a business service.	`Dim oService as Service` `oService.SetProperty` *`propName`* `as String,` *`propValue`* `as String`

Table 62 lists a summary of the Business Service Events.

Table 62. Business Service Events Syntax Summary

Method	Description	Syntax
Service_InvokeMethod	Called when a business service is accessed.	`Service_InvokeMethod(methodName as String)`
Service_PreCanInvokeMethod	Called before the PreInvokeMethod, allowing the developer to determine whether or not the user has the authority to invoke the business service method.	`Service_PreCanInvokeMethod(methodName as String, CanInvoke As String) as Integer)`
Service_PreInvokeMethod	Called before a specialized method is invoked on a business service.	`Service_PreInvokeMethod(methodName as String, Inputs as PropertySet, Outputs as PropertySet)`

Property Set

Table 63 lists a summary of the Property Set Methods syntax.

Table 63. Property Set Methods Syntax Summary

Method	Description	Syntax

AddChild	Adds subsidiary property sets to a property set.	`Dim oPropSet as PropertySet`
		`Dim iIndex as Integer`
		`oPropSet.AddChild(`*`childObject`*` as Property Set)`

Copy	Returns a copy of a property set.	`Dim oPropSet1 as PropertySet`
		`Dim oPropSet2 as PropertySet`
		`oPropSet2 = oPropSet1.Copy()`

GetChild	Returns a specified child property set of a property set.	`Dim oPropSet as PropertySet`
		`Dim sPropVal as String`
		`sPropVal = oPropSet.GetChild(`*`index`*` as Integer)`

GetChildCount	Returns the number of child property sets attached to a parent property set.	`Dim oPropSet as PropertySet`
		`Dim iCount as Integer`
		`iCount = oPropSet.GetChildCount()`

GetFirstProperty	Returns the name of the first property in a property set.	`Dim oPropSet as PropertySet`
		`Dim sPropName as String`
		`sPropName = oPropSet.GetFirstProperty()`

GetNextProperty	Returns the name of the next property in a property set.	`Dim oPropSet as PropertySet`
		`Dim sPropName as String`
		`sPropName = oPropSet.GetNextProperty()`

GetProperty	Returns the value of a property when given the property name.	`Dim oPropSet as PropertySet`
		`Dim sPropVal as String`
		`sPropVal = oPropSet.GetProperty(`*`propName`*` as String)`

GetPropertyCount	Returns the number of properties attached to a property set.	`Dim oPropSet as PropertySet`
		`Dim count as Long`
		`count = oPropSet.GetPropertyCount`

GetType	Returns the value stored in a type in a property set.	`Dim oPropSet as PropertySet`
		`Dim sTypeVal as String`
		`sTypeVal = oPropSet.GetType(`*`value`*` as String)`

GetValue	Returns a value stored as part of a property set.	`Dim oPropSet as PropertySet`
		`Dim sValVal as String`
		`sValVal = oPropSet.GetValue(`*`value`*` as String)`

Table 63. Property Set Methods Syntax Summary

Method	Description	Syntax
InsertChildAt	Inserts a child property set into a parent property set at a specific location.	`Dim oPropSet as PropertySet` `oPropSet.InsertChildAt` *`childObject`* `as String,` *`index`* `as Integer`
PropertyExists	Returns a Boolean value indicating whether the property specified in the argument exists.	`Dim oPropSet as Property Set` `oPropSet.PropertyExists(`*`propName`* `as String)`

GetPropertyCount	Returns the number of properties attached to a property set.	`Dim oPropSet as Property Set`
		`Dim count as Long`
		`count=oPropSet.GetPropertyCount`

RemoveChild	Removes a child property set as a specified index from a parent property set.	`Dim oPropSet as PropertySet` `oPropSet.RemoveChild` *`index`* `as Integer`
RemoveProperty	Removes the property specified in its argument from a property set.	`Dim oPropSet as PropertySet` `oPropSet.RemoveProperty` *`propName`* `as String`
Reset	Removes every property and child property set from a property set.	`Dim oPropSet as PropertySet` `oPropSet.Reset()`
SetProperty	Assigns a value to the property of a property set specified in its argument.	`Dim oPropSet as PropertySet` `oPropSet.SetProperty` *`propName`* `as String,` *`propValue`* `as String`
SetType	Assigns a data value to a type member of a property set.	`Dim oPropSet as PropertySet` `oPropSet.SetType` *`value as String`*
SetValue	Assigns a data value to a value member of a property set.	`Dim oPropSet as PropertySet` `oPropSet.SetValue` *`value`* `as String`

Miscellaneous Interfaces and Events

Table 64 lists a summary of the Miscellaneous Methods syntax.

Table 64. Miscellaneous Methods Syntax Summary

Method	Description	Syntax
TheApplication	Global method that returns the unique object of type Application.	`TheApplication.`*`Application_method`*

Browser Scripting 12

Browser Script executes in and is interpreted by the browser. Browser Scripts are written in JavaScript and interact with the Document Object Model (DOM) as well as with the Siebel Object Model available in the browser through the Browser Interaction Manager. A developer can script the behavior of Siebel events as well as the browser events that are exposed through the DOM. The DOM for Internet Explorer and Netscape Navigator are different. Using Siebel Tools you can write scripts for the appropriate browser type by selecting the appropriate User Agent.

NOTE: Browser Script may only be used with applications which run in High Interactivity mode, except when scripting Control events supported by the Browser Document Object Model. Refer to Table 76 and Table 77 for a list of supported DOM events.

Do not use browser scripts to manipulate the location of a frame or form in the Siebel application because this causes a new page to be loaded. The result is a permission denied error, as it is a violation of good security practices.

For information on generating browser scripts, read *Siebel Developer's Reference*.

Browser Script Events and Methods

The following is a list of the Events and Methods available in Browser Script.

- "Applet" on page 424
- "Application" on page 425
- "Business Component" on page 427
- "Business Object" on page 428

- "Business Service" on page 429
- "PropertySet" on page 430
- "Control" on page 432

See Also "Supported DOM Events for High Interactivity Mode" on page 433
"Supported DOM Events for Standard Interactivity Mode" on page 434

Applet

Table 65 lists a summary of the Applet Methods syntax.

Table 65. Applet Methods Syntax Summary

Method	Description	Syntax
ActiveMode	Returns a string containing the name of the current Web Template mode.	`var oApplet;` `var mode = oApplet.ActiveMode();`
BusComp	Returns the business component that is associated with the applet.	`var oApplet;` `var busComp = oApplet.BusComp()`
BusObject	Returns the business object for the business component for the applet.	`var oApplet;` `var oBusObject = oApplet.BusObject();`

FindActiveXControl	Returns the ActiveX control whose name is specified in the argument.	`var oApplet;`
		`var oControl;`
		`oControl = oApplet.FindActiveXControl(`*`controlName`*`);`

FindControl	Returns the control whose name is specified in the argument.	`var oApplet;`
		`var oControl;`
		`oControl = oApplet.FindControl(`*`controlName`*`);`

InvokeMethod	Calls an argument-specified specialized method.	`var oApplet;`
		`var outPs = theApplication().NewPropertySet();`
		`outPs = oApplet.InvokeMethod(`*`MethodName, inputPropSet`*`);`

Name	Returns the name of the applet.	`var oApplet;` `var name = oApplet.Name();`

Table 66 lists a summary of the Applet Events.

Table 66. Applet Events Summary

Event	Description	Syntax
Applet_ChangeFieldValue	Called when the user updates a field value in the browser.	`Applet_ChangeFieldValue (field, value)`
Applet_ChangeRecord	Called when the user moves to a different row or view.	`Applet_ChangeRecord()`
Applet_InvokeMethod	Called after a specialized method or a user-defined method is invoked.	`Applet_InvokeMethod (name, inputPropSet)`
Applet_Load	Triggered after an applet has loaded and after data is displayed.	`Applet_Load()`
Applet_PreInvokeMethod	Called before a specialized method for the Web applet is invoked or a user-defined method is invoked through oWebApplet.InvokeMethod.	`Applet_PreInvokeMethod (name, inputPropSet)`

Application

Table 67 lists a summary of the Application Methods syntax.

Table 67. Application Methods Syntax Summary

Method	Description	Syntax
ActiveApplet	Returns the name of the applet that has input focus.	`var applet;` `applet = TheApplication().ActiveApplet();`
ActiveBusComp	Returns the business component associated with the active applet.	`var busComp;` `busComp = theApplication().ActiveBusComp()`
ActiveBusObject	Returns the business object for the business component of the active applet.	`var busObject;` `busObject = theApplication().ActiveBusObject();`
ActiveViewName	Returns the name of the active view.	`var viewName;` `viewName = theApplication().ActiveViewName();`

Table 67. Application Methods Syntax Summary

Method	Description	Syntax
FindApplet	Returns the applet object identified in the argument.	`var applet;` `applet = theApplication().FindApplet(appletName);`
GetProfileAttr	Returns the value of an attribute in a user profile.	`var sAttr;` `sAttr = theApplication().GetProfileAttr(name);`
GetService	Instantiates and returns a new instance of the service specified in the argument.	`var svc;` `svc = theApplication().GetService(serviceName);`
InvokeMethod	Calls the named specialized method.	`var outPs;` `outPs = theApplication().InvokeMethod(methodName, inputPropSet);`
Name	Returns the value of an attribute in a user profile.	`var appName;` `appName = theApplication().Name();`
NewPropertySet	Constructs and returns a new property set object.	`var PropSet;` `PropSet = theApplication().NewPropertySet();`
SetProfileAttr	Used in personalization to assign values to attributes in a user profile.	`theApplication().SetProfileAttr(name, value);`
SWEAlert	Displays a modal dialog box containing a message to the user.	`TheApplication().SWEAlert(message)`

Table 68 lists a summary of the Application Events syntax.

Table 68. Application Events Syntax Summary

Event	Description	Syntax
Application_InvokeMethod	Called after a specialized method is invoked.	`Application_InvokeMethod (name, inputPropSet)`
Application_PreInvokeMethod	Called before a specialized method is invoked.	`Application_PreInvokeMethod (name, inputPropSet)`

Business Component

Table 69 lists a summary of the Business Component Methods syntax.

Table 69. Business Component Methods Syntax Summary

Method	Description	Syntax

BusObject	Returns the business object that contains the business component.	`var busComp;`
		`var busObject;`
		`busObject = busComp.BusObject();`

GetFieldValue	Returns a value for the field specified in the argument.	`var busComp;`
		`var value;`
		`value = busComp.GetFieldValue(`*`fieldName`*`);`

GetFormattedFieldValue	Returns a formatted value for the field specified in the argument.	`var busComp;`
		`var sValue;`
		`sValue =`
		`busComp.GetFormattedFieldValue(`*`fieldName`*`);`

GetSearchExpr	Returns the current search expression.	`var busComp;`
		`var sExpr;`
		`sExpr = busComp.GetSearchExpr();`

GetSearchSpec	Returns the current search specification for the field specified in the argument.	`var busComp;`
		`var sSpec;`
		`sSpec = busComp.GetSearchSpec(`*`fieldName`*`);`

Name	Returns the name of the business component.	`var busComp;`
		`var sName;`
		`sName = busComp.Name();`

SetFieldValue	Assigns a new value to the named field for the current row of the business component.	`var busComp;` `busComp.SetFieldValue(`*`fieldName, fieldValue`*`);`
SetFormattedFieldValue	Accepts the field value in the current local format and assigns the new value to the named field for the current row of the business component.	`var busComp;` `busComp.SetFormattedFieldValue(`*`fieldName, fieldValue`*`);`
WriteRecord	Commits to the database any changes made to the current record.	`var busComp;` `busComp.WriteRecord();`

Table 70 lists a summary of the Business Component Events syntax.

Table 70. Business Component Events Syntax Summary

Event	Description	Syntax
BusComp_PreSetFieldValue	Called when a value is pushed down into the business component from the user interface. This Browser Script event is not invoked if the 'Immediate Post Changes' property of the Business Component field is set to TRUE.	`BusComp_PreSetFieldValue(fieldName, value);`

Business Object

Table 71 lists a summary of the Business Object Methods syntax.

Table 71. Business Object Methods Syntax Summary

Method	Description	Syntax

GetBusComp	Function that returns the specified business component.	`var busObject;`
		`var Comp;`
		`busComp = busObject.GetBusComp(busCompName);`

Name	Function that returns the name of the control.	`Var sName;`
		`var busObject;`
		`sName = budObject.Name();`

Business Service

Table 72 lists a summary of the Business Service Methods syntax.

Table 72. Business Service Methods Syntax Summary

Method	Description	Syntax
GetFirstProperty	Retrieves the name of the first property of a business service.	`var svc;` `var sName = svc.GetFirstProperty();`
GetNextProperty	Once the name of the first property has been retrieved, retrieves the name of the next property of a business service.	`var svc;` `var sName = svc.GetNextProperty();`

GetProperty	Retrieves the value stored in the specified property.	`var svc;`
		`var value;`
		`value = svc.GetProperty(`*name*`);`

InvokeMethod	Calls a specialized method or a user-created method on the business service.	`var svc;`
		`var oPropSet =theApplication().NewPropertySet();`
		`oPropSet = svc.InvokeMethod(`*methodName*`,`
		inputPropSet`);`

Name	Returns the name of the business service.	`var svc;`
		`var name;`
		`name = svc.Name();`

PropertyExists	Returns a Boolean value indicating whether the property specified in the argument exists.	`var svc;`
		`var bool;`
		`bool = svc.PropertyExists(`*name*`);`

RemoveProperty	Removes a property from a business service.	`var svc;` `svc.RemoveProperty(`*name*`);`
SetProperty	Assigns a value to a property of a business service.	`var svc;` `svc.SetProperty(`*name*`, `*value*`);`

Table 73 lists a summary of the Business Service Events syntax.

Table 73. Business Service Events Syntax Summary

Method	Description	Syntax
Service_InvokeMethod	Called when a business service is accessed.	`Service_InvokeMethod(methodName,input, output);`
Service_PreCanInvokeMethod	Called before the PreInvokeMethod, allowing the developer to determine whether or not the user has the authority to invoke the business service method.	`Service_PreCanInvokeMethod (methodName)`
Service_PreInvokeMethod	Called before a specialized method is invoked on a business service.	`Service_PreInvokeMethod(methodName, inputPropSet,outputPropSet);`

PropertySet

Table 74 lists a summary of the PropertySet Methods syntax.

Table 74. PropertySet Methods Syntax Summary

Method	Description	Syntax
AddChild	Adds subsidiary property sets to a property set.	`var oPropSet;` `oPropSet.AddChild(childObject);`

Copy	Returns a copy of a property set.	`var oPropSet1;`
		`var oPropSet2;`
		`oPropSet2 = oPropSet1.Copy();`

GetChild	Returns a specified child property set of a property set.	`var oPropSet;`
		`var oChildPropSet;`
		`oChildPropSet = oPropSet.GetChild(index);`

GetChildCount	Returns the number of child property sets attached to a parent property set.	`var oPropSet;`
		`var iCount;`
		`iCount = oPropSet.GetChildCount();`

GetFirstProperty	Returns the name of the first property in a property set.	`var oPropSet;`
		`var sPropName`
		`sPropName = oPropSet.GetFirstProperty();`

Table 74. PropertySet Methods Syntax Summary

Method	Description	Syntax

GetNextProperty	Returns the name of the next property in a property set.	`var oPropSet;`
		`var sPropName`
		`sPropName = oPropSet.GetNextProperty();`

GetProperty	Returns the value of a property when given the property name.	`var oPropSet;`
		`var sValue`
		`sValue = oPropSet.GetProperty(`*`propName`*`)`

GetPropertyCount	Returns the number of properties attached to a property set.	`var oPropSet;`
		`var iCount;`
		`iCount = oPropSet.GetPropertyCount();`

GetType	Returns the value stored in a type in a property set.	`var oPropSet;`
		`var type;`
		`type = oPropSet.GetType();`

GetValue	Returns a value stored as part of a property set.	`var oPropSet;`
		`var sValue;`
		`sValue = oPropSet.GetValue();`

InsertChildAt	Inserts a child property set into a parent property set at a specific location.	`var oPropSet;` `oPropSet.InsertChildAt(`*`childObject, index`*`);`

PropertyExists	Returns a Boolean value indicating whether the property specified in the argument exists.	`var oPropSet;`
		`var bool;`
		`bool = oPropSet.PropertyExists(propName);`

RemoveChild	Removes a child property set as a specified index from a parent property set.	`var oPropSet;` `oPropSet.RemoveChild(`*`index`*`);`
RemoveProperty	Removes the property specified in its argument from a property set.	`var oPropSet;` `oPropSet.RemoveProperty(`*`propName`*`);`
Reset	Removes every property and child property set from a property set.	`var oPropSet;` `oPropSet.Reset();`
SetProperty	Assigns a value to the property of a property set specified in its argument.	`var oPropSet;` `oPropSet.SetProperty` *`propName , propValue`*

Table 74. PropertySet Methods Syntax Summary

Method	Description	Syntax
SetType	Assigns a data value to a type member of a property set.	`var oPropSet;` `oPropSet.SetType(value);`
SetValue	Assigns a data value to a value member of a property set.	`var oPropSet;` `oPropSet.SetValue(value);`

Control

Table 75 lists a summary of the Control Methods syntax.

Table 75. Control Methods Syntax Summary

Method	Description	Syntax

Applet	Returns the parent applet for the control.	`var oControl;`
		`var oApplet;`
		`oApplet = oControl.Applet();`

BusComp	Returns the corresponding business component for the control.	`var oControl;`
		`var busComp;`
		`busComp = oControl.Buscomp();`

GetProperty	Returns the value of the property of a control.	`var oControl;` `oControl.GetProperty(propName);`

GetValue	Returns the value of a control.	`var oControl;`
		`var sValue;`
		`sValue = oControl.GetValue();`

Name	Returns the name of the control.	`var oControl;`
		`var sName;`
		`sName = oControl.Name();`

SetProperty	Sets the visual properties of a control.	`var oControl;` `oControl.SetProperty(propName, propValue);`
SetValue	Sets the contents of the control to the indicated value.	`var oControl;` `oControl.SetValue(value);`

Supported DOM Events for High Interactivity Mode

Table 76 lists the supported DOM Events for High Interactivity mode.

Table 76. Supported DOM Events for High Interactivity Mode

Control	Siebel Control Type	Supported Events	Comments
Button	Native	OnFocus OnBlur	
CheckBox	Native	OnFocus OnBlur	Rendered as Input Type = CHECKBOX.
Link	Native	OnFocus OnBlur	Rendered through paired anchor tags or as INPUT TYPE = TEXT in edit mode.
List Column	Native	This control does not expose any scriptable events.	
Mailto	Native	OnFocus OnBlur	Rendered as anchor tags with HREF = mailto or as INPUT TYPE = TEXT in Edit mode.
MiniButton	Native	OnFocus OnBlur	
Password	Native	OnFocus OnBlur	Rendered as Input Type = password.
Text	Native	OnFocus OnBlur	Rendered as INPUT TYPE = TEXT or as SELECT when attached to a pick list. If there is a pop-up window, it renders as an editbox plus a button.
TextArea	Native	OnFocus OnBlur	Rendered as TEXTAREA.
Tree	Native	At this time, tree applets and controls do not expose any scriptable events.	
URL	Native	OnFocus OnBlur	Rendered through paired anchor tags with an HREF = underlying field value or as INPUT TYPE = TEXT in edit mode.

NOTE: Siebel objects (business components, applets, and so on.) cannot be accessed from DOM events.

To associate a script with the control_OnClick event (High Interactivity mode only), use the Applet_PreInvokeMethod event associated with the applet. For additional information and example, read Chapter 14, "Invoking Custom Methods with MiniButtons."

Supported DOM Events for Standard Interactivity Mode

Table 77 lists the supported DOM Events and template modes for Standard Interactivity mode.

Table 77. Supported DOM Events and Template Modes for Standard Interactivity Mode

Control	Siebel Control Type	Supported Events	Comments
Button	Native	OnFocus (Base/Edit) OnBlur (Base/Edit) OnMouseOut (Base/Edit) OnMouseOver (Base/Edit)	
CheckBox	Native	OnBlur (Base/Edit) OnFocus (Edit) OnChange (Edit) OnMouseOut (Edit) OnMouseOver(Edit)	In Base mode, a CheckBox appears as a Y or N text value. In Edit mode, a CheckBox is rendered as Input Type = CHECKBOX.
Link	Native	OnFocus (Base/Edit) OnBlur (Base/Edit) OnMouseOut (Base/Edit) OnMouseOver (Base/Edit) OnClick (Base/Edit)	Rendered through paired anchor tags or as INPUT TYPE = TEXT in Edit mode.
List Column	Native	List Columns currently do not expose any scriptable events.	

Table 77. Supported DOM Events and Template Modes for Standard Interactivity Mode

Control	Siebel Control Type	Supported Events	Comments
Mailto	Native	OnChange (Edit) OnFocus (Base/Edit) OnBlur (Base/Edit) OnMouseOut (Base/Edit) OnMouseOver (Base/Edit)	Rendered as anchor tags with HREF = mailto or as INPUT TYPE = TEXT in Edit mode.
MiniButton	Native	OnFocus (Base/Edit) OnBlur (Base/Edit) OnMouseOut (Base/Edit) OnMouseOver (Base/Edit) OnClick (Base/Edit)	
Password	Native	OnChange (Edit) OnFocus (Edit) OnBlur (Edit) OnMouseOut (Edit) OnMouseOver (Edit)	In Edit mode, a Password control is rendered as Input type = password.
Text	Native	OnChange (Edit) OnFocus (Edit) OnBlur (Edit) OnMouseOut (Edit) OnMouseOver (Edit)	In base mode, a text control is rendered as plain text, unless there is a pop-up window associated with it. In Edit mode, a TEXT control is rendered as INPUT TYPE = TEXT or as SELECT when attached to a pick list.
TextArea	Native	OnChange (Edit) OnFocus (Edit) OnBlur (Edit) OnMouseOut (Base/Edit) OnMouseOver (Edit)	In base mode, a TEXTAREA control is rendered as plain text, unless there is a pop-up window associated with it. In Edit mode, a TEXTAREA is rendered as INPUT TYPE = TEXTAREA.

Table 77. Supported DOM Events and Template Modes for Standard Interactivity Mode

Control	Siebel Control Type	Supported Events	Comments
Tree	Native	At this time, tree applets and controls do not expose any scriptable events.	
URL	Native	OnChange (Edit) OnFocus (Base/Edit) OnBlur (Base/Edit) OnMouseOut (Base/Edit) OnMouseOver (Base/Edit)	Rendered through paired anchor tags with an HREF = underlying field value or as INPUT TYPE = TEXT in Edit mode.

Siebel eScript Quick Reference 13

This quick reference has the following topics:

- "Applet"
- "Application" on page 439
- "Business Component" on page 441
- "Business Object" on page 447
- "Business Service" on page 447
- "PropertySet" on page 449
- "Miscellaneous Interfaces and Events" on page 450

Applet

Table 78 lists a summary of the Applet Methods syntax.

Table 78. Applet Methods Syntax Summary

Method	Description	Syntax

BusComp	Returns the business component that is associated with the applet.	`var applet;`
		`var myBusComp;`
		`myBusComp = applet.BusComp();`

BusObject	Returns the business object for the business component for the applet.	`var applet;`
		`var busObject;`
		`busObject = applet.BusObject();`

Table 78. Applet Methods Syntax Summary

Method	Description	Syntax
InvokeMethod	Calls an argument-specified specialized method.	`var applet;` `applet.InvokeMethod(`*`methodName, methodArg1, methodArg2, …, methodArgn`*`);`

Name	Returns the name of the applet.	`var applet;`
		`var sApplet;`
		`sApplet = applet.Name();`

Table 79 lists a summary of the WebApplet Events.

Table 79. WebApplet Events Summary

Event	Description	Syntax
WebApplet_InvokeMethod	Called after a specialized method or a user-defined method on the Web applet has been executed.	`Applet_InvokeMethod(`*`MethodName`*`);`
WebApplet_Load	Called when the applet loses focus.	`Applet_Load`
WebApplet_PreCanInvokeMethod	Called before the PreInvokeMethod, allowing the developer to determine whether or not the user has the authority to invoke the business service method.	`WebApplet_PreCanInvokeMethod(`*`MethodName, CanInvoke`*`)`
WebApplet_PreInvokeMethod	Called before a specialized method for the Web applet is invoked or a user-defined method is invoked through oWebApplet.InvokeMethod.	`Applet_PreInvokeMethod(`*`MethodName`*`);`
WebApplet_ShowControl	Allows scripts to modify the HTML generated by the Siebel Web Engine to render a control on a Web page in a Standard Activity application.	`WebApplet_ShowControl (`*`controlName, property, mode, HTML`*`)`
WebApplet_ShowListColumn	Allows scripts to modify the HTML generated by the Siebel Web Engine to render a list column on a Web page in a Standard Activity application.	`WebApplet_ShowListColumn (`*`columnName, property, mode, HTML`*`)`

Application

Table 80 lists a summary of the Application Methods syntax.

Table 80. Application Methods Syntax Summary

Method	Description	Syntax
ActiveBusObject	Returns the business object for the business component for the active applet.	`var busObject;` `busObject = TheApplication().ActiveBusObject();`
ActiveViewName	Returns the name of the active view.	`var sView;` `sView = TheApplication().ActiveViewName();`
CurrencyCode	Returns the three-letter operating currency code.	`var sCur;` `sCur = TheApplication().CurrencyCode();`

GetBusObject	Instantiates and returns a new instance of the business object specified in the argument.	`var myBusObject;`
		`myBusObject =`
		`TheApplication().GetBusObject(`*`BusObject`*`);`

Name	Returns the value of an attribute in a user profile.	`var sAttr;` `sAttr = TheApplication().GetProfileAttr(`*`name`*`);`
GetService	Instantiates and returns a new instance of the service specified in the argument.	`var Service;` `Service = TheApplication().GetService(`*`serviceName`*`);`
GetSharedGlobal	Gets the shared user-defined global variables.	`var sName;` `sName = TheApplication().GetSharedGlobal(`*`varName`*`);`
GotoView	Activates the named view and its business object.	`TheApplication().GotoView(`*`viewName`*`,[` *`BusinessObjectName`*`]);`
InvokeMethod	Calls the named specialized method.	`var sReturn;` `sReturn = TheApplication().InvokeMethod(`*`methodName, methodArg1, methodArg2,..., methodArgn`*`);`
LoginId	Returns the login ID of the user who started the Siebel application.	`var sID;` `sID = TheApplication().LoginId();`
LoginName	Returns the login name of the user who started the Siebel application.	`var sUser;` `sUser = TheApplication().LoginName();`
NewPropertySet	Constructs and returns a new property set object.	`var oPropSet;` `oPropSet = TheApplication().NewPropertySet();`

Table 80. Application Methods Syntax Summary

Method	Description	Syntax
PositionId	Returns the position ID that describes the user's current position.	`var sRow;` `sRow = TheApplication().PositionId();`
PositionName	Returns the position name of the user's current position.	`var sPosition;` `sPosition = TheApplication().PositionName();`

RaiseError	Raises a scripting error message to the browser. The error code is a canonical number.	`var keyVal;`
		`var param1;`
		`TheApplication().RaiseError(keyVal, param1, …)`

RaiseErrorText	Raises a scripting error message to the browser. The error text is the specified literal string.	`var message;` `TheApplication().RaiseErrorText(message);`
SetPositionId	Sets the active position to the position ID specified in the argument.	`TheApplication().SetPositionId(posId);`
SetPositionName	Sets the active position to the position name specified in the argument. Returns a Boolean value indicating whether or not method succeeded.	`TheApplication().SetPositionName(posName);`
SetProfileAttr	Used in personalization to assign values to attributes in a user profile.	`TheApplication().SetProfileAttr(`*`name, value`*`);`
SetSharedGlobal	Sets a shared user-defined global variable.	`TheApplication().SetSharedGlobal(`*`varName, value`*`);`
Trace	Appends a message to the trace file.	`TheApplication().Trace(`*`message`*`);`
TraceOff	Turns off the tracing started by TraceOn.	`TheApplication().TraceOff();`
TraceOn	Turns tracing on.	`TheApplication().TraceOn(`*`filename, type, selection`*`);`

Table 81 lists a summary of the Application Events syntax.

Table 81. Application Events Syntax Summary

Event	Description	Syntax
Application_Close	Called before the application exits.	`Application_Close();`
Application_InvokeMethod	Called after a specialized method is invoked.	`Application_InvokeMethod(methodName);`
Application_Navigate	Called after the client has navigated to a view.	`Application_Navigate()`
Application_PreInvokeMethod	Called before a specialized method is invoked.	`Application_PreInvokeMethod(methodName);`
Application_PreNavigate	Called before the client has navigated from one view to the next.	`Application_PreNavigate (DestViewName, DestBusObjName)`
Application_Start	Called when the client starts.	`Application_Start(commandLine);`

Business Component

Table 82 lists a summary of the Business Component Methods syntax.

Table 82. Business Component Methods Syntax Summary

Method	Description	Syntax
ActivateField	Allows queries to retrieve data for the specified field.	`var myBusComp;` `myBusComp.ActivateField(fieldName);`
ActivateMultipleFields	Allows queries to retrieve data for the fields specified in the property set.	`var myBusComp;` `myBusComp.ActivateMultipleFields(oPropSet);`
Associate	Creates a new many-to-many relationship for the parent object through an association business component.	`var myBusComp;` `myBusComp.Associate(whereIndicator);`

BusObject	Returns the business object that contains the business component.	`var myBusComp;`
		`var busObject;`
		`busObject = myBusComp.BusObject();`

Table 82. Business Component Methods Syntax Summary

Method	Description	Syntax
ClearToQuery	Clears the current query and sort specifications on the business component.	`var myBusComp;` `myBusComp.ClearToQuery();`
DeactivateFields	Deactivates every currently activated field.	`var myBusComp;` `myBusComp.DeactivateFields();`
DeleteRecord	Removes the current record from the business component.	`var myBusComp;` `myBusComp.DeleteRecord()`
ExecuteQuery	Retrieves a set of BusComp records.	`var myBusComp;` `myBusComp.ExecuteQuery(`*`cursorMode`*`);`
ExecuteQuery2	Retrieves a set of BusComp records.	`var myBusComp;` `myBusComp.ExecuteQuery2(`*`cursorMode, ignoreMaxCursorSize`*`);`

FirstRecord	Moves to the first record in the business component.	`var myBusComp;`
		`var bIsRecord;`
		`bIsRecord = myBusComp.FirstRecord();`

FirstSelected	Moves to the first record of the multiple selection in the business component.	`var myBusComp;`
		`var bIsMultipleSelection;`
		`bIsMultipleSelection =`
		`myBusComp.FirstSelected();`

GetAssocBusComp	Returns the association business component.	`var myBusComp;`
		`var AssocBusComp;`
		`AssocBusComp = myBusComp.GetAssocBusComp();`

GetFieldValue	Returns a value for the field specified in the argument.	`var myBusComp;`
		`var sValue`
		`sValue = myBusComp.GetFieldValue(`*`FieldName`*`);`

GetFormattedFieldValue	Returns a formatted value for the field specified in the argument.	`var myBusComp;`
		`var sValue;`
		`sValue =`
		`myBusComp.GetFormattedFieldValue(`*`FieldName`*`);`

GetMultipleFieldValues	Returns a value for the fields specified in the property set.	`var myBusComp;`
		`myBusComp.GetMultipleFieldValues(oFields,`
		`oValues );`

GetMVGBusComp	Returns the MVG business component associated with the field specified in the argument.	`var myBusComp;`
		`var MvgBusComp;`
		`MvgBusComp=`
		`myBusComp.GetMVGBusComp(`*`FieldName`*`);`

Table 82. Business Component Methods Syntax Summary

Method	Description	Syntax

GetNamedSearch	Returns the named search specification specified in the argument.	`var myBusComp;`
		`var sValue;`
		`sValue = myBusComp.GetNamedSearch(SearchName);`

GetPicklistBusComp	Returns the pick business component associated with the field specified in the argument.	`var myBusComp;`
		`var pickBusComp;`
		`pickBusComp =`
		`myBusComp.GetPicklistBusComp(FieldName);`

GetSearchExpr	Returns the current search expression.	`var myBusComp;`
		`var sExpr;`
		`sExpr = myBusComp.GetSearchExpr()`

GetSearchSpec	Returns the current search specification for the field specified in the argument.	`var myBusComp;`
		`var sSpec;`
		`sSpec = myBusComp.GetSearchSpec(FieldName);`

GetUserProperty	Returns the value for a property name specified in the argument.	`var myBusComp;`
		`var sValue;`
		`sValue =`
		`myBusComp.GetUserProperty(propertyName);`

GetViewMode	Returns the visibility mode for the business component.	`var myBusComp;`
		`var iMode;`
		`iMode = myBusComp.GetViewMode();`

InvokeMethod	Calls the specialized method named in the argument.	`var myBusComp;`
		`var sReturn;`
		`sReturn = myBusComp.InvokeMethod(methodName,`
		`methodArg1, methodArg2,..., methodArgn);`

LastRecord	Moves to the last record in the business component.	`var myBusComp;`
		`var iReturn;`
		`iReturn = myBusComp.LastRecord();`

Name	Returns the name of the business component.	`var myBusComp;`
		`var sName;`
		`sName = myBusComp.Name();`

NewRecord	Adds a new record to the business component.	`var myBusComp;` `myBusComp.NewRecord(whereIndicator);`

NextRecord	Moves to the next record in the business component.	`var myBusComp;`
		`var bFound;`
		`bFound = myBusComp.NextRecord();`

NextSelected	Moves to the next record of the current multiple selection.	`var myBusComp;`
		`var iReturn;`
		`iReturn = myBusComp.NextSelected();`

Table 82. Business Component Methods Syntax Summary

Method	Description	Syntax

ParentBusComp	Returns the parent business component.	`var myBusComp;`
		`var parentBusComp;`
		`parentBusComp = myBusComp.ParentBusComp();`

Pick	Places the currently selected record in a picklist business component into the appropriate fields of the parent business component.	`var myBusComp;` `myBusComp.Pick();`

PreviousRecord	Moves to the previous record in the business component.	`var myBusComp;`
		`var iReturn;`
		`iReturn = myBusComp.PreviousRecord();`

RefineQuery	Refines a query after a query has been executed.	`var myBusComp;` `myBusComp.RefineQuery();`
SetFieldValue	Assigns a new value to the named field for the current row of the business component.	`var myBusComp;` `myBusComp.SetFieldValue(`*`FieldName, FieldValue`*`);`
SetFormattedFieldValue	Accepts the field value in the current local format and assigns the new value to the named field for the current row of the business component.	`var myBusComp;` `myBusComp.SetFormattedFieldValue(`*`FieldName, FieldValue`*`);`
SetMultipleFieldValues	Assigns a new value to the fields specified in the property set for the current row of the business component.	`var myBusComp;` `myBusComp.SetMultipleFieldValues(oPropSet)`
SetNamedSearch	Sets a named search specification on the business component.	`var myBusComp;` `myBusComp.SetNamedSearch(`*`searchName, searchSpec`*`);`
SetSearchExpr	Sets the search specification for the business component.	`var myBusComp;` `myBusComp.SetSearchExpr(`*`searchSpec`*`);`
SetSearchSpec	Sets the search specification for the specified field.	`var myBusComp;` `myBusComp.SetSearchSpec(`*`FieldName, searchSpec`*`);`
SetSortSpec	Sets the sort specification for a query.	`var myBusComp;` `myBusComp.SetSortSpec(`*`sortSpec`*`);`
SetUserProperty	Sets the value of the specified User Property.	`var myBusComp;` `myBusComp.SetUserProperty(`*`propertyName, newValue`*`);`

Table 82. Business Component Methods Syntax Summary

Method	Description	Syntax
SetViewMode	Sets the visibility type for the business component.	`var myBusComp;` `myBusComp.SetViewMode(viewMode);`
UndoRecord	Deletes an active record created by NewRecord.	`var myBusComp;` `myBusComp.UndoRecord();`
WriteRecord	Commits to the database any changes made to the current record.	`var myBusComp;` `myBusComp.WriteRecord();`

Table 83 lists a summary of the Business Components Events syntax.

Table 83. Business Component Events Syntax Summary

Event	Description	Syntax
BusComp_Associate	Called after a record is added to a business component to create an association.	`BusComp_Associate();`
BusComp_ChangeRecord	Called after the current row changes in the business component.	`BusComp_ChangeRecord();`
BusComp_CopyRecord	Called after a new row is copied in the business component.	`BusComp_CopyRecord();`
BusComp_DeleteRecord	Called after a row is deleted in the business component.	`BusComp_DeleteRecord();`
BusComp_InvokeMethod	Called when the value of a business component field is accessed.	`BusComp_InvokeMethod(`*`methodName,methodArg1, methodArg2,..., methodArgn`*`);`
BusComp_NewRecord	Called after a new row has been created and made active in the business component.	`BusComp_NewRecord();`
BusComp_PreAssociate	Called before a record is added to a business component to create an association.	`BusComp_PreAssociate();`

Table 83. Business Component Events Syntax Summary

Event	Description	Syntax
BusComp_PreCopyRecord	Called before a new row is copied in the business component.	`BusComp_PreCopyRecord();`
BusComp_PreDeleteRecord	Called before a row is deleted in the business component.	`BusComp_PreDeleteRecord();`
BusComp_PreGetFieldValue	Called when the value of the business component field is accessed.	`BusComp_PreGetFieldValue(`*`FieldName, &FieldValue`*`);`
BusComp_PreInvokeMethod	Called before a specialized method is invoked on a business component.	`BusComp_PreInvokeMethod(`*`methodName`*`);`
BusComp_PreNewRecord	Called before a new row is created in the business component.	`BusComp_PreNewRecord();`
BusComp_PreQuery	Called before query execution.	`BusComp_PreQuery();`
BusComp_PreSetFieldValue	Called when a value is pushed down into the business component from the user interface.	`BusComp_PreSetFieldValue(`*`FieldName, FieldValue`*`);`
BusComp_PreWriteRecord	Called before a row is written out to the database.	`BusComp_PreWriteRecord();`
BusComp_Query	Called after the query is complete and every row has been retrieved, but before they have been displayed.	`BusComp_Query();`
BusComp_SetFieldValue	Called after a value has been pushed down into the business component from the user interface.	`BusComp_SetFieldValue(`*`FieldName`*`);`
BusComp_WriteRecord	Called after a row is written to the database.	`BusComp_WriteRecord();`

Business Object

Table 84 lists a summary of the Business Object Methods syntax.

Table 84. Business Object Methods Syntax Summary

Method	Description	Syntax

GetBusComp	Function that returns the specified business component.	`var myBusObject;`
		`var myBusComp;`
		`myBusComp = myBusObject.GetBusComp(`*`BusCompName`*`);`

Name	Function that returns the name of the control.	`var myBusObject as BusObject;`
		`var sName;`
		`sName = myBusObject.Name();`

Business Service

Table 85 lists a summary of the Business Service Methods syntax.

Table 85. Business Service Methods Syntax Summary

Method	Description	Syntax
GetFirstProperty	Retrieves the name of the first property of a business service.	`var oService;` `var sName;sName = oService.GetFirstProperty();`

GetNextProperty	Once the name of the first property has been retrieved, retrieves the name of the next property of a business service.	`var oService;`
		`var sName;`
		`sName = oService.GetNextProperty();`

GetProperty	Retrieves the value stored in the specified property.	`var oService;`
		`var sValue;`
		`sValue = oService.GetProperty(`*`propName`*`);`

Name	Returns the name of the business service.	`var oService;`
		`var sName;`
		`sName = oService.Name();`

InvokeMethod	Calls a specialized method or a user-created method on the business service.	`var oService;`
		`var Return;`
		`Return = oService.InvokeMethod(methodName,`
		`InputArguments, OutputArguments);`

Table 85. Business Service Methods Syntax Summary

Method	Description	Syntax
PropertyExists	Returns a Boolean value indicating whether the property specified in the argument exists.	`var oService;` `oService.PropertyExists(`*`propName`*`);`
RemoveProperty	Removes a property from a business service.	`var oService;` `oService.RemoveProperty(`*`propName`*`);`
SetProperty	Assigns a value to a property of a business service	`var oService;` `oService.SetProperty(`*`propName, propValue`*`);`

Table 86 lists a summary of the Business Service Events syntax.

Table 86. Business Service Events Syntax Summary

Method	Description	Syntax
Service_InvokeMethod	Called when a business service is accessed.	`Service_InvokeMethod(methodName);`
Service_PreCanInvokeMethod	Called before the PreInvokeMethod, allowing the developer to determine whether or not the user has the authority to invoke the business service method.	`Service_PreCanInvokeMethod (MethodName, &CanInvoke)`
Service_PreInvokeMethod	Called before a specialized method is invoked on a business service.	`Service_PreInvokeMethod(methodName, Inputs, Outputs);`

PropertySet

Table 87 lists a summary of the PropertySet Methods syntax.

Table 87. PropertySet Methods Syntax Summary

Method	Description	Syntax

AddChild	Adds subsidiary property sets to a property set.	`var oPropSet;`
		`var iIndex;`
		`oPropSet.AddChild(childObject);`

Copy	Returns a copy of a property set.	`var oPropSet1;`
		`var oPropSet2;`
		`oPropSet2 = oPropSet1.Copy();`

GetChild	Returns a specified child property set of a property set.	`var oPropSet;`
		`var sPropVal;`
		`sPropVal = oPropSet.GetChild(index);`

GetChildCount	Returns the number of child property sets attached to a parent property set.	`var oPropSet;`
		`var iCount;`
		`iCount = oPropSet.GetChildCount();`

GetFirstProperty	Returns the name of the first property in a property set.	`var oPropSet;`
		`var sPropName`
		`sPropName = oPropSet.GetFirstProperty()`

GetNextProperty	Returns the name of the next property in a property set.	`var oPropSet;`
		`var sPropName`
		`sPropName = oPropSet.GetNextProperty()`

GetProperty	Returns the value of a property when given the property name.	`var oPropSet;`
		`var sPropVal`
		`sPropVal = oPropSet.GetProperty(propName)`

GetPropertyCount	Returns the number of properties attached to a property set.	`var count;` `count = oPropSet.GetPropertyCount();`

GetType	Returns the value stored in a type in a property set.	`var oPropSet;`
		`var sTypeVal`
		`sTypeVal = oPropSet.GetType(value)`

GetValue	Returns a value stored as part of a property set.	`var oPropSet;`
		`var sValVal;`
		`sValVal = oPropSet.GetValue(value);`

InsertChildAt	Inserts a child property set into a parent property set at a specific location.	`var oPropSet;` `oPropSet.InsertChildAt(childObject, index);`

Table 87. PropertySet Methods Syntax Summary

Method	Description	Syntax
PropertyExists	Returns a Boolean value indicating whether the property specified in the argument exists.	`var oPropSet;` `oPropSet.PropertyExists(`*propName*`);`
RemoveChild	Removes a child property set as a specified index from a parent property set.	`var oPropSet;` `oPropSet.RemoveChild(`*index*`);`
RemoveProperty	Removes the property specified in its argument from a property set.	`var oPropSet;` `oPropSet.RemoveProperty(`*propName*`);`
Reset	Removes every property and child property set from a property set.	`var oPropSet;` `oPropSet.Reset();`
SetProperty	Assigns a value to the property of a property set specified in its argument.	`var oPropSet;` `oPropSet.SetProperty (`*propName* `,` *propValue*`)`
SetType	Assigns a data value to a type member of a property set.	`var oPropSet;` `oPropSet.SetType(`*value*`);`
SetValue	Assigns a data value to a value member of a property set.	`var oPropSet;` `oPropSet.SetValue(`*value*`);`

Miscellaneous Interfaces and Events

Table 88 lists a summary of the Miscellaneous Method syntax.

Table 88. Miscellaneous Method Syntax Summary

Method	Description	Syntax
TheApplication	Global method that returns the unique object of type Application.	`TheApplication().`*`Application_method;`*

Invoking Custom Methods with MiniButtons 14

Use the following procedure to invoke a custom method with a MiniButton.

Invoking Custom Methods

To invoke a custom method with a MiniButton

Be sure to set up Tools for the appropriate Target Browser Group.

1. Choose an applet (for example, Account List Applet) and create a control with the following properties:

   ```
   Name = ButtonTest
   Caption = Test
   HTML Type = MiniButton
   Method Invoked = MyTest
   ```

2. Right click the Applet and choose Edit Web Layout.

   The Web layout editor appears.

3. Change the template mode on the Web Controls toolbar to 3: Edit List.

   A window opens with the available controls, including the one you just created.

4. Drag and drop the control the ButtonTest control onto an available location. When you release the mouse button, the button appears.

5. Click Save and then choose File > Close.

6. To add a server script to the applet that enables the button, right-click the applet and choose Edit Server Scripts. Add the following script to the WebApplet_PreCanInvokeMethod() function:

```
function WebApplet_PreCanInvokeMethod (MethodName, &CanInvoke)
{
   if ( MethodName == "MyTest" )
   {
      CanInvoke = "TRUE";
      return( CancelOperation );
   }
   return (ContinueOperation);
}
```

7 Add the following browser script to the applet you are using (for example, the Account List Applet):

```
function Applet_PreInvokeMethod (name, inputPropSet)
{
   switch (name) {
      case "MyTest":
         alert( "Siebel 7 browser script!" );
         return("CancelOperation");
      break;
   }
   return ("ContinueOperation");
}
```

8 Run any application that has access to accounts, and go to the Accounts screen.

The new button should appear.

Index

A

ActivateField business component method, about 198
ActivateMultipleFields business component method, about 199
ActiveApplet application method, about 153
ActiveBusObject application method, about 153
ActiveMode applet method, about 133
ActiveViewName application method, about returning name of active view 154
ActiveX control, about using Login method 167
AddChild property set method, about 310, 311
allocations, about using TraceOn application method to track 185
applet
  ActiveApplet. about returning reference to currently focused applet 153
  object interface events, table of 129
applet events
  Applet_ChangeFieldValue, about 139
  Applet_ChangeRecord even, about 140
  Applet_InvokeMethod, about 140
  Applet_Load, about 141
  Applet_PreInvokeMethod, about 142
  WebApplet_InvokeMethod, about 142
  WebApplet_Load applet event 143
  WebApplet_Load, about 141
  WebApplet_PreCanInvokeMethod, about 144
  WebApplet_PreInvokeMethod, about 145
  WebApplet_ShowControl 145
  WebApplet_ShowListColumn, about 148
applet methods
  ActiveMode, about 133
  BusComp, about 134
  BusObject, about 134
  Find control, about 136
  FindActiveXControl, about 135
  InvokeMethod, about 136
  Name, about 138
  syntax summary (Browser Script), table of 424
  syntax summary (eScript), table of 437
Applet_ChangeFieldValue event, about 139
Applet_ChangeRecord event, about 140
Applet_InvokeMethod event, about 140
Applet_Load, about 141
Applet_PreInvokeMethod event, about 142
applets
  applet methods syntax summary (Browser Script), table of 424
  applet methods syntax summary (eScript), table of 437
  applet methods syntax summary (Siebel VB), table of 407
  Browser or Server script, adding to applet 64
  events, about and list of 111
  FindApplet, about returning applet identified by argument 157

object type, described 64
parent applet object, about returning for control 302
Webapplet events summary (Siebel VB), table of 408
WebApplet events summary, table of (eScript), table of 438
WebApplet events syntax summary (Browser Script), table of 425
application events
Application_Close event, about 189
Application_InvokeMethod, about 190
Application_Navigate, about 191
Application_PreNavigate, about 193
Application_Start, about 194
PreInvokeMethod, about 191
syntax summary, table of (eScript) 441
application events, about and list of 112
application extensions, running with 31
Application factory object
object type, described 65
Orbix, initializing and binding 88
Visibroker, initializing and binding 90
application methods
ActiveApplet, about 153
ActiveBusComp, about returning business component associated with 153
ActiveBusObject, about 153
ActiveViewName, about returning name of active view 154
Attach, about 155
CurrencyCode, about 155
Detach, about 156
EnableExceptions, about 156
FindApplet, about 157
GerBusObject, about 157
GetLastErrCode, about 159
GetLastErrText, about 160
GetProfileAttr, about 160
GetService, about 161
GetSharedGlobal, about 162
GotoView, about 163
InvokeMethod, about 164
LoadObjects, about 165
LoadUserAttributes, about using to load user profile 166
Login. about 167
LoginID, about 169
LoginName, about 170
Logoff, about 170
LookupMessage, about 171
Name, about 172
NewPropertySet, about 173
PositionID, about 173
PositionName, about 174
RaiseError, about 175
RaiseErrorText, about 176
Release, about 178
SetPositionID, about 178
SetPositionName, about 179
SetProfileAttr, about 179
SetSharedGlobal, about 180
syntax summary (COM data control), table 345
syntax summary (COM data server), table 357
syntax summary, table of (eScript) 439
Trace, about 182
TraceOff, about 184
TraceOn, about 185
application object type
described 62
unique object type, about using to return 329
application, table of object interface events 130
Application_Close event, about 189
Application_InvokeMethod application event, about 190
Application_Navigate application event, about 191
Application_PreNavigate application event, about 193
Application_Start application event, about 194

applications
application events syntax summary (eScript), table of 441
application methods summary (Siebel VB), table of 409
application methods syntax summary (COM data control), table 345
application methods syntax summary (COM data server), table 357
application methods syntax summary (eScript), table of 439
application methods syntax summary (mobile/dedicated Web client), table 369
CORBA interface SiebelApplication, table of 396
events summary (Siebel VB), table of 411
methods syntax summary (Browser Script), table of 425
arguments, syntax 19
association business component
Associate, about creating many-to-many relationship 201
BusComp_Associate, about calling after record added to create association 267
GetAssocBusComp, returning association business component 212
Attach application method, about 155
audience of guide 17

B

Basic program, verifying syntax 30
breakpoint, about and setting when debugging 40
Browser Script
See also Siebel Script Editor
about 26
about Browser Script 423
applet methods syntax summary, table 424
application methods syntax summary, table 425
business component methods syntax summary, table 427
business object methods syntax summary, table 428
business service events syntax summary, table 430
business service methods syntax summary, table 429
Control methods syntax summary, table 432
PropertySet methods syntax summary, table 430
WebApplet events syntax summary, table 425
Browser, adding to applet 64
BusComp
applet method, about 134
control method, about 302
ExecuteQuery, about return record using method 206
ExecuteQuery2, about returning records using method 209
object interface events, table of 130
BusComp_Associate business component event, about 267
BusComp_ChangeRecord business component event, about 267
BusComp_CopyRecord business component event, about 269
BusComp_DeleteRecord business component event, about 269
BusComp_InvokeMethod business component event, about 270
BusComp_NewRecord business component event, about 271
BusComp_PreAssociate business component event, about 271
BusComp_PreCopyRecord business component event, about 272
BusComp_PreDeleteRecord business component event, about 272

BusComp_PreGetFieldValue business component event, about 274
BusComp_PreInvokeMethod business component event, about 275
BusComp_PreNewRecord business component event, about 275
BusComp_PreQuery business component event, about 276
BusComp_PreSetFieldValue business component event, about 277
BusComp_PreWriteRecord business component event, about 279
BusComp_Query business component event, about 280
BusComp_SetFieldValue business component event, about 281
BusComp_WriteRecord business component event, about 282
business active application associated with 153
business component
  applet, associated with 134
  BusComp method, about returning for the control 302
  GetBusComp, about returning for a business component 283
  methods syntax summary, table of (eScript) 441
  name property, returning 234
business component events
  BusCom_PreCopyRecord, about 272
  BusComp_Associate, about 267
  BusComp_ChangeRecord, about 267
  BusComp_CopyRecord, about 269
  BusComp_DeleteRecord, about 269
  BusComp_InvokeMethod, about 270
  BusComp_NewRecod, about 271
  BusComp_PreAssociate, about 271
  BusComp_PreDeleteRecord, about 272
  BusComp_PreGetFieldValue, about 274
  BusComp_PreInvokeMethod, about 275
  BusComp_PreNewRecord, about 275
  BusComp_PreQuery, about 276
  BusComp_PreSetFieldValue, about 277
  BusComp_PreWriteRecord, about 279
  BusComp_Query, about 280
  BusComp_SetFieldValue, about 281
  BusComp_WriteRecord, about 282
  syntax summary, table of (eScript) 445
business component methods
  ActivateField, about 198
  ActivateMultipleFields, about 199
  Associate, about 201
  BusObject, about 202
  ClearToQuery, about 203
  DeactivateFields, about 204
  DeleteRecord, about 205
  ExecuteQuery, about 206
  ExecuteQuery2, about 209
  FirstRecord, about 210
  FirstSelected, about 211
  GetAssocBusComp, about 212
  GetFieldValue, about 214
  GetFormattedFieldValue, about 216
  GetLasErrCode, about 219
  GetLastErrText, about 219
  GetMultipleFieldValues, about 220
  GetMVGBusComp, about 220
  GetNamedSearch, about 222
  GetPicklistBusComp, about 222
  GetSearchExpr, about 224
  GetSearchSpec, about 225
  GetUserProperty, about 225
  GetViewMode, about 226
  InvokeMethod, about 228
  LastRecord, about 233
  Name, about 234
  NewRecord, about 234
  NextRecord, about 236
  NextSelected, about 237
  ParentBusComp, about 237
  Pick, about 238
  PreviousRecord, about 239
  RefineQuery, about 241
  Release, about 241
  SearchExpr, about 242

SetFieldValue, about 243
SetFormattedFieldValue, about 245
SetMultipleFieldValues, about 247
SetNamedSearch, about 248
SetSearchExpr, about 249
SetSearchSpec, about 251
SetSortSpec, about 255
SetUserProperty, about 257
SetViewMode, about 258
syntax summary (COM data control), table 348
syntax summary (COM data server), table 360
UndoRecord, about 262
ViewMode, about 262
WriteRecord, about 263
business components
about 94
BusComp object, logical flow of instantiating 96
business component events summary (Siebel VB), table of 416
business component events syntax summary (eScript), table of 445
business component methods syntax summary (COM data control), table 348
business component methods syntax summary (COM data server), table 360
business component methods syntax summary (eScript), table of 441
business component methods syntax summary (Siebel VB), table of 412
business rules, adding to 30
CORBA interface SiebelBusComp, table of 398
database, committing records to 94
methods for accessing, list of 97
methods syntax summary (Browser Script), table of 427
methods syntax summary (mobile/dedicated Web client), table 372
object type, described 63
records, adding and inserting 94
scenarios 95
SiebelBusComp methods syntax summary (Java), table of 387
business object methods
GetBusComp, about 283
GetLastErrCode, about 284
GetLastErrText, about 285
Name, about 285
Release, about 286
syntax summary (COM data control), table 352
syntax summary (COM data server), table 364
table of 125
business objects
active applet, about returning for business component 153
business object methods syntax summary (COM data control), table 352
business object methods syntax summary (COM data server), table 364
business object methods syntax summary (eScript), table of 447
business object methods syntax summary (Siebel VB), table of 418
BusObject, about returning business object for applet 134
BusObject, about returning business object that contains business component 202
CORBA interface SiebelBusObjects, table of 401
methods syntax summary (Browser Script), table of 428
methods syntax summary (mobile/dedicated Web client), table 376
Name, about using to return name of business object 285
object type, described 62

business rules
business component, adding to 30
described 27
business service
object interface events, table of 131
object interface methods, table of 125
business service events
Service_InvokeMethod, about 295
Service_PreCanInvokeMethod, about 298
Service_PreInvokeMethod, about 298
syntax summary, table of (eScript) 448
business service methods
GetFirstProperty, about 287
GetLasErrCode, about 288
GetLasErrText, about 288
GetNextProperty, about 289
GetProperty, about 290
InvokeMethod, about 290
Name, about 292
PropertyExists, about 293
Release, about 293
RemoveProperty, about 294
SetProperty, about 294
syntax summary (COM data control), table 352, 353
syntax summary (COM data server), table 364
syntax summary, table of (eScript) 447
business services
business service events syntax summary (eScript), table of 448
business service events syntax summary (Siebel VB), table of 419
business service methods syntax summary (COM data control), table 352, 353
business service methods syntax summary (COM data server), table 364
business service methods syntax summary (eScript), table of 447
business service methods syntax summary (Siebel VB), table of 418
CORBA interface SiebelService, table of 402
events syntax summary (Browser Script), table of 430
methods syntax summary (Browser Script), table of 429
methods syntax summary (mobile/dedicated Web client), table 377
retrieving property names 289
SetProperty, about assigning values to members of 294
SiebelService methods syntax summary (Java), table of 391
business services object type, described 63
BusObject
applet method, about 134
business component method, about 202

C

C++
Siebel COM Server, building in 331
Siebel COM Server, testing program 344
Calls window, about and accessing 40
ChangeFieldValue, about 139
ChangeRecord event, about 140
ClearToQuery business component method, about 203
coding, caution, about and using Siebel Tools 25
COM data control
application methods syntax summary (table) 345
business component methods syntax summary (table) 348
business object methods syntax summary (table) 352
business service methods syntax summary (table) 352, 353
installation, about 53

property set methods syntax summary (table) 353
COM data server
application methods syntax summary (table) 357
business component methods syntax summary (table) 360
business object methods syntax summary (table) 364
business service methods syntax summary (table) 364
installation, about 53
interface method, about COM error handling 115
LoadObjects method, about using to start object and return reference 165
property set methods syntax summary (table) 365
COM error handling, about and methods 115
COM interfaces
Siebel COM client in C++, building 331
Siebel COM client in C++, testing program 344
comparison operators, using in search expressions 251
connect string
about, syntax, and example 112
Siebel Server, substitutions when logging into (table) 113
constants, table of 132
control
FindControl, about argument specified in 136
GetValue, about returning value of control 303
object interface methods, table of 126
SetValue, about using to set the contents of the control 309
control methods
Applet method, about returning parent applet object 302
BusComp, about 302
GetProperty, about 303
GetValue, about returning control value 303
Name, about returning object name 304
SetProperty, about 305, 307
SetValue, about using to set contents of the control 309
syntax summary, table of (Browser Script), table of 432
controls
GetProperty, assigning values to properties 303
SetProperty, assigning values to properties 305, 307
conventions, used in manual 19
Copy property set method, about 311
copying records, using NewRecord method 235
CORBA. See *individual Siebel CORBA entries*
CreateSiebelAppObject SiebelAppFactory method, about 325
CurrencyCode application method, about 155
custom method, invoking with a MiniButton 451

D

data bean, table of SiebelDataBean methods syntax summary (Java), table of 385
data value
SetProperty, about using to assign value to 323
SetType, about using to assign data value of type to property set 324
database, about using WriteRecord to commit to database 263
DeactivateFields business component method, about 204
deallocations, using TraceOn application method to track 185
debug tracings methods, table of 104

debugger. *See* Siebel Debugger
DeleteRecord business component method, about 205
desc, about 405
Detach application method, about 156
Document Object Model (DOM). *See* Browser Script

E

EnableExceptions application method, about 156
error code
  application methods, about using GetLastErrCode to return last error code 159
  business component methods, about using GetLastErrCode to return most recent 219
  business object methods, about using GetLastErrCode to return last error code 284
  business service methods, about using GetLastErrCode to return most recent 288
  GetErrorCode, about using with Java Data Bean to display numeric code 328
error handling
  See also *individual Siebel object interface entries*
  COM error handling, about and examples 115
  CORBA exception handling 117
  error message tracking 116
  native COM error handling, enabling and disabling 156
error messages
  function_name Is An Unknown Function, about and correcting 43
  GetErrorMessage, about using with Java Data Bean to display message 328
  GetLastErrText, about returning last text error message 160
error text messages
  business component methods, about using GetLastErrText 219
  business object methods, about using GetLastErrText 285
  business service methods, about using GetLastErrText 288
event method syntax 104
events, object interface events, table of 129
exception handling, CORBA 117
ExecuteQuery business component method, about 206
ExecuteQuery2 business component method, about 209
exposed object types, table of 66
external applications
  logging in 167
  using CORBA interfaces 59

F

field value, method of retuning in the current local format 216
FindActiveXControl applet method, about 135
FindApplet application method, about 157
FindControl applet method, about 136
FirstRecord business component method, about 210
FirstSelected business component method, about 211

G

GetAssocBusComp business component method, about 212
GetBusComp business object method, about 283
GetBusObject application method, about 157
GetChild property set method, about 313
GetChildCount property set method, about 314

GetErrorCode method, about 328
GetErrorMessage method, about using to display error messages 328
GetFieldValue business component method, about 214
GetFirstProperty
business service methods, about 287
property set methods, about 315
GetFormattedFieldValue business component method, about 216
GetLastErrCode
application methods, about 159
business component methods, about 219
business object methods, about 284
business service methods, about 288
GetLastErrText
application methods, about 160
business object methods, about 285
business service methods, about 288
note, about availability to interfaces 45
GetLastErrText business component method, about 219
GetMultipleFieldValues business component method, about 220
GetMVGBusComp business component method, about 220
GetNamedSearch business component method, about 222
GetNextProperty
business service methods, about 289
property set methods, about 316
GetPicklistBusComp business component method, about 222
GetProfileAttr application method, about 160
GetProperty
business service methods, about 290
control methods, about 303
property set methods, about 317
GetPropertyCount property set method, about 318
GetSearchExpr business component method, about 224
GetSearchSpec business component method, about 225
GetService application method, about 161
GetSharedGlobal application method, about 162
GetType property set method 319
GetUserProperty business component method, about 225
GetValue
control methods, about 303
property set methods, about 319
GetViewMode business component method, about 226
global state, properties and functions 100
global variables
about and VB example 103
GetSharedGlobal application method, about 162
GotoView application method, about 163
guide
audience 17
organization of 18

H

High Interactivity mode, about running Browser scripts 423

I

Id, about using to represent error numeric value 406
InsertChildAt property set method, about 320
installation procedures, object interfaces 53
inter-application variable methods, table of 103
interface methods, table grouped by object interface type 120

InvokeMethod
applet methods, about 136
Applet_InvokeMethod, about 140
application methods, about 164
business component methods, about 228
business service methods, about 290
WeApplet_InvokeMethod, about 142

J

java Bean. See *individual Siebel Java entries*
Java Cryptography Extension (JCE), enabling 87
Java Data Bean
GetErrorCode, about using to display numeric error codes 328
GetErrorMessage, about using to display error messages 328
JavaScript. *See* Siebel eScript
JCE (Java Cryptography Extension), enabling 87

L

LastRecord business component method, about 233
Load event
Applet_Load, about triggering after applet is loaded 141
WebApplet_Load event, about triggering just after applet is loaded 143
LoadObjects application method, about 165
LoadUserAttributes application method, about 166
local variables, described and VB example 101
locating objects method, about and list of methods 93
logical operators in search expressions 251
Login method application method, about 167
LoginId application method, about 169
LoginName application method, about 170
Logoff application method, about 170
LookupMessage application method, about 171

M

methods
custom method, invoking with a MiniButton 451
table grouped by interface type 120
Microsoft Foundation Class (MFC) library. *See* Siebel COM Data Server
Microsoft Visual Basic
Siebel COM Data Control Interface, setting up to access 78
Siebel COM Data Server, setting up to access 75
Siebel Mobile/Dedicated Web Client Automation Server, setting up to access 73
Siebel Web Client Automation Server, setting up to access 71
MiniButton, using to invoke custom method 451
mobile Web client automation server, about installation 53
mobile/dedicated Web client
application methods syntax summary, table of 369
business component methods syntax summary, table of 372
business object methods syntax summary, table of 376
business service methods syntax summary, table of 377
property set methods syntax summary, table of 378
module variables, about and VB example 101
MVG business component, returning 220

N

Name
  applet method, about 138
  application method, about 172
  business component method, about 234
  business object method, about 285
  business service method, about 292
  control method, about 304
named field value, about using SetFieldValue to assign new value to 243
navigation methods, object interfaces 99
NewPropertySet application method, about 173
NewRecord business component method, about 234
NextRecord business component method, about 236
NextSelected business component method, about 237

O

object interface events
  applet, table of 129
  application, table of 130
  BusComp, table of 130
  business service, table of 131
object interface methods tables
  applet, table of 120
  application, table of 121
  business component, table of 123
  business object, table of 125
  business service, table of 125
  control, table of 126
  miscellaneous methods and events, table of 128
  property set, table of 127
  Siebel Exceptions, table of 128
  SiebelApp Factory, table of 128
object interfaces
  component of Siebel programming environment described 27
  Siebel object interfaces, about 51
object types
  applet object type, described 64
  application factory object type, described 65
  application, described 62
  business component, described 63
  business object, described 62
  business service, described 63
  property set, described 65
  Script flag, about 34
  Siebel Object Interface, object types, table of 66
object, about using Name method to return object name 304
operating currency code, returning 155
Orbix, initializing and binding Siebel Application factory object 88
organization of guide 18

P

ParentBusComp business component method, about 237
Pick business component method
  GetPicklistBusComp, returns component 222
  Pick method, about 238
PositionId application method, about 173
PositionName application method, about 174
PreCanInvokeMethod, about WebApplet_PreCanInvokeMethod 14 4
preferences
  debugging preferences, accessing and settings 37
  run-time preferences, accessing and settings 38
  Siebel Script Editor, accessing and window features 34

PreInvokeMethod
Applet_PreInvokeMethod, about 142
Application_PreInvokeMethod, about 191
WebApplet_PreInvokeMethod, about 145
PreviousRecord business component method, about 239
programming
custom extension routines, about extending data validation 27
environment, component of 25
languages, about 25
scripting language, choosing 29
user interface components, about customizing behavior 28
properties of controls
GetProperty, about assigning 303
SetProperty, about assigning visual properties 305, 307
property set methods
AddChild, about adding subsidiary property set to 310
Copy, about returning copy of set 311
GetChild, about returning child property of property set 313
GetChildCount, about returning child property sets attached to 314
GetFirstProperty, about returning name of first property 315
GetNextProperty, about returning next property 316
GetProperty, about returning property value when given name 317
GetPropertyCount, about returning number of properties attached to 318
GetValue, about retrieving data value 319
InsertChildAt, about inserting child property set into parent property 320
object interface methods, table of 127
RemoveChild, about removing child property set from parent property set 321
RemoveProperty, about removing a property from property set 322
SetProperty, about assigning a data value to property 323
SetType, about assigning data value of type 324
syntax summary (COM data control), table 353
syntax summary (COM data server), table 365
syntax summary table (eScript) 449
property set object type, described 65
property sets
business service methods syntax summary (COM data control), table 353
business service methods syntax summary (COM data server), table 365
Copy, about returning copy of 311
CORBA interface SiebelService, table of 403
GetChild, about retrieving child property set 313
GetFirstProperty, about retrieving property names 315
GetNextProperty, about retrieving property names 316
GetProperty, about retrieving property values 317
GetPropertyCount, about retrieving values of type members 318
GetType, about retrieving values of type members 319
GetValue, about retrieving value values 319
InsertChildAt, about adding subsidiary 320
methods syntax summary (mobile/dedicated Web client), table 378

property set methods syntax summary (eScript), table of 449
property set methods syntax summary (Siebel VB), table of 420
RemoveChild, about removing child property set 321
RemoveProperty, about removing properties of 322
Reset, about removing properties and child properties 323
SetProperty, about assigning values to members of 323
SetType, about assigning values to type members 324
SetValue, about assigning values to value member 325
SiebelPropertySet methods syntax summary (Java), table of 392
tree-structured data structures, for 311
PropertyExists
business service method, about 293
property set method, about retuning Boolean value 321
PropertySet
methods syntax summary (Browser Script), table of 430
methods syntax summary (Siebel Web client), table of 382

Q

queries
ClearToQuery, about using to clear query 203
RefineQuery, about using to define after execution 241
SetSortSpec, about using to set sort specification 255
quotation marks, about using in search expressions 252

R

RaiseError application method, about 175
RaiseErrorText application method, about 176
records
LastRecord, about using to move to 233
NewRecord, about adding a new record (row) 234
NextSelected, about using to move focus to next record 237
Pick, about placing record in a picklist 238
PreviousRecord, about moving to previous record 239
UndoRecord, about using to reverse uncommitted changes 262
WriteRecord, about committing database changes 263
RefineQuery business component method, about 241
Release
application method, about 178
business component method, about 241
business object method, about 286
business service method, about 293
SiebelAppFactory method, about 327
RemoveChild property set method, about 321
RemoveProperty
business service method, about 294
property set method, about 322
Reset property set method, about removing properties and child property sets 323
Run-time Engine, invoking 43

S

scripts
language, choosing 29
syntax, checking 39
search expression
GetSearchExpr, about using to return current search expression 224

SetSearchExpr, about setting on entire search expression 249
search specification
Field name argument, about returning for field specified in 225
SearchExp, about setting and getting specification for business component 242
searchName, returns named search specification 222
SetNamedSearch, about setting a named search specification on the business component 248
SetSearchSpec, about setting for a particular field 251
SetSearchSpec, about setting for particular field 251
SearchExpr, about business component method 242
Server Script, components 25
server, about Logoff method 170
Service_InvokeMethod business service event, about 295
Service_PreCanInvokeMethod business service event, about 298
Service_PreInvokeMethod business service event, about 298
SetFieldValue business component method, about 243
SetFormattedFieldValue business component method, about 245
SetMultipleFieldValues business component method, about 247
SetNamedSearch business component method, about 248
SetPositionID application method, about 178
SetPositionName application method, about 179
SetProfileAttr application method, about 179
SetProperty
business service method, about assigning 294
control property, about returning value of 303
control, about setting visual properties 305, 307
property set method, about assigning data value to 323
SetSearchExpr business component method, about 249
SetSearchSpec business component method, about 251
SetSharedGlobal application method, about 180
SetSortSpec business component method, about 255
SetType property set method, about 324
SetUserProperty business component method, about 257
SetValue
control, about using to set contents of 309
property set, about assigning data value to 325
SetViewMode business component method, about 258
Siebel Application factory
object, about 65
Orbix, initializing and binding 88
Visibroker, initializing and binding 90
Siebel business components, about events and list of 109
Siebel COM Data Control
about and diagram 54
instantiating 78
Siebel COM Data Server
about and diagram 56
building in C++ 331
C++, testing program 344
instantiating 75

Siebel COM interfaces
accessing 70
COM Data Control interfaces, about and diagram 54
COM Data Server, about and diagram 56
COM error handling 115
Siebel Mobile/Dedicated Web Client Automation Server, about and diagram 58
Siebel Web Client Automation Server, about and diagram 56
Siebel Compiler
compiler/interpreter described 26
invoking 43
order considerations and error message 43
Siebel constants table 132
Siebel CORBA
CORBA exception handling, about and code example 117
CORBA interface SiebelAppFactory, table of 405
CORBA interface SiebelApplication, table of 396
CORBA interface SiebelBusComp, table of 398
CORBA interface SiebelBusObjects, table of 401
CORBA interface SiebelService, table of 402
CORBA Object Manager, about installation 53
object interface method, creating Siebel application object in CORBA 100
Object Manager (SCOM), about 60
Object Manager Login method, about 167
object reference, returns for an Application object 325
property sets, table of 403
Siebel CORBA interfaces
accessing 88
business objects, accessing 59
objects, about 59
Siebel Debugger
about using 35
accessing 36
breakpoint, about and setting when debugging 40
Debug toolbar, diagram and description of 36
debugging preferences, accessing and settings 37
run-time preferences, accessing and settings 38
script syntax, checking 39
subroutines and function calls, displaying 40
variable window, about 40
Siebel Dedicated Automation Server
installation, about 53
Siebel eScript
See also Siebel Script Editor
about 25
applet methods, syntax summary (table) 437
application events syntax summary, table of 441
application methods syntax summary, table of 439
business component events syntax summary, table of 445
business component methods syntax summary, table of 441
business object methods syntax summary, table of 447
business service events syntax summary, table of 448
business service methods syntax summary, table of 447
objects, destroying and example 50
property set methods syntax summary, table of 449
script syntax, checking 39
Siebel VB, differences between 47

Switch construct, making effective use of 49
syntax conventions 68
syntax, verifying 30
theApplication, method syntax summary, table of 450
this object reference, about using and example 48
variables, declaring 48
WebApplet event summary, table of 438
with shortcut, about and example 49
Siebel eScript language, about 26
Siebel Exceptions, table of object interface methods 128
Siebel extension events
applet events, about and list of 111
applications events, about and list of 112
events occur, determining when 109
method syntax 104
program flow, process affected by script 106
Siebel business component events, about and list of 109
Siebel interface objects
Siebel application object in CORBA, creating 100
Siebel methods and events, about accessing from scripts 61
usage evaluation matrix, table 61
Siebel Java Bean
codepage support (table) 84
data Bean, about installation 53
JDB and Siebel Server, encrypting communication between 86
SiebelBusComp methods syntax summary, table of 387
SiebelDataBean methods syntax summary, table of 385
SiebelExceptions methods syntax summary, table of 394
SiebelPropetySet methods syntax summary, table of 392
SiebelService methods syntax summary, table of 391
supported platforms, and instantiating 81
Siebel Java interfaces
multiple threads, using with 59
object, about using to access 58
Siebel Mobile/Dedicated Web Client Automation Server
about and diagram 58
accessing 72
Siebel object interface
See also error handling
Application factory object (Orbix), initializing and binding 88
Application Factory object (Visibroker), initializing and binding 90
interface installations, about 53
Java Data Bean, about, supported platforms, and instantiating 81
Siebel COM Data Control, instantiating 78
Siebel COM Data Server, instantiating 75
Siebel COM interfaces, accessing method 54
Siebel CORBA interfaces, about accessing objects 59
Siebel Java interfaces 58
Siebel Object Interface method
examples 68
syntax 66
Siebel object interface, events
See also *individual Siebel object interface entries*
applet events, about and list of 111
application events, about and list of 112
events occur, determining when 109
method syntax 104
program flow, process affected by script 106
Siebel business component events, about and list of 109

Siebel object interface, getting started
See also *individual Siebel object interface entries*
connect string, about, syntax, and example 112
connect string, substitutions when logging into a Siebel Server (table) 113
Siebel COM Data Control, accessing and screen example 78
Siebel COM interfaces, accessing 70
Siebel CORBA interfaces, accessing 88
Siebel mobile/dedicated Web client automation server, accessing 72
Siebel Web Client Automation Server, accessing 71
Siebel object interface, methods
See also *individual Siebel object interface entries*
business components, accessing 94
global state properties and functions 100
list of 92
locating objects, about and list of methods 93
navigation methods 99
user interaction, about and methods 99
Siebel Object Model. *See* Browser Script
Siebel programming
constants, table of 132
custom extension routines, about extending data validation 27
environment, components of 25
scripting language, choosing 29
user interface components, about customizing behavior 28
Siebel script
debug tracing methods, table of 104
global variables, about and VB example 103
inter-application communication methods, list of 103
local variables. about and VB example 101
module variables, about and VB example 101
Siebel Script Editor
about 26
accessing and screen example 32
custom programs, tips about creating 30
menu options, list of 33
preferences, accessing and window features 34
Scripted flag, about 34
Siebel Server
applet, adding to 64
JDB and Siebel Server, encrypting between 86
Siebel session ID, about returning string containing Id 156
Siebel VB
See also Siebel Script Editor
about 25
applet methods syntax summary, table of 407
application events summary, table of 411
application methods syntax summary, table of 409
business component methods syntax summary, table of 412
business components events summary, table of 416
business object methods syntax summary, table of 418
business service events syntax summary, table of 419
business service methods syntax summary, table of 418
components of 25
getting started 43
picklist, picking a value from 223
property set methods syntax summary, table of 420
script syntax, checking 39

Siebel eScript, differences between 47
syntax conventions 68
theApplication method, syntax summary 422
Webapplet events, summary (table) 408
Siebel VB language, about 26
Siebel VB, getting started
date variables, about working with 46
Me object reference, about using and example 44
naming conventions, about using standardized 44
objects, destroying and example 47
run-time errors, about trapping 45
Select Case, making effective use of 45
variables, declaring 44
With shortcut, using and example 46
Siebel Web client
PropertySet methods syntax summary, table of 382
Siebel Service methods syntax summary, table of 382
SiebelHTMLApplication methods syntax summary, table of 381
Siebel Web Client Automation Server
about and diagram 56
accessing 71
installation, about 53
SiebelAppFactory methods
CORBA interface SiebelAppFactory, table of 405
CreateSiebelAppObject, about 325
object interface methods, table of 128
Release, about 327
SiebelBusComp methods syntax summary (Java), table of 387
SiebelDataBean methods syntax summary (Java), table of 385
SiebelException methods
desc, about 405
syntax summary (Java), table of 394
SiebelHTMLApplication methods syntax summary, table of 381
SiebelPropertySet methods syntax summary (Java), table of 392
SiebelService methods
syntax summary (Java), table of 391
syntax summary (Siebel Web client), table of 382
sorting specification, setting 255
special characters, using in search expressions 252
specialized methods, calling 228
subsidiary property sets, about using AddChild to add to a property set 310
syntax checking 39

T

theApplication method
object type, about using to return 329
syntax summary (eScript), table of 450
syntax summary (Siebel VB) 422
Trace application method, about 182
TraceOff application method
about 184
debug tracing, about 104
TraceOn application method
about 185
debug tracing, about 104
tree-structured data structures, creating using property sets 311
typographic conventions, used in manual 19

U

UndoRecord business component method, about 262
user interaction, object interface methods 99
user interface control object type 65
user property value
GetUserProperty, about using to return value 225

SetUserProperty, about using to set the value of named business user property 257
user-created methods, calling 228

V

value, about returning value of control 303
variables, about 40
ViewMode business component method, about 262
visibility mode, about returning current visibility mode 226
visibility type
SetViewMode, about setting for business component 258
ViewMode, about using to set and get for business component 262
Visibroker. initializing and binding Siebel Application factory object 90

W

Web Client Automation Server, enabling 57
WebApplet events
summary, table of (eScript) 438
syntax summary, table of (Browser Script) 425
WebApplet_InvokeMethod event, about 141
WebApplet_Load, about 143
WebApplet_PreInvokeMethod event, about 144, 145
WebApplet_ShowControl event, about 145
WebApplet_ShowListColumn, about 148
WriteRecord business component method, about 263