Copyright © 2005, 2008, Oracle. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

PRODUCT MODULES AND OPTIONS. This guide 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 guide. To find out more about the modules your organization has purchased, see your corporate purchasing agent or your Oracle sales representative.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS. Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.
Chapter 1: What’s New in This Release

Chapter 2: Siebel Programming Tools
About the Siebel Programming Tools 19
Components of the Siebel Programming Environment 20
Supported Uses of Siebel Programming Languages 21
  Business Logic Definition 21
  Custom Behavior for User Interface Components 21
Adding New Business Logic to a Business Component 22
Tracing Scripts 22
About the Siebel Compiler and Run-Time Engine 24
About Siebel VB 25
About Siebel eScript 27

Chapter 3: Programming
About Programming with Siebel Object Interfaces 31
About Siebel Object Interfaces 32
  Siebel COM Interfaces 32
  Siebel Java Interfaces 36
  Built-In Scripting 36
  Usage Evaluation Matrix 37
Installing Siebel Object Interfaces 37
Exposed Object Types 38
  Application Object Type 38
  Business Object Object Type 38
  Business Component Object Type 39
  Business Service Object Type 39
  Applet Object Type 40
  Property Set Object Type 40
  User Interface Control Object Type 40
  Summary of Exposed Object Types 41
Object Interface Event Tables 93
  Applet Events 93
  Application Events 94
  Business Component Events 94
  Business Service Events 95

Siebel Constants 95

Applet Methods 96
  ActiveMode Method 96
  BusComp Method 97
  BusObject Method 98
  FindActiveXControl Method 98
  FindControl Method 99
  InvokeMethod Method 100
  Name Method 102

Applet Events 103
  Applet_ChangeFieldValue Event 103
  Applet_ChangeRecord Event 105
  Applet_InvokeMethod Event 106
  Applet_Load Event 107
  Applet_PreInvokeMethod Event 108
  WebApplet_InvokeMethod Event 109
  WebApplet_Load Event 110
  WebApplet_PreCanInvokeMethod Event 112
  WebApplet_PreInvokeMethod Event 113
  WebApplet_ShowControl Event 114
  WebApplet_ShowListColumn Event 116

Application Methods 119
  ActiveApplet Method 120
  ActiveBusComp Method 121
  ActiveBusObject Method 122
  ActiveViewName Method 124
  Attach Method 125
  CurrencyCode Method 127
  Detach Method 128
  EnableExceptions Method 129
  FindApplet Method 131
  GetBusObject Method 131
  GetLastErrCode Method 133
  GetLastErrText Method 134
  GetProfileAttr Method 135
  GetService Method 135
Contents

GetSharedGlobal Method 138
GotoView Method 139
InvokeMethod Method 142
LoadObjects Method 146
LoadUserAttributes Method 148
Login Method 148
LoginId Method 150
LoginName Method 151
Logoff Method 152
LookupMessage Method 153
Name Method 154
NewPropertySet Method 154
PositionId Method 156
PositionName Method 157
RaiseError Method 158
RaiseErrorText Method 159
SetPositionId Method 161
SetPositionName Method 162
SetProfileAttr Method 162
SetSharedGlobal Method 164
ShowModalDialog Method 166
SWEAlert Method 168
Trace Method 169
TraceOff Method 171
TraceOn Method 172

Application Events 175
  Application_Close Event 176
  Application_InvokeMethod Event 176
  Application_Navigate Event 177
  Application_PreInvokeMethod Event 177
  Application_PreNavigate Event 179
  Application_Start Event 180

Business Component Methods 181
  ActivateField Method 183
  ActivateMultipleFields Method 184
  Associate Method 186
  BusObject Method 188
  ClearToQuery Method 189
  CountRecords Method 190
  DeactivateFields Method 191
  DeleteRecord Method 193
  ExecuteQuery Method 193
Contents

ExecuteQuery2 Method 195
FirstRecord Method 196
FirstSelected Method 198
GetAssocBusComp Method 200
GetFieldValue Method 201
GetFormattedFieldVal Method 203
GetLastErrCode Method 205
GetLastErrText Method 206
GetMultipleFieldValues Method 206
GetMVGBusComp Method 207
GetNamedSearch Method 208
GetPicklistBusComp Method 209
GetSearchExpr Method 211
GetSearchSpec Method 212
GetSortSpec Method 212
GetUserProperty Method 213
GetViewMode Method 214
InvokeMethod Method 215
LastRecord Method 224
Name Method 225
NewRecord Method 225
NextRecord Method 227
NextSelected Method 228
ParentBusComp Method 228
Pick Method 229
PreviousRecord Method 231
RefineQuery Method 232
Release Method 233
SetFieldValue Method 235
SetFormattedFieldVal Method 237
SetMultipleFieldValues Method 238
SetNamedSearch Method 240
SetSearchExpr Method 242
SetSearchSpec Method 244
SetSortSpec Method 248
SetUserProperty Method 250
SetViewMode Method 251
UndoRecord Method 254
WriteRecord Method 255

Business Component Events 256
BusComp_Associate Event 257
BusComp_ChangeRecord Event 258
BusComp_CopyRecord Event 259
BusComp_DeleteRecord Event 260
BusComp_InvokeMethod Event 261
BusComp_NewRecord Event 261
BusComp_PreAssociate Event 262
BusComp_PreCopyRecord Event 262
BusComp_PreDeleteRecord Event 263
BusComp_PreGetFieldValue Event 264
BusComp_PreInvokeMethod Event 265
BusComp_PreNewRecord Event 266
BusComp_PreQuery Event 266
BusComp_PreSetFieldValue Event 267
BusComp_PreWriteRecord Event 269
BusComp_Query Event 270
BusComp_SetFieldValue Event 272
BusComp_WriteRecord Event 272

Business Object Methods 273
  GetBusComp Method 273
  GetLastErrCode Method 275
  GetLastErrText Method 275
  Name Method 276
  Release Method 276

Business Service Methods 278
  GetFirstProperty Method 278
  GetNextProperty Method 280
  GetProperty Method 281
  InvokeMethod Method 282
  Name Method 283
  PropertyExists Method 284
  Release Method 284
  RemoveProperty Method 286
  SetProperty Method 287

Business Service Events 287
  Service_InvokeMethod Event 287
  Service_PreCanInvokeMethod Event 289
  Service_PreInvokeMethod Event 290

Control Methods 293
  Applet Method 294
  BusComp Method 294
  GetProperty Method 295
  GetValue Method 295
Contents

Name Method 296
setLabelProperty Method 297
setProperty Method 299
setValue Method 300

Property Set Methods 302
addChild Method 303
Copy Method 304
getBytesValue Method 305
getChild Method 306
getChildCount Method 307
getFirstProperty Method 308
getLastErrCode Method 309
getLastErrText Method 310
getNextProperty Method 310
getProperty Method 311
getPropertyCount Method 312
getType Method 312
getValue Method 313
insertChildAt Method 314
propertyExists Method 314
removeChild Method 315
removeProperty Method 316
reset Method 316
setByteValue Method 317
setProperty Method 318
setType Method 319
setValue Method 319

Miscellaneous Methods 320
getErrorCode Method 320
getErrorMessage Method 321
TheApplication Method 322

Chapter 5: Accessing Siebel COM Data Server with C++
Building the Siebel COM Client in C++ 325
Testing Your Program 329

Chapter 6: COM Data Control Quick Reference
Application Methods for COM Data Control 331
Business Component Methods for COM Data Control 334
Business Object Methods for COM Data Control 338
### Business Service Methods for COM Data Control
338
### Property Set Methods for COM Data Control
339

#### Chapter 7: COM Data Server Quick Reference

- Application Methods for COM Data Server 343
- Business Component Methods for COM Data Server 346
- Business Object Methods for COM Data Server 350
- Business Service Methods for COM Data Server 351
- Property Set Methods for COM Data Server 352

#### Chapter 8: Mobile Web Client Automation Server Quick Reference

- Application Methods for Mobile Web Client Automation Server 355
- Business Component Methods for Mobile Web Client Automation Server 358
- Business Service Methods for Mobile Web Client Automation Server 363
- Property Set Methods for Mobile Web Client Automation Server 364

#### Chapter 9: Java Data Bean Quick Reference

- Data Bean Methods for Java Data Bean 367
- Business Component Methods for Java Data Bean 369
- Business Object Methods for Java Data Bean 373
- Business Service Methods for Java Data Bean 373
- Property Set Methods for Java Data Bean 374
- SiebelException Methods for Java Data Bean 376

#### Chapter 10: Siebel Web Client Automation Server Quick Reference

- SiebelHTMLApplication Methods for Siebel Web Client Automation Server 377
- SiebelService Methods for Siebel Web Client Automation Server 378
- Property Set Methods for Siebel Web Client Automation Server 379

#### Chapter 11: Siebel VB Quick Reference

- Applet Methods for Siebel VB 381
WebApplet Events for Siebel VB 382
Application Methods for Siebel VB 383
Application Events for Siebel VB 386
Business Component Methods for Siebel VB 386
Business Component Events for Siebel VB 391
Business Object Methods for Siebel VB 393
Business Service Methods for Siebel VB 393
Business Service Events for Siebel VB 394
Property Set Methods for Siebel VB 395
Miscellaneous Methods for Siebel VB 397

Chapter 12: Browser Scripting
About Browser Script 399
Applet Methods for Browser Script 400
Applet Events for Browser Script 401
Application Methods for Browser Script 401
Application Events for Browser Script 403
Business Component Methods for Browser Script 403
Business Component Events for Browser Script 405
Business Object Methods for Browser Script 405
Business Service Methods for Browser Script 406
Business Service Events for Browser Script 407
Property Set Methods for Browser Script 407
Control Methods for Browser Script 409
Supported DOM Events for High Interactivity Mode 410
Supported DOM Events for Standard Interactivity Mode 411

Chapter 13: Siebel eScript Quick Reference
Applet Methods for eScript 415
WebApplet Events for eScript 416
Application Methods for eScript 417
Application Events for eScript 419
Contents

Business Component Methods for eScript  420
Business Component Events for eScript  424
Business Object Methods for eScript  426
Business Service Methods for eScript  426
Business Service Events for eScript  427
Property Set Methods for eScript  428
Miscellaneous Methods for eScript  429

Chapter 14: Invoking Custom Methods with MiniButton Controls
Invoking Custom Methods with MiniButton Controls  431

Index
What’s New in Siebel Object Interfaces Reference, Version 8.0, Rev. B

Table 1 lists changes in this version of the documentation to support release 8.0 of Oracle’s Siebel software.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Siebel Web Client Automation Server&quot; on page 34</td>
<td>The EnableWebClientAutomation parameter is set for the Application Object Manager in version 8.0, not in the application configuration file.</td>
</tr>
<tr>
<td>&quot;Instantiating the Java Data Bean&quot; on page 52</td>
<td>Updated the connect string to include explicitly the port for the SCBroker component.</td>
</tr>
<tr>
<td>&quot;Java Data Bean and the siebel.properties File&quot; on page 53</td>
<td>The maximum possible value for both siebel.conmgr.txtimeout and siebel.conmgr.sesstimeout in the siebel.properties file is the largest positive integer, 2,147,483,647.</td>
</tr>
<tr>
<td>Inter-Application Variable Methods</td>
<td>Removed the topic. These methods are described elsewhere in the documentation.</td>
</tr>
<tr>
<td>&quot;Connect String&quot; on page 74</td>
<td>Removed the http transport protocol, which is not valid for connect strings.</td>
</tr>
<tr>
<td>&quot;Using Load Balancing with the Connect String&quot; on page 76</td>
<td>When the COM Data Control operates in server mode in an environment that implements load balancing, the VirtualServer parameter in the connect string must match the VirtualServer parameter in the session manager rules in the lbconfig.txt file.</td>
</tr>
<tr>
<td>&quot;Name Method&quot; on page 102</td>
<td>Changed the function name in the Browser Script example to Applet_Load.</td>
</tr>
<tr>
<td>&quot;Applet_ChangeRecord Event&quot; on page 105</td>
<td>Added a note describing the difference between the BusComp.GetFieldValue() method and the control.GetValue() method when used in this event.</td>
</tr>
<tr>
<td>&quot;GetSharedGlobal Method&quot; on page 138</td>
<td>This method and the SetSharedGlobal method are not used with the Java Data Bean. Use GetProfileAttr and SetProfileAttr instead.</td>
</tr>
<tr>
<td>&quot;GetDataSource Method&quot; on page 143</td>
<td>Returns the data source defined in the DataSource server parameter, not in the application configuration file.</td>
</tr>
</tbody>
</table>
### What’s New in This Release

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“LoadObjects Method” on page 146</td>
<td>The LoadObjects method does not return an Application object. It either returns nothing or throws an error.</td>
</tr>
<tr>
<td>“LoadUserAttributes Method” on page 148</td>
<td>If the row ID argument is the row ID of the current user, the user profile will be loaded into the “Me” profile.</td>
</tr>
<tr>
<td>“SetProfileAttr Method” on page 162</td>
<td>Added a note that system fields cannot be used with this method.</td>
</tr>
<tr>
<td>“Application_Start Event” on page 180</td>
<td>Added a caution not to use the RaiseErrorText() method in the Application_Start event. That method does not work in this event, and can cause the Application Object Manager to abort.</td>
</tr>
<tr>
<td>“ActivateField Method” on page 183</td>
<td>If the Show In List property of a list column is TRUE, the field will be active even if it is not shown on the applet.</td>
</tr>
<tr>
<td>“ExecuteQuery Method” on page 193</td>
<td>Added a note that you must activate fields by using the ActivateField method before executing a query for a business component.</td>
</tr>
<tr>
<td>“FirstRecord Method” on page 196</td>
<td>Added a note that this method can cause extra SQL code to be generated.</td>
</tr>
<tr>
<td>“InvokeMethod Methods for the Business Component Object” on page 216</td>
<td>Added the ClearLOVCache method.</td>
</tr>
<tr>
<td></td>
<td>Added examples for the CreateFile, GetFile, and PutFile methods.</td>
</tr>
<tr>
<td>“BusComp_DeleteRecord Event” on page 260 and “BusComp_PreDeleteRecord Event” on page 263</td>
<td>The BusComp_PreDeleteRecord and BusComp_DeleteRecord events do not execute for child records that are deleted due to the Cascade Delete property on a link.</td>
</tr>
<tr>
<td>“Service_PreInvokeMethod Event” on page 290</td>
<td>Added a note that Browser Script does not support output property sets with this event.</td>
</tr>
<tr>
<td></td>
<td>Added explanations of the intended uses of the Browser Script and Server Script versions of the Service_PreInvokeMethod event.</td>
</tr>
<tr>
<td></td>
<td>Described the differences in the handling of standard and custom business service methods.</td>
</tr>
<tr>
<td>“SetLabelProperty Method” on page 297 and “SetProperty Method” on page 299</td>
<td>Color formats for the BgColor and FontColor properties start with a hash mark (#).</td>
</tr>
<tr>
<td>“Property Set Methods” on page 302</td>
<td>Removed the ToString method, which does not return a useful result.</td>
</tr>
<tr>
<td>Business Component Methods for COM Data Control</td>
<td>Added the SetUserProperty method.</td>
</tr>
</tbody>
</table>
Table 1. What’s New in Siebel Object Interfaces Reference, Version 8.0, Rev. B

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Application Methods for COM Data Server&quot; on page 343</td>
<td>Removed the GetLastErrCode method, which is not used with the COM Data Server.</td>
</tr>
<tr>
<td>&quot;Application Methods for Browser Script&quot; on page 401</td>
<td>Added the ShowModalDialog method.</td>
</tr>
<tr>
<td>&quot;Invoking Custom Methods with MiniButton Controls” on page 431</td>
<td>The declarative method of adding a button is strongly recommended over the server script method for performance reasons.</td>
</tr>
</tbody>
</table>

What’s New in Siebel Object Interfaces Reference, Version 8.0 Rev. A

Table 2 lists changes in this version of the documentation to support release 8.0 of the software.

Table 2. What’s New in Siebel Object Interfaces Reference, Version 8.0 Rev. A

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Instantiating the Siebel COM Data Server” on page 48</td>
<td>Corrected the VB code example.</td>
</tr>
<tr>
<td>&quot;About Java Data Bean” on page 52</td>
<td>Siebel JAR files are version specific. For more information, see Doc ID 478113.1 on OracleMetaLink 3. This document was previously published as Siebel FAQ 2219. Removed the version number from JDK and added a reference to the Sun Microsystems Java Web site.</td>
</tr>
<tr>
<td>&quot;Java Data Bean and the siebel.properties File” on page 53</td>
<td>Corrected the siebel.conmgr.txtimeout and siebel.conmgr.sesstimeout parameters.</td>
</tr>
<tr>
<td>&quot;Connect String” on page 74</td>
<td>Added the language parameter to the connect string syntax.</td>
</tr>
<tr>
<td>&quot;WebApplet_PreCanInvokeMethod Event” on page 112</td>
<td>Updated the conditions under which this event is called.</td>
</tr>
<tr>
<td>&quot;GetService Method” on page 135</td>
<td>In version 8.0, to invoke a business service using the Web Client Automation Server and Browser Script, the business service must first be registered in Siebel Tools as an application user property.</td>
</tr>
<tr>
<td>&quot;InvokeMethod Methods for the Application Object” on page 143</td>
<td>Added the Language method.</td>
</tr>
<tr>
<td>&quot;RaiseError Method” on page 158 and &quot;RaiseErrorText Method” on page 159</td>
<td>These methods cause execution of the script to terminate. CancelOperation is not required after RaiseError and RaiseErrorText. Added a caution that because these methods cancel operations, be careful when using them.</td>
</tr>
</tbody>
</table>
Table 2. What's New in Siebel Object Interfaces Reference, Version 8.0 Rev. A

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Trace Method” on page 169 and “TraceOn Method” on page 172</td>
<td>Added notes that these methods are meant for debugging purposes and are not recommended for use in production environments.</td>
</tr>
<tr>
<td>“InvokeMethod Methods for the Business Component Object” on page 216</td>
<td>Added the RefreshBusComp method.</td>
</tr>
<tr>
<td></td>
<td>Noted the class dependency for the RefreshRecord method.</td>
</tr>
<tr>
<td>“NewRecord Method” on page 225</td>
<td>In certain cases, using the NewRecord method in a server script results in slow performance for this method.</td>
</tr>
<tr>
<td></td>
<td>For more information, see Doc ID 477556.1 on OracleMetaLink 3. This document was previously published as Siebel FAQ 2079.</td>
</tr>
<tr>
<td>“SetSearchExpr Method” on page 242 and “SetSearchSpec Method” on page 244</td>
<td>It is not necessary to activate fields that are used in SetSearchExpr and SetSearchSpec statements.</td>
</tr>
<tr>
<td>“BusComp_PreWriteRecord Event” on page 269 and “BusComp_WriteRecord Event” on page 272</td>
<td>Added a caution to be careful when using the RaiseError or RaiseErrorText method in these events, because these methods cancel operations.</td>
</tr>
<tr>
<td>“Service_InvokeMethod Event” on page 287</td>
<td>Browser Script does not support output property sets with this event. Added a note and rewrote the example in Siebel eScript.</td>
</tr>
<tr>
<td>“GetByteValue Method” on page 305 and “SetByteValue Method” on page 317</td>
<td>Added these property set methods for Java Data Bean.</td>
</tr>
<tr>
<td>“GetChild Method” on page 306</td>
<td>When using the Web Client Automation Server, the child object retrieved is a copy of the actual object.</td>
</tr>
<tr>
<td>“Business Component Methods for Java Data Bean” on page 369</td>
<td>The ExecuteQuery method requires Boolean cursor modes.</td>
</tr>
<tr>
<td>“Business Component Events for Browser Script” on page 405</td>
<td>The BusComp_PreSetFieldValue browser script event is invoked after the server round trip if the Immediate Post Changes property of the Business Component field is set to TRUE.</td>
</tr>
</tbody>
</table>
What’s New in This Release

Table 3 lists changes in this version of the documentation to support release 8.0 of the software.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ST eScript Engine&quot; on page 24</td>
<td>The ST eScript engine is the default eScript scripting engine for version 8.0.</td>
</tr>
<tr>
<td>&quot;Installing Siebel Object Interfaces&quot; on page 37</td>
<td>The references to BizTalk Connector and OLE DB Provider were removed.</td>
</tr>
<tr>
<td>&quot;Siebel Constants&quot; on page 95</td>
<td>The values for ForwardBackward and ForwardOnly were corrected to 256 and 257, respectively.</td>
</tr>
<tr>
<td>&quot;WebApplet_PreCanInvokeMethod Event&quot; on page 112</td>
<td>The CanInvokeMethod applet user property can be used to enable and disable methods declaratively at the applet level, and is often easier to use than PreCanInvokeMethod.</td>
</tr>
</tbody>
</table>
| "InvokeMethod Methods for the Application Object" on page 143 | The following methods are supported for use with the InvokeMethod method:  
  - GetDataSource  
  - IsViewReadOnly  
  - LookupValue |
| "Business Component Methods" on page 181 | The GetSortSpec method was added. |
| "Business Service Methods" on page 278 | The GetLastErrCode and GetLastErrText methods were deleted. |
| "Property Set Methods" on page 302 | The GetLastErrCode and GetLastErrText methods were added. |
| "Business Component Methods for COM Data Server" on page 346 | The FirstSelected method was added and the cross-reference for GetAssocBusComp Method was corrected. |
| "Application Methods for Mobile Web Client Automation Server" on page 355 | The Login method was added. |
| "Business Component Events for Browser Script" on page 405 | The BusComp_PreSetFieldValue event is not invoked on picklists and multivalue fields. |
| "Invoking Custom Methods with MiniButton Controls" on page 431 | The procedure was updated to reflect changes in the Siebel Tools user interface in version 8.0 and to include an example of the use of the CanInvokeMethod applet user property. |
This chapter describes the Siebel programming tools. It contains the following topics:

- “About the Siebel Programming Tools” on page 19
- “Components of the Siebel Programming Environment” on page 20
- “Supported Uses of Siebel Programming Languages” on page 21
- “Adding New Business Logic to a Business Component” on page 22
- “Tracing Scripts” on page 22
- “About the Siebel Compiler and Run-Time Engine” on page 24
- “About Siebel VB” on page 25
- “About Siebel eScript” on page 27

## About the Siebel Programming Tools

Oracle’s Siebel Business 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. For more information, see *Siebel Performance Tuning Guide*.
Components of the Siebel Programming Environment

The individual components of the Siebel programming environment include:

- **Server Script.** The following scripting languages can be used in server scripts:
  - **Siebel VB.** 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.** 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 the Microsoft Windows operating system only.

- **Browser Script.** A type of script (introduced 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 script editor has a code editing feature called Script Assist (introduced in version 7.8). Script Assist provides auto-complete, auto-indentation, method listing, and method signature capabilities to help minimize errors as you develop script. For more information about the Siebel Script Editor, including how to enable Script Assist, see *Using Siebel Tools*.

- **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 startup line. The Debugger can also be invoked from the Debug toolbar and Debug menu. The Debugger is described in more detail in *Using Siebel Tools*.

- **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 Logic Definition" on page 21
- "Custom Behavior for User Interface Components" on page 21

Business Logic 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 can incorporate validations based on data from sources within or outside the Siebel application. For example, a validation routine might 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 the Siebel Applet Layout Editor, 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 Microsoft 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 current user’s position.

### Adding New Business Logic to a Business Component

You use browser scripts and server scripts to add business logic to business components.

**To add business logic 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, and then choose Edit Browser Scripts or Edit Server Scripts.
   - The Script Editor appears.
5. Select the event from the Event List Tree applet, and then add your server scripts in the Script Editor.
6. Validate the Siebel script syntax by choosing Debug > Check Syntax.
   - **NOTE:** The Check Syntax menu item is available for server scripts only.
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.
10. Choose Repository > Check In to check the modified project into the server repository.

### Tracing Scripts

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.

For more information on configuring server components, see *Siebel Applications Administration Guide*. For information on logging events, see *Siebel System Monitoring and Diagnostics Guide*.

**To enable logging**

1. Navigate to Server Configuration > 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 the following:
   - **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.

<table>
<thead>
<tr>
<th>Information to Trace</th>
<th>Parameter Alias</th>
<th>Settings for Current Value and Value on Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocations</td>
<td>TraceAlloc</td>
<td>0 (zero) to disable logging, 1 to enable logging</td>
</tr>
<tr>
<td>Events</td>
<td>TraceEvents</td>
<td>0 (zero) to disable logging, 1 to enable logging</td>
</tr>
<tr>
<td>SQL Commands</td>
<td>TraceSql</td>
<td>0 (zero) to disable logging, 1 to enable logging</td>
</tr>
<tr>
<td>Users</td>
<td>TraceUser</td>
<td>Comma-separated list of user names. Do not use spaces (for example: sadmin,mmasters). The length of this parameter is limited to 20 characters.</td>
</tr>
</tbody>
</table>

**NOTE:** Server-side tracing can have a significant impact on performance. Use caution when making it available for multiple users simultaneously.

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 15:37:20 Login name : SADMIN
ObjMgrSessionInfoObjMgrAuth32003-04-09 15:37:20 Authentication name : SADMIN
ObjMgrSessionInfoObjMgrLogin32003-04-09 15:37:20 Session Type: Regular Session
GenericLogGenericError12003-04-09 15:37:20 Invocation of Applet Menu New Service::NewExpense is not allowed.
```
About the 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 Siebel VB Language Reference.

Siebel eScript does not require forward declaration of functions.

ST eScript Engine

In version 7.8 and higher, the ST eScript engine is available. It is the default eScript scripting engine in version 8.0. The new engine provides support for strongly typed objects (compliant with the ECMAScript edition 4 specification). In addition, the new eScript engine provides other enhancements, such as late and early binding. For information about the differences between the ST eScript engine and the T engine, see Siebel eScript Language Reference. For information on using the ST engine, see Using Siebel Tools.
About Siebel VB

If you have never programmed in Visual Basic before, you may want to start by reading *Siebel VB Language Reference*. It includes information on 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 in Table 4.

### Table 4. Naming Conventions for Variables in Scripts

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Symbol</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>s</td>
<td>sName</td>
</tr>
<tr>
<td>Integer</td>
<td>i</td>
<td>iReturn</td>
</tr>
<tr>
<td>Long integer</td>
<td>l</td>
<td>lBigCount</td>
</tr>
<tr>
<td>Single-precision number</td>
<td>si</td>
<td>siAllowance</td>
</tr>
<tr>
<td>Double-precision number</td>
<td>d</td>
<td>dBudget</td>
</tr>
<tr>
<td>Object</td>
<td>o</td>
<td>oBusComp</td>
</tr>
<tr>
<td>Currency</td>
<td>c</td>
<td>cAmtOwed</td>
</tr>
</tbody>
</table>

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:

```vba
Function BusComp_PreSetFieldValue(FIELDNAME As String, FIELDDATE As String) As Integer
    If Val(Me.GetFieldValue("Rep %")) > 75 Then
        TheApplication.RaiseErrorText("You cannot set the Rep% to greater than 75")
    End If
    BusComp_PreSetFieldValue = ContinueOperation
End Function
```
You can see other examples of Me in "ParentBusComp Method" on page 228, "SetViewMode Method" on page 251, "BusComp_PreQuery Event" on page 266, "BusComp_PreWriteRecord Event" on page 269, and "ActiveMode Method" on page 96.

**Trap run-time errors.** The standard VB methods return numeric error codes, which are documented in *Siebel VB Language Reference*. 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. 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.

```vbnet
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, see the Language Overview topics in *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 reduces typing and makes the code easier to read. Instead of a series of statements such as:

```vbnet
Set oBusObject = TheApplication.GetBusObject("Opportunity")
Set oBusComp = oBusObject.GetBusComp("Opportunity")
oBusComp.ActivateField "Account"
oBusComp.ClearToQuery
oBusComp.SetSearchSpec "Name", varname
oBusComp.ExecuteQuery ForwardBackward
If (oBusComp.FirstRecord = 1) Then
  sAccount = oBusComp.GetFieldValue "Account"
End If
```

use the following:

```vbnet
Set oBusObject = TheApplication.GetBusObject("Opportunity")
Set oBusComp = oBusObject.GetBusComp("Opportunity")
With oBusComp
  .ActivateField "Account"
  .ClearToQuery
  .SetSearchSpec "Name", varname
  .ExecuteQuery ForwardBackward
  If (.FirstRecord = 1) Then
```

26  *Siebel Object Interfaces Reference* Version 8.0, Rev. B
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 noncompliance. 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 Method" on page 201.
2. Convert it into a date variable using the DateSerial() function.
3. Perform the required date arithmetic.

The following example is in Siebel VB:

```vbscript
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 takes care of object cleanup, it is a best practice to write code that explicitly destroys objects when they are no longer used. Explicit destruction of Siebel objects should occur in the procedure in which they are created.

To destroy objects in Siebel VB, set each object to Nothing in the reverse order of creation. Destroy child objects before parent objects. For example:

```vbscript
Set oBusObj = TheApplication.GetBusObject("contact")
Set oBusComp = oBusObj.GetBusComp("contact")
\[ Your code here \]
Set oBusComp = Nothing
Set oBusObj = Nothing
```

### 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, see Siebel eScript Language Reference.

Declare your variables. Standard ECMAscript does not require that you declare variables. Variables are declared implicitly as soon as they are used. As a general rule, you should 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.

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 in Table 4 on page 25.

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:

```javascript
if (condition)
{ ...
    this.SetSearchSpec(...);
    this.ExecuteQuery
    return (CancelOperation)
}
else
    return(ContinueOperation);
```

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

```javascript
var oBusObject = TheApplication().GetBusObject("Opportunity");
var oBusComp = oBusObject.GetBusComp("Opportunity");
oBusComp.ActivateField("Account");
oBusComp.ClearToQuery();
oBusComp.SetSearchSpec("Name", varname);
oBusComp.ExecuteQuery(ForwardBackward);
if oBusComp.FirstRecord();
```
Use the following:

```javascript
var oBusObject = TheApplication().GetBusObject("Opportunity");
var oBusComp = oBusObject.GetBusComp("Opportunity");
with (oBusComp)
{
  ActivateField("Account");
  ClearToQuery();
  SetSearchSpec("Name", varname);
  ExecuteQuery(ForwardBackward);
  if (FirstRecord())
  {
    var sAccount = GetFieldValue("Account");
  }
} //end with

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.

```javascript
switch (FieldName)
{
  case "Status":
  {
    var sysdate = new Date();
    var sysdatestring = "\" + sysdate.getMonth() + 1) + "/" + sysdate.getDate() + "/" + sysdate.getFullYear() + " + sysdate.getHours()+":" +
      sysdate.getMinutes()+":" + sysdate.getSeconds());
    this.SetFieldValue("Sales Stage Date", sysdatestring);
    if ((FieldValue) == "Not Attempted")
    {
      if (this.GetFieldValue("Primary Revenue Amount") > 0)
        this.SetFieldValue("Primary Revenue Amount",0);
    }
    break;
  }
  case "Revenue":
  {
    if (newrecSw =="Y")
    {
      newrecSw = "";
      this.SetFieldValue("Account Revenue", (FieldValue));
    }
    break;
  }
}
Destroy any objects you have created when you no longer need them. While the interpreter takes care of object cleanup, it is a best practice to write code that explicitly destroys objects when they are no longer used. Explicit destruction of Siebel objects should occur in the procedure in which they are created.

To destroy objects in Siebel eScript, set each object to null in the reverse order of creation. Destroy child objects before parent objects. For example:

```javascript
var oBusObject = TheApplication().GetBusObject("Contact");
var oBusComp = oBusObject.GetBusComp("Contact");

[ Your code here ]

oBusComp = null;
oBusObject = null;
```
This chapter provides information about installing and using Siebel object interfaces. It includes the following topics:

- “About Programming with Siebel Object Interfaces” on page 31
- “About Siebel Object Interfaces” on page 32
- “Installing Siebel Object Interfaces” on page 37
- “Exposed Object Types” on page 38
- “Siebel Object Interface Method Syntax” on page 41
- “Getting Started with the Siebel Object Interfaces” on page 43
- “Siebel Object Interface Methods” on page 58
- “Variable Scoping for Siebel Script Variables” on page 65
- “Siebel Object Interface Events and Siebel Extension Events” on page 67

### About Programming with Siebel Object Interfaces

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
- 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.

Oracle 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 Oracle. Oracle 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 4, “Interfaces Reference” describes the supported methods of the Siebel object interfaces and provides examples of how you can use them.

## About Siebel Object Interfaces

Siebel object interfaces include:

- “Siebel COM Interfaces” on page 32
- “Siebel Java Interfaces” on page 36
- Built-in scripting of Siebel objects using Siebel VB, Siebel eScript, and Browser Script. For more information, read “Built-In Scripting” on page 36.

**Related Topic**

“Usage Evaluation Matrix” on page 37

## Siebel COM Interfaces

The supported languages for accessing Siebel COM interfaces are JavaScript, Visual Basic, and C++. The Perl language is not supported with Siebel COM interfaces.

**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 System Administration Guide* for information about installing and configuring the Siebel Object Manager.

Any external application or component that uses Siebel COM Data Control connects and communicates with Siebel Application Object Manager using the Siebel Internet Session Network API (SISNAPI) protocol. The Siebel Application Object Manager, which could be running on a remote Siebel Server, is a multithreaded, multiprocess application server that hosts Siebel business objects and supports session-based connections by clients.

For information on the SISNAPI protocol, see *Siebel Deployment Planning Guide*.

Figure 1 shows how external applications use Siebel COM Data Control to communicate with the Siebel application.

![Figure 1. Siebel COM Data Control](image-url)
**COM Data Server**

Figure 2 shows how external applications use Siebel COM Data Server without having to access the user interface objects. COM Data Server uses the same mechanism as the Mobile Web Client to connect to the server database.

![COM Data Server Diagram](image)

Figure 2. Siebel COM Data Server

You can expect differences in performance between Siebel COM Data Server and Siebel Mobile 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.

**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 Interactivity 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 for the Application Object Manager. With this parameter set to TRUE, a small ActiveX Control downloads to the desktop and the SiebelHTMLApplication process starts.

In the Windows Task Manager, this process is named one of the following:

- siebelhtml.exe
- siebelhtmlapplication.exe
- SIEBEL~1.EXE
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 3 shows how external applications can invoke business services and manipulate property sets in the Siebel Web Client Automation Server.

**Siebel Mobile Web Client Automation Server**

When accessing the Mobile Web Client Automation Server, Siebel Mobile Web Client must be running. Figure 4 shows how the Siebel Mobile Web Client Automation Server is used by external applications to control the Siebel application.

Figure 4. Siebel Mobile 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 using the SISNAPI protocol. 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.

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 5** to determine which types of Siebel object interfaces to use.

**NOTE:** Throughout this guide, X in a table denotes that an object interface type can be used with a specified method or object type, or for a specified purpose.

### Table 5. Usage Evaluation

<table>
<thead>
<tr>
<th>Usage</th>
<th>Web Client Automation Server</th>
<th>Mobile Web Client Automation Server</th>
<th>Siebel COM Data Control</th>
<th>Siebel COM Data Server</th>
<th>Siebel Java Data Bean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Siebel user interface from your external application</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Access Siebel business objects without Siebel user interface</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Objects execute on a Siebel Server</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Execute on the client side in mobile environments</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Installing Siebel Object Interfaces**

**Table 6** lists the installation procedure for each object interface.

### Table 6. Interface Installation

<table>
<thead>
<tr>
<th>Interface</th>
<th>Installation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Data Bean</td>
<td>Installed by the Siebel Enterprise Server Installer under a Typical installation, with the “EAI Siebel Connectors” option. Also installed by default with Siebel Tools. For more information on the EAI Siebel Connectors option, see the <em>Siebel Installation Guide</em> for the operating system you are using.</td>
</tr>
<tr>
<td>COM Data Control</td>
<td>Installed by the Siebel Enterprise Server Installer under a Typical installation, with the “EAI Siebel Connectors” option. For more information on the EAI Siebel Connectors option, see the <em>Siebel Installation Guide</em> for the operating system you are using.</td>
</tr>
<tr>
<td>COM Data Server</td>
<td>Installed by default with the Mobile Web Client.</td>
</tr>
</tbody>
</table>
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 38
- "Business Object Object Type" on page 38
- "Business Component Object Type" on page 39
- "Business Service Object Type" on page 39
- "Applet Object Type" on page 40
- "Property Set Object Type" on page 40
- "User Interface Control Object Type" on page 40

There are additional object types used in Siebel Business 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 specialized C++ classes. 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.

**NOTE:** To invoke a business service using the Web Client Automation Server and Browser Script, the business service must first be registered in Siebel Tools as an application user property. This prevents Service Not Found errors. For more information, see “GetService Method” on page 135.
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 Web Client Automation Server, or Siebel Java Data Bean.

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 Web Client Automation Server, Web Client Automation Server, or Siebel Java Data Bean.

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.
Summary of Exposed Object Types

Table 7 summarizes the names and types of objects exposed.

Table 7. Exposed Object Types for Each Siebel Object Interface

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Siebel Web Client Automation Server</th>
<th>Siebel Mobile Web Client Automation Server</th>
<th>Siebel COM Data Control</th>
<th>Siebel COM Data Server</th>
<th>Siebel Java Data Bean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applet</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Application</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Business Component</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Business Object</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Business Service</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Property Set</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg1</td>
<td>Description of arg1</td>
</tr>
<tr>
<td>arg2</td>
<td>Description of arg2</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>argn</td>
<td>Description of argn</td>
</tr>
</tbody>
</table>

Returns

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

The following conventions are used in the syntax diagram:
Siebel Object Interface Method Syntax

- **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 argument. In the description of the argument, the default value for the optional argument 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 using 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:

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

If there are no arguments:

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

If there is no return value:

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

**Examples**

```javascript
acctName = acctBC.GetFieldValue("Name");
acctBC.SetViewMode(AllView);
```

Using parentheses ( ) 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 or COM Controls:

- "Accessing Siebel COM Interfaces" on page 44
- "Accessing the Siebel Web Client Automation Server" on page 44
- "Accessing the Siebel Mobile Web Client Automation Server" on page 46
- "Instantiating the Siebel COM Data Server" on page 48
- "Instantiating the Siebel COM Data Control" on page 50
- "About Java Data Bean" on page 52
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 5 displays an example of an object browser in Microsoft Visual Basic 5.0.

![Syntax window](image)

Figure 5. 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 Web Client. The Web Client Automation Server is supported with the high interactivity mode only.

**NOTE:** You cannot invoke the Siebel Web Client Automation Server directly from the active Siebel instance. Trying to do so will cause an error.

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:

```vba
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 Siebel HTML Object
    Set siebApp = CreateObject("Siebel.Desktop_Integration_Application.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
        End If
    Set siebApp = Nothing
    End If
End Sub
```
Accessing the Siebel Mobile Web Client Automation Server

The Siebel Mobile Web Client Automation Server accesses the server object instantiated by the Siebel Business Application. When 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 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 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 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 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 Web Client Automation Server.

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

```vba
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")
```

46 Siebel Object Interfaces Reference Version 8.0, Rev. B
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

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

   End Sub
Instantiating the Siebel COM Data Server

Because the Siebel COM Data Server acts without the regular Siebel Business 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.

When using COM Data Server, the COM client cannot create multiple connections to the COM Server. The COM client must be restarted before another connection attempt can be successful. Use COM Data Control instead.

**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 dialog box that appears, click on Siebel Data BusObject Interfaces, but do not check its checkbox. Near the bottom of the dialog box, note the directory in which the file sobjsrv.tlb resides, as shown in the following illustration.

5. Select the Siebel Data BusObject Interfaces checkbox, and then 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
      'Create 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
        siebApp.RaiseErrorText("Business Object Creation failed: " & errCode & ":" & errText);
      End If
    End If
    'Create A New Property Set
    Set siebPropSet = siebApp.NewPropertySet(errCode)
    If errCode = 0 Then
      Set siebPropSet = Nothing
    Else
      errText = siebApp.GetLastErrText
      siebApp.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
      Set siebSvcs = Nothing
    End If
  End If
End If
End Sub
errText = siebApp.GetLastErrText
siebApp.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
    siebApp.RaiseErrorText("Login Failed: " & errCode & ":" & errText);
End If
Else
    errText = siebApp.GetLastErrText
    siebApp.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 37.
2. Start Microsoft Visual Basic.
3. Select Standard EXE.
4. Choose Project > References.
5 In the dialog box that appears, select the Siebel BusObject Interfaces Type Library checkbox.

6 Click OK to open the Object Browser, and then 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 currently 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:

```vbscript
Sub CreateDataControl()
  Dim errCode As Integer
  Set SiebelApplication = CreateObject("SiebelDataControl.SiebelDataControl.1")
  SiebelApplication.Login "host=" & "siebel://hostname/EnterpriseServer/AppObjMgr='", "CCONWAY", "CCONWAY"
  errCode = SiebelApplication.GetLastError()
  If errCode <> 0 Then
    ErrText = SiebelApplication.GetLastErrText
    TheApplication().RaiseErrorText(ErrText)
    Exit Sub
  End If
  Set OpptyBC = OpptyBO.GetBusComp("Opportunity", errCode)
End Sub
```

See Table 20 on page 75 for values to substitute for the placeholders in the login string.

The following sample code instantiates 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.

```html
<%
  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) & "<hostname>:2321/EntServer/ObjMgr\" & 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 74.

### About 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 J2SE Development Kit (JDK). Users can incorporate these libraries to build Java Applications, Applets, Servlets, JSPs, or Enterprise Java Beans into their Java-based applications. For more information on Java, refer to http://java.sun.com.

A Java client that uses the Java Data Bean interface to connect to the Siebel server needs several JAR files that provide the objects and methods of the Siebel Object Interface to the Java language. The JAR files of the Java Data Bean interface are specific to the Siebel application version with which they were delivered. Do not use these JAR files with other versions of the Siebel server.

**NOTE:** Before compilation or execution, add the Siebel JAR files (`Siebel.jar` and `SiebelJI_<lang>.jar`) to the CLASSPATH.

### Supported Platforms and JDKs

Refer to Siebel System Requirements and Supported Platforms on Oracle Technology Network.

### 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 sample code for the Siebel Java Data Bean:

```java
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();

            // log in to the server
            // SiebelServerhost = the name or IP address of your Siebel Server
            // SCBPort = listening port number for the SCBroker component (default 2321)
            m_dataBean.login("Siebel://SiebelServerhost:SCBPort/enterpriseServer/
                AppObjMgr_enu", "CCONWAY", "CCONWAY", "enu");

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

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

            // log off
            m_dataBean.logoff();
        }
        catch (SiebelException e)
        {
            System.out.println(e.getErrorMessage());
        }
    }
}

NOTE: If you are using a single sign-on (SSO) with Java Data Bean, you must use the login ID of an employee as the username and the value of the trust token (the TrustToken parameter in the application's CFG file) in the connect string to log into the server. For more information, see “Connect String” on page 74.

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 8 shows the properties in the siebel.properties file.

### Table 8. Properties in the siebel.properties File

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siebel Connection Manager</td>
<td>siebel.conmgr.txtimeout</td>
<td>Indicates the transaction timeout (in milliseconds). Defaults to 600000 = 10 minutes. The maximum value is 2,147,483,647 ms (approximately 25 days).</td>
</tr>
<tr>
<td></td>
<td>siebel.conmgr.poolsize</td>
<td>Indicates the connection pool size. Connection pool maintains a set of connections to a specific server process. Defaults to 2. Max connection pool size is 500.</td>
</tr>
<tr>
<td></td>
<td>siebel.conmgr.sesstimeout</td>
<td>Indicates the transaction timeout (in seconds) on the client side. Defaults to 2700 = 45 minutes. The maximum value is 2,147,483,647 s (approximately 68 years).</td>
</tr>
<tr>
<td></td>
<td>siebel.conmgr.retry</td>
<td>Indicates the number of open session retries. Defaults to 3.</td>
</tr>
<tr>
<td></td>
<td>siebel.conmgr.jce</td>
<td>Indicates the usage of Java Cryptography Extension (JCE). For more information on JCE, see “Encrypting Communication Between JDB and Siebel Server” on page 56. The value is 1 if using JCE and 0 if not using JCE.</td>
</tr>
<tr>
<td>Siebel generated code for Java EE Connector Architecture (JCA) and Java Data Bean (JDB)</td>
<td>siebel.connection.string</td>
<td>Specifies the Siebel connection string.</td>
</tr>
<tr>
<td></td>
<td>siebel.user.name</td>
<td>Specifies the user name to be used for logging in to Object Manager.</td>
</tr>
<tr>
<td></td>
<td>siebel.user.password</td>
<td>Specifies the password to be used for logging in to Object Manager.</td>
</tr>
<tr>
<td></td>
<td>siebel.user.language</td>
<td>Specifies the user's preferred language.</td>
</tr>
<tr>
<td></td>
<td>siebel.user.encrypted</td>
<td>Specifies whether the username and password is encrypted.</td>
</tr>
<tr>
<td></td>
<td>siebel.jdb.classname</td>
<td>Specifies the default JDB classname</td>
</tr>
<tr>
<td>Java System Properties</td>
<td>file.encoding</td>
<td>Indicates the codepage on the client side. For example, cp1252, utf8, unicodeBig, cp942.</td>
</tr>
</tbody>
</table>
NOTE: Java System Properties are System Properties, not Siebel Properties.

The following is a sample siebel.properties file:

```java
siebel.connection.string = siebel.tcpip.rsanone://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 9 lists codepage mappings for JDB.

<table>
<thead>
<tr>
<th>Java Value</th>
<th>Siebel Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ascii</td>
<td>1</td>
</tr>
<tr>
<td>cp1252</td>
<td>1252</td>
</tr>
<tr>
<td>iso8859_1</td>
<td>1252</td>
</tr>
<tr>
<td>iso8859-1</td>
<td>1252</td>
</tr>
<tr>
<td>unicodebig</td>
<td>1201</td>
</tr>
<tr>
<td>unicodelittle</td>
<td>1200</td>
</tr>
<tr>
<td>utf8</td>
<td>65001</td>
</tr>
<tr>
<td>big5</td>
<td>950</td>
</tr>
</tbody>
</table>
Encrypting Communication Between JDB and Siebel Server

Siebel Business Applications 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 *Siebel System Requirements and Supported Platforms* on Oracle Technology Network.

### Table 9. Codepage Mappings for Java Data Bean

<table>
<thead>
<tr>
<th>Java Value</th>
<th>Siebel Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cp942</td>
<td>932</td>
</tr>
<tr>
<td>cp942c</td>
<td>932</td>
</tr>
<tr>
<td>cp943</td>
<td>932</td>
</tr>
<tr>
<td>cp943c</td>
<td>932</td>
</tr>
<tr>
<td>cp949</td>
<td>949</td>
</tr>
<tr>
<td>cp949c</td>
<td>949</td>
</tr>
<tr>
<td>cp950</td>
<td>950</td>
</tr>
<tr>
<td>cp1250</td>
<td>1250</td>
</tr>
<tr>
<td>cp1251</td>
<td>1251</td>
</tr>
<tr>
<td>cp1253</td>
<td>1253</td>
</tr>
<tr>
<td>cp1254</td>
<td>1254</td>
</tr>
<tr>
<td>cp1255</td>
<td>1255</td>
</tr>
<tr>
<td>cp1256</td>
<td>1256</td>
</tr>
<tr>
<td>cp1257</td>
<td>1257</td>
</tr>
<tr>
<td>cp1258</td>
<td>1258</td>
</tr>
<tr>
<td>gbk</td>
<td>936</td>
</tr>
<tr>
<td>ms874</td>
<td>874</td>
</tr>
<tr>
<td>ms932</td>
<td>932</td>
</tr>
<tr>
<td>ms936</td>
<td>936</td>
</tr>
<tr>
<td>ms949</td>
<td>949</td>
</tr>
<tr>
<td>ms950</td>
<td>950</td>
</tr>
<tr>
<td>sjis</td>
<td>932</td>
</tr>
<tr>
<td>tis620</td>
<td>874</td>
</tr>
</tbody>
</table>
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 System 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>`

   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, Oracle supports the Java Cryptography Extension (JCE) v1.2.1 specification. JCE provides a framework and implementations for encryption, key generation and key agreement, and Message Authentication Code (MAC) algorithms. JCE is designed so that other qualified cryptography libraries can be used as service providers. For more information on JCE, see http://java.sun.com/j2se/1.4.2/docs/guide/security/jce/JCERefGuide.html.

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 can 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
Siebel Object Interface Methods

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

- **Locating objects.** These methods allow the user to locate instances of objects so that they can be manipulated by other methods.
- **Accessing business components.** These methods provide the ability to access and modify data within Siebel applications.
- **Navigation.** These methods 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 Web Client Automation Server. When Siebel is accessed through the COM Data Control, COM Data Server, or Java Data Bean, no Siebel user interface is present.
- **Manipulating controls.** These methods 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 methods get information on the current state.
- **User interaction.** These methods provide user interface elements similar to those in standard Windows programs.
- **Tracing.** These methods control debug tracing.

**Related Topics**
- “Locating Objects”
- “Accessing Business Components” on page 59
- “Navigation Methods” on page 64
- “User Interaction Methods” on page 64
- “Global State Properties and Functions” on page 64
- “Tracing Methods” on page 65

**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 Method” on page 122
- “ActiveMode Method” on page 96
- “ActiveViewName Method” on page 124
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.Associate
  - BusComp.DeleteRecord (DeleteRecord commits automatically, because it moves the cursor to another record.)
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:

```vba
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, unless you want to change the view mode from the default)
3. ActivateField
4. ClearToQuery
5. SetSearchSpec or SetSearchExpr

**NOTE:** It is not necessary to activate fields on which search specs and search expressions are set, unless the fields are also referenced by the GetFieldValue or SetFieldValue method.

6. ExecuteQuery
The following example shows how to instantiate objects in eScript:

```javascript
function BusComp_PreSetFieldValue (FieldName, FieldValue) {
  var RetValue = ContinueOperation;
  switch (FieldName)
  {
    case "Sales Stage":
      if (FieldValue == "08 - Negotiation")
      {
        // 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
        var oBusObj;
        var sRowId;
        var iViewMode;
        sRowId = this.GetFieldValue("Id");
        iViewMode = this.GetViewMode();
        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.
        ContactBC = oBusObj.GetBusComp("Contact");
        with (ContactBC)
        {
          ClearToQuery();
          SetSearchSpec("Job Title", "*VP*");
          ExecuteQuery(ForwardOnly);
          if (FirstRecord())
          {
            TheApplication().RaiseErrorText("Found a decision maker");
          }
          else
          {
            RetVal = ContinueOperation;
          }
        }
        ContactBC = null;
        oBusObj = null;
      }
      break;
  }
  return(RetVal);
}
```

The following example shows how to instantiate objects in Siebel VB:

```vb
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
        .ClearToQuery
        .SetSearchSpec "Job Title", "*VP*
        .ExecuteQuery ForwardOnly
        If (.FirstRecord = 1) Then
            TheApplication.RaiseErrorText "Found a decision maker"
        Else
            RetVal = ContinueOperation
        End If
    End With
    Set ContactBC = Nothing
    Set oBusObj = Nothing
End If
End Select
BusComp_PreSetFieldValue = RetValue
End Function

Methods for Accessing Business Components
To access business components, use the following methods:

- "ActivateField Method" on page 183
- "ActivateMultipleFields Method" on page 184
- "Associate Method" on page 186
- "ClearToQuery Method" on page 189
- "CountRecords Method" on page 190
- "DeactivateFields Method" on page 191
- "DeleteRecord Method" on page 193
- "ExecuteQuery Method" on page 193
- "ExecuteQuery2 Method" on page 195
- "FirstRecord Method" on page 196
"FirstSelected Method" on page 198
"GetAssocBusComp Method" on page 200
"GetFieldValue Method" on page 201
"GetFormattedFieldValue Method" on page 203
"GetMultipleFieldValues Method" on page 206
"GetMVGBusComp Method" on page 207
"GetNamedSearch Method" on page 208
"GetPicklistBusComp Method" on page 209
"GetSearchExpr Method" on page 211
"GetSearchSpec Method" on page 212
"GetSortSpec Method" on page 212
"GetUserProperty Method" on page 213
"GetViewMode Method" on page 214
"InvokeMethod Method" on page 215
"LastRecord Method" on page 224
"NewRecord Method" on page 225
"NextRecord Method" on page 227
"ParentBusComp Method" on page 228
"Pick Method" on page 229
"PreviousRecord Method" on page 231
"RefineQuery Method" on page 232
"SetFieldValue Method" on page 235
"SetFormattedFieldValue Method" on page 237
"SetMultipleFieldValues Method" on page 238
"SetNamedSearch Method" on page 240
"SetSearchExpr Method" on page 242
"SetSearchSpec Method" on page 244
"SetSortSpec Method" on page 248
"SetViewMode Method" on page 251
"UndoRecord Method" on page 254
"WriteRecord Method" on page 255
Navigation Methods

The navigation methods set the focus for user interaction to the named view. Table 10 identifies the navigation methods. Cannot be invoked from Browser Script.

**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>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>“InvokeMethod Method” on page 100</td>
</tr>
<tr>
<td>“GotoView Method” on page 139</td>
</tr>
</tbody>
</table>

User Interaction Methods

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 11.

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>“RaiseError Method” on page 158</td>
</tr>
<tr>
<td>“RaiseErrorText Method” on page 159</td>
</tr>
</tbody>
</table>

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 12.

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>“CurrencyCode Method” on page 127</td>
</tr>
<tr>
<td>“EnableExceptions Method” on page 129</td>
</tr>
<tr>
<td>“GetLastErrCode Method” on page 133</td>
</tr>
<tr>
<td>“GetLastErrText Method” on page 134</td>
</tr>
<tr>
<td>“LoginId Method” on page 150</td>
</tr>
</tbody>
</table>
Variable Scoping for Siebel Script Variables

Three levels of scope exist for Siebel variables:

- "Local Variables" on page 65
- "Module Variables" on page 66
- "Global Variables" on page 67

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.

The following example is in Siebel eScript:

```javascript
function WebApplet_Load ()
{
    var localStr;
}
```
The following example is in Siebel VB:

```vbnet
Sub WebApplet_Load
    Dim localStr As String
End Sub
```

### 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.

The following example is in Siebel VB:

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

Code in the VB Editor in the (general) (declarations) section is illustrated in Figure 6.

![Figure 6. Declarations in the (general) (declarations) Section](image)

**Figure 6.** 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.

The following example is in Siebel eScript:

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

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 71 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 68
- "How Your Script Affects Program Flow” on page 68
- "Using Tracing to Find Out When Events Occur” on page 71
- “Siebel Business Component Events” on page 71
- “Applet Events” on page 73
- “Application Events” on page 74
- “Connect String” on page 74
- "Error Handling” on page 77

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(_arguments) As RetValue.`
- `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.

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.

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.

**CAUTION:** You must use `CancelOperation` with custom methods. If you do not, the code flow would continue to the C++ code, which does not have the ability to handle the custom method, and would therefore throw an “Unknown method name” error. For more information on the differences in handling standard and custom methods, see Figure 7 on page 291.

The following eScript example sets up a validation routine in which a specific field is queried to determine whether the event should fire:
function BusComp_PreSetFieldValue (FieldName, FieldValue) 
{
  var iReturn = ContinueOperation;
  //Routine to check if a quote discount > 20%
  //if it is, notify user and cancel the operation
  var varvalue;
  var msgtext;
  if (FieldName == "Discount")
  {
    varvalue = ToNumber(FieldValue);
    if (varvalue > 20)
    {
      msgtext = "Discounts greater than 20% must be approved";
      TheApplication().RaiseErrorText(msgtext); // cancels execution
    }
    else
    {
      iReturn = ContinueOperation;
    }
  }
}

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) ' cancels execution
    Else
      BusComp_PreSetFieldValue = ContinueOperation
    End If
  End If
End Function

Note the logical structure of this routine:

If (condition is true)
[perform custom routine]
[cancel operation by raising error text]
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:

```plaintext
defaultReturnValue = continueOperation
if (condition is true)
    [perform custom routine]
end if
```

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

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

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

```plaintext
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
```

The following example is in Siebel eScript:

```plaintext
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.
Unique Names

Make sure that every function you create has a unique name. If two functions on the same view have the same name, results are unpredictable. Good coding practice is to make sure all such names are unique. Consider using a naming convention, such as using the view name as a function name prefix.

Using Tracing to Find Out 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). For Siebel eScript the syntax resembles the following:

```csharp
TheApplication().TraceOn("filename, type, selection");
TheApplication().TraceOn(" Event_Name has fired.");
```

For Siebel VB the syntax resembles the following:

```csharp
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.

```csharp
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 multi-value field is through the underlying MVG business component. If the multi-value 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 multi-value field is if the field is updated within the MVG applet. If the user makes a change to the multi-value 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 14 and Table 15 list BusComp events.

Table 14. Server Side BusComp Events

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>“BusComp_Associate Event” on page 257</td>
</tr>
<tr>
<td>“BusComp_ChangeRecord Event” on page 258</td>
</tr>
<tr>
<td>“BusComp_PreCopyRecord Event” on page 262</td>
</tr>
<tr>
<td>“BusComp_CopyRecord Event” on page 259</td>
</tr>
<tr>
<td>“BusComp_InvokeMethod Event” on page 261</td>
</tr>
<tr>
<td>“BusComp_NewRecord Event” on page 261</td>
</tr>
<tr>
<td>“BusComp_PreAssociate Event” on page 262</td>
</tr>
<tr>
<td>“BusComp_PreDeleteRecord Event” on page 263</td>
</tr>
<tr>
<td>“BusComp_PreGetFieldValue Event” on page 264</td>
</tr>
<tr>
<td>“BusComp_PreInvokeMethod Event” on page 265</td>
</tr>
<tr>
<td>“BusComp_PreNewRecord Event” on page 266</td>
</tr>
<tr>
<td>“BusComp_PreQuery Event” on page 266</td>
</tr>
<tr>
<td>“BusComp_PreSetFieldValue Event” on page 267</td>
</tr>
<tr>
<td>“BusComp_PreWriteRecord Event” on page 269</td>
</tr>
<tr>
<td>“BusComp_Query Event” on page 270</td>
</tr>
<tr>
<td>“BusComp_SetFieldValue Event” on page 272</td>
</tr>
<tr>
<td>“BusComp_WriteRecord Event” on page 272</td>
</tr>
</tbody>
</table>

Table 15. Browser Side BusComp Events

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>“BusComp_PreSetFieldValue Event” on page 267</td>
</tr>
</tbody>
</table>
Applet Events

Events are invoked in response to user interactions. These can be managed for each applet. Applet events are only supported in high interactivity mode. Table 16 and Table 17 list the User interface events.

Table 16. Server-side Applet Events

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;WebApplet_InvokeMethod Event&quot; on page 109</td>
</tr>
<tr>
<td>&quot;WebApplet_Load Event&quot; on page 110</td>
</tr>
<tr>
<td>&quot;WebApplet_PreCanInvokeMethod Event&quot; on page 112</td>
</tr>
<tr>
<td>&quot;WebApplet_PreInvokeMethod Event&quot; on page 113</td>
</tr>
</tbody>
</table>

Table 17. Browser-side Applet Events

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Applet_ChangeFieldValue Event&quot; on page 103</td>
</tr>
<tr>
<td>&quot;Applet_ChangeRecord Event&quot; on page 105</td>
</tr>
<tr>
<td>&quot;Applet_InvokeMethod Event&quot; on page 106</td>
</tr>
<tr>
<td>&quot;Applet_PreInvokeMethod Event&quot; on page 108</td>
</tr>
</tbody>
</table>
**Application Events**

Application events are listed in Table 18 and Table 19.

Table 18. Server-side Application Events

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Application_InvokeMethod Event” on page 176</td>
<td></td>
</tr>
<tr>
<td>“Application_Navigate Event” on page 177</td>
<td></td>
</tr>
<tr>
<td>“Application_PreInvokeMethod Event” on page 177</td>
<td></td>
</tr>
<tr>
<td>“Application_PreNavigate Event” on page 179</td>
<td></td>
</tr>
<tr>
<td>“Application_Start Event” on page 180</td>
<td></td>
</tr>
</tbody>
</table>

Table 19. Browser-side Application Events

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Application_InvokeMethod Event” on page 176</td>
<td></td>
</tr>
<tr>
<td>“Application_PreInvokeMethod Event” on page 177</td>
<td></td>
</tr>
</tbody>
</table>

**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 follows:

```
host="siebel[.transport][.encryption][.compression]://host[:port]/
EnterpriseServer/AppObjMgr_lang" lang="lang_code"
```

The following is an example of a connect string. **Siebel Application** is an application instance:

```
Siebel Application.Login "host="siebel://host/EnterpriseServer/SCCObjMgr_enu"
"lang=ENU", "CCONWAY", "CCONWAY"
```

Note that the entire protocol string is optional. You can specify the transport protocol alone and separate it from **siebel** with a single period:

```
siebel.tcpip://host/siebel/AppObjMgr_lang
```

However, if you specify any of the other protocols, you must use a period as a placeholder for each protocol not specified. The following is an example:

```
siebel...zlib://myhost/siebel/SCCObjMgr_enu
```

Protocols that are not specified receive their default values, as shown in Table 20 on page 75.
Make the following substitutions for the placeholders in the example:

Table 20. Placeholder Substitutions When Logging into a Siebel Server

<table>
<thead>
<tr>
<th>In Place Of</th>
<th>Insert</th>
</tr>
</thead>
<tbody>
<tr>
<td>transport</td>
<td>The default value, tcpip, or leave blank</td>
</tr>
<tr>
<td>encryption</td>
<td>One of the following values:</td>
</tr>
<tr>
<td></td>
<td>■ none (default)</td>
</tr>
<tr>
<td></td>
<td>■ msckrypto (not supported by Java Data Bean)</td>
</tr>
<tr>
<td></td>
<td>■ rsa (supported by Java Data Bean)</td>
</tr>
<tr>
<td>compression</td>
<td>One of the following values:</td>
</tr>
<tr>
<td></td>
<td>■ none</td>
</tr>
<tr>
<td></td>
<td>■ zlib (the default)</td>
</tr>
<tr>
<td>host</td>
<td>The name of the computer on which the Siebel Server is installed</td>
</tr>
<tr>
<td>port</td>
<td>The SCBroker port; by default 2321. This changes only if the Siebel administrator changes the default during installation. For information about load-balancing with SCBroker, see Siebel Deployment Planning Guide, Siebel System Administration Guide, and Siebel Installation Guide for the operating system you are using.</td>
</tr>
<tr>
<td>EnterpriseServer</td>
<td>The name of the Siebel Enterprise Server</td>
</tr>
<tr>
<td>AppObjMgr</td>
<td>The name of the defined Application Object Manager that you want the Web client to access; this can be a user-defined component or one of these predefined components:</td>
</tr>
<tr>
<td></td>
<td>■ ISSObjMgr_&lt;lang&gt;</td>
</tr>
<tr>
<td></td>
<td>■ SCCObjMgr_&lt;lang&gt;</td>
</tr>
<tr>
<td></td>
<td>■ SSEObjMgr_&lt;lang&gt;</td>
</tr>
<tr>
<td></td>
<td>■ SSVObjMgr_&lt;lang&gt;</td>
</tr>
<tr>
<td></td>
<td>For more information, see Siebel System Administration Guide.</td>
</tr>
</tbody>
</table>

For more information about the Login method, see “Login Method” on page 148.

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

```sql
'COM Data Control : SERVER Mode
lstr = "host=" + ""siebel://frashid/Siebel/SSEobjMgr_enu"
'Format of the connect string is
"host=" + ""siebel://<host>/<Enterprise>/<App. Object Mgr_lang>""
lng = "lang=" + ""ENU"
retVal = siebDataCtl>Login(lng + lstr, "username", "password")
```
The following 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
lng = "lang=" + "ENU"
retval = siebDataCtl.Login(lng + lstr, "username", "password")
```

The following 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_enu/
SIEBEL_SERVER" + char(34) + " Lang = " + char(34) + "LANG" + char(34)
```

If you are using a single sign-on (SSO) with Java Data Bean, you must use the login ID of an employee as the username and the value of the trust token (the TrustToken parameter in the application’s CFG file) in the connect string to log into the server.

For example, the connect string would be:

```
mdataBean.login("Siebel://gatewayserver:2321/enterpriseServer/SCCObjMgr_enu",
SADMIN, HELLO,"enu");
```

where SADMIN is an employee and TrustToken = HELLO in the [LDAPSecAdpt] section of uagent.cfg.

### Using Load Balancing with the Connect String

Siebel COM Data Control operating in server mode and Java Data Bean support Siebel native load balancing across Siebel Servers. The standard connect string is modified to direct requests to an appropriate virtual host that includes specific Siebel Servers with the desired object manager, and to provide the path to the file that defines the virtual host.

Connect strings that use Siebel native load balancing have the following structures:

- **COM Data Control:**
  
  ```
  host ="siebel://VirtualHost/EnterpriseServer/AppObjMgr_lang"vhosts="<path to lbconfig.txt>"
  ```

  where lbconfig.txt is the file that defines virtual hosts.

  For information on lbconfig.txt definition of virtual hosts, see *Siebel System Administration Guide*.

- **Java Data Bean:**
host="siebel://VirtualHost/EnterpriseServer/AppObjMgr_lang"

When using generated code, by default, virtual host definitions are read from the siebel.conmgr.virtualhosts property in the siebel.properties file. The siebel.properties file must be in the classpath of the Java Virtual Machine.

For information on definition of virtual hosts in siebel.properties, see Transports and Interfaces: Siebel Enterprise Application Integration.

The following is a sample connect string for the COM Data Control operating in Server Mode in an environment that implements Siebel round-robin load balancing across Siebel Servers:

'COM Data Control : Load Balancing
lstr = "host=" + ""siebel://VirtualServer1/Siebel/SSEObjMgr_enu"" + "vhosts=" + "m:siebel\admin\lbconfig.txt"
lng = "lang=" + "ENU"
retval = siebDataCtl.Login(lng + lstr, "username", "password")

where VirtualServer1 matches the VirtualServer parameter in the session manager rules in lbconfig.txt, for example:

"VirtualServer1=1:SiebServA:2321;2:SiebServB:2321;"

For information on the structure of the lbconfig.txt file, see Siebel System Administration Guide.

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 Web Client Automation Server, Web Client Automation Server, or Java Data Bean. The following examples illustrate the difference between calling a COM Data Server interface method and calling the version of the method that is compatible with COM Data Control and Mobile Web Client Automation Server.

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 Server

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:

```java
import com.siebel.data.SiebelException;
import com.siebel.data.SiebelDataBean;
...
SiebelDataBean mySiebelBean=null;
try {
    mySiebelBean = new SiebelDataBean();
    mySiebelBean.login("Siebel://SOMSERVER/somsiebel/AppObjMgr/", "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, you can use either exceptions or methods. The following methods are available:

- EnableExceptions
- GetLastErrCode
- GetLastErrText

#### 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.
GetLastErrorCode, GetLastErrorText Method

After execution of a method, the GetLastErrorCode can be invoked to check if any error was returned from the previous operation. The GetLastErrorText method can be invoked to retrieve the text of the error message, for example:

```plaintext
GetLastErrorCode() ' retrieves errorCode As Integer
GetLastErrorText() ' retrieves text As String
```
This chapter lists the methods and events available to Siebel object interfaces:

- "Object Interface Method Tables" on page 81
- "Object Interface Event Tables" on page 93
- "Siebel Constants" on page 95
- "Applet Methods" on page 96
- "Applet Events" on page 103
- "Application Methods" on page 119
- "Application Events" on page 175
- "Business Component Methods" on page 181
- "Business Component Events" on page 256
- "Business Object Methods" on page 273
- "Business Service Methods" on page 278
- "Business Service Events" on page 287
- "Control Methods" on page 293
- "Property Set Methods" on page 302
- "Miscellaneous Methods" on page 320

**Object Interface Method Tables**

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

- "Applet Methods" on page 82
- "Application Methods" on page 82
- "Business Component Methods" on page 85
- "Business Object Methods" on page 89
- "Business Service Methods" on page 89
- "Control Methods" on page 90
- "Property Set Methods" on page 91
- "Miscellaneous Methods" on page 92
### Applet Methods

Table 21 lists the applet methods.

Table 21. Applet Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Web Client Automation Server</th>
<th>Mobile Web Client Automation Server</th>
<th>Siebel COM Data Control</th>
<th>COM Data Server</th>
<th>Java Data Bean</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveMode Method</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusObject Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FindActiveXControl Method</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FindControl Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Application Methods

Table 22 lists the application methods.

Table 22. Application Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Web Client Automation Server</th>
<th>Mobile Web Client Automation Server</th>
<th>Siebel COM Data Control</th>
<th>COM Data Server</th>
<th>Java Data Bean</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveApplet Method</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ActiveBusComp Method</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ActiveBusObject Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ActiveViewName Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 22. Application Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Server</th>
<th>Browser</th>
<th>Web Client</th>
<th>Mobile Web</th>
<th>Siebel COM</th>
<th>COM</th>
<th>Java Data</th>
<th>Bean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach Method</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CurrencyCode Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detach Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EnableExceptions Method</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FindApplet Method</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetBusObject Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>GetDataSource Method Called only with InvokeMethod</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetLastErrCode Method</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetLastErrText Method</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetProfileAttr Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetService Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetSharedGlobal Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GotoView Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsViewReadOnly Method Called only with InvokeMethod</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language Method Called only with InvokeMethod</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 22. Application Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Web Client Automation Server</th>
<th>Mobile Web Client Automation Server</th>
<th>Siebel COM Data Control</th>
<th>COM Data Server</th>
<th>Java Data Bean</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoadObjects Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LoadUserAttributes Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Login Method</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>LoginId Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>LoginName Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Logoff Method</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>LookupMessage Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LookupValue Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name Method</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NewPropertySet Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PositionId Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PositionName Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RaiseError Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RaiseErrorText Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetPositionId Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SetPositionName Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SetProfileAttr Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SetSharedGlobal Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Table 22. Application Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Web Client Automation Server</th>
<th>Mobile Web Client Automation Server</th>
<th>Siebel COM Data Control</th>
<th>COM Data Server</th>
<th>Java Data Bean</th>
</tr>
</thead>
<tbody>
<tr>
<td>“ShowModalDialog Method”</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SWEAlert Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trace Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TraceOff Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TraceOn Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 23 lists the business component methods.

Table 23. Business Component Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Web Client Automation Server</th>
<th>Mobile Web Client Automation Server</th>
<th>Siebel COM Data Control</th>
<th>COM Data Server</th>
<th>Java Data Bean</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActivateField Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ActivateMultipleFields Method</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Associate Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusObject Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ClearLOVCache Method</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Called only with InvokeMethod</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ClearToQuery Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CreateFile Method</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Called only with InvokeMethod</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeactivateFields Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 23. Business Component Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Web Client Automation Server</th>
<th>Mobile Web Client Automation Server</th>
<th>Siebel COM Data Control</th>
<th>COM Data Server</th>
<th>Java Data Bean</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeleteRecord Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ExecuteQuery Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ExecuteQuery2 Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>FirstRecord Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>FirstSelected Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GenerateProposal Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Called only with InvokeMethod</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetAssocBusComp Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GetFieldValue Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetFile Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Called only with InvokeMethod</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetFormattedFieldValue Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GetLastErrCode Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetLastErrText Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetMultipleFieldValues Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GetMVGBusComp Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetNamedSearch Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GetPicklistBusComp Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 23. Business Component Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Web Client Automation Server</th>
<th>Mobile Web Client Automation Server</th>
<th>Siebel COM Data Control</th>
<th>COM Data Server</th>
<th>Java Data Bean</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetSearchExpr Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetSearchSpec Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetSortSpec Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetUserProperty Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetViewMode Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LastRecord Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NewRecord Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NextRecord Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NextSelected Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ParentBusComp Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pick Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PreviousRecord Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PutFile Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Called only with InvokeMethod</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RefineQuery Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RefreshBusComp Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Called only with InvokeMethod</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 23. Business Component Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Web Client Automation Server</th>
<th>Mobile Web Client Automation Server</th>
<th>Siebel COM Data Control</th>
<th>COM Data Server</th>
<th>Java Data Bean</th>
</tr>
</thead>
<tbody>
<tr>
<td>RefreshRecord Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Called only with InvokeMethod</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release Method</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>SetAdminMode Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Called only with InvokeMethod</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetFieldValue Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SetFormattedFieldValue Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SetMultipleFieldValues Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SetNamedSearch Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SetSearchExpr Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SetSearchSpec Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SetSortSpec Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SetUserProperty Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SetViewMode Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>UndoRecord Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>WriteRecord Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
## Business Object Methods

Table 24 lists the business object methods.

### Table 24. Business Object Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Web Client Automation Server</th>
<th>Mobile Web Client Automation Server</th>
<th>Siebel COM Data Control</th>
<th>COM Data Server</th>
<th>Java Data Bean</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetBusComp Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>GetLastErrCode Method</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>GetLastErrText Method</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Name Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Release Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Business Service Methods

Table 25 lists the business service methods.

### Table 25. Business Service Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Web Client Automation Server</th>
<th>Mobile Web Client Automation Server</th>
<th>Siebel COM Data Control</th>
<th>COM Data Server</th>
<th>Java Data Bean</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetFirstProperty Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>GetNextProperty Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>GetProperty Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Name Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>PropertyExists Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Release Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Control Methods

Table 26 lists the control methods.

Table 26. Control Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Web Client Automation Server</th>
<th>Mobile Web Client Automation Server</th>
<th>Siebel COM Data Control</th>
<th>COM Data Server</th>
<th>Java Data Bean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applet Method</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp Method</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetProperty Method</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetValue Method</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name Method</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetLabelProperty Method</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetProperty Method</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetValue Method</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Property Set Methods

Table 27 lists the property set methods.

### Table 27. Property Set Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Web Client Automation Server</th>
<th>Mobile Web Client Automation Server</th>
<th>Siebel COM Data Control</th>
<th>COM Data Server</th>
<th>Java Data Bean</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddChild Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Copy Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetByteValue Method</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GetChild Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetChildCount Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetFirstProperty Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetLastErrCode Method</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetLastErrText Method</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetNextProperty Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetProperty Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetPropertyCount Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetType Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetValue Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
### Miscellaneous Methods

Table 28 lists miscellaneous methods.

**Table 28.  Miscellaneous Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Web Client Automation Server</th>
<th>Mobile Web Client Automation Server</th>
<th>Siebel COM Data Control</th>
<th>COM Data Server</th>
<th>Java Data Bean</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetErrorCode Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetErrorMessage Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TheApplication Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Object Interface Event Tables

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 Events”
- ”Application Events” on page 94
- ”Business Component Events” on page 94
- ”Business Service Events” on page 95

Applet Events

Table 29 lists the applet events.

Table 29. Applet Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applet_ChangeFieldValue Event</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Applet_ChangeRecord Event</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Applet_InvokeMethod Event</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Applet_Load Event</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Applet_PreInvokeMethod Event</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>WebApplet_InvokeMethod Event</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>WebApplet_Load Event</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>WebApplet_PreCanInvokeMethod Event</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>WebApplet_PreInvokeMethod Event</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>WebApplet_ShowControl Event</td>
<td></td>
<td>X</td>
<td>Not available in high interactivity mode</td>
</tr>
<tr>
<td>WebApplet_ShowListColumn Event</td>
<td></td>
<td>X</td>
<td>Not available in high interactivity mode</td>
</tr>
</tbody>
</table>
Application Events

Table 30 lists the application events.

Table 30. Application Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application_Close Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application_InvokeMethod Event</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Application_Navigate Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application_PreInvokeMethod Event</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Application_PreNavigate Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application_Start Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Business Component Events

Table 31 lists the application events.

Table 31. Business Component Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusComp_Associate Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp_ChangeRecord Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp_CopyRecord Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp_DeleteRecord Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp_InvokeMethod Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp_NewRecord Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp_PreAssociate Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp_PreCopyRecord Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp_PreDeleteRecord Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp_PreGetFieldValue Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp_PreInvokeMethod Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp_PreNewRecord Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp_PreQuery Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 31. Business Component Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusComp_PreSetFieldValue Event</td>
<td>X</td>
<td>X</td>
<td>Available only in high interactivity mode. Requires a field property to be set for the event to be immediately executed on the server.</td>
</tr>
<tr>
<td>BusComp_PreWriteRecord Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp_Query Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp_SetFieldValue Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp_WriteRecord Event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Business Service Events

Table 32 lists the business service events.

Table 32. Business Service Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Server Script</th>
<th>Browser Script</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service_InvokeMethod Event</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Service_PreCanInvokeMethod Event</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Service_PreInvokeMethod Event</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Siebel Constants

The Siebel programming languages provide constants for the convenience of programmers. These constants are listed in Table 33. 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. The integer values are included only to aid in debugging. A constant’s integer value appears in the Debugger when the constant is stored in a local variable and the value of the local variable is exposed.

Table 33. Siebel Constants

<table>
<thead>
<tr>
<th>Used With</th>
<th>Constant Name</th>
<th>Integer Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre Event Handler Methods</td>
<td>ContinueOperation</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>CancelOperation</td>
<td>2</td>
</tr>
</tbody>
</table>
Table 33. Siebel Constants

<table>
<thead>
<tr>
<th>Used With</th>
<th>Constant Name</th>
<th>Integer Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search Methods</td>
<td>ForwardBackward</td>
<td>256</td>
</tr>
<tr>
<td></td>
<td>ForwardOnly</td>
<td>257</td>
</tr>
<tr>
<td>NewRecord Method</td>
<td>NewBefore</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>NewAfter</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>NewBeforeCopy (Not available with Java Data Bean)</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>NewAfterCopy (Not available with Java Data Bean)</td>
<td>3</td>
</tr>
<tr>
<td>Siebel ViewMode Methods</td>
<td>SalesRepView</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ManagerView</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>PersonalView</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>AllView</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>OrganizationView</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>GroupView</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>CatalogView</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>SubOrganizationView</td>
<td>9</td>
</tr>
</tbody>
</table>

Applet Methods

In the following methods, the placeholder Applet in the syntax represents an applet instance:

- "ActiveMode Method"
- “BusComp Method” on page 97
- “BusObject Method” on page 98
- “FindActiveXControl Method” on page 98
- “FindControl Method” on page 99
- “InvokeMethod Method” on page 100
- “Name Method” on page 102

ActiveMode Method

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

*Applet.ActiveMode*

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

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

**Used With**

Browser Script

**Example**

The following example is in Browser Script:

```javascript
function Applet_Load ()
{
    var currMode = this.ActiveMode();
    theApplication().SWEAlert("The active mode for the selected applet is: " + currMode);
}
```

**BusComp Method**

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

**Syntax**

*Applet.BusComp();*

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

The business component associated with the applet.

**Used With**

Browser Script, Server Script
**BusObject Method**

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

**Syntax**

```javascript
Applet.BusObject()
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**
The business object for the applet’s business component.

**Used With**
Browser Script, Server Script

**Example**
The following example is in Browser Script:

```javascript
function Applet_Load ()
{
    var appletname = this.Name();
    var currBO = this.BusObject();
    var currBOName = currBO.Name();
    theApplication().SWEAlert('The active Business Object for the " + appletname + 
    " is: " + currBOName);
}
```

The following example is in Siebel eScript:

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

The following example is in Siebel VB:

```vbnet
Sub WebApplet_Load
    Dim oBusObject As BusObject
    Set oBusObject = Me.BusObject
End Sub
```

**FindActiveXControl Method**

FindActiveXControl returns a reference to a DOM element based upon the name specified in the name argument.
Syntax
Applet.FindActiveXControl(controlName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlName</td>
<td>Literal string or string variable containing the name of the desired control</td>
</tr>
</tbody>
</table>

Returns
The control object identified in controlName.

Usage
FindActiveXControl is used for finding controls on form applets. It does not locate list columns on list applets.

Used With
Browser Script

Example
The following Browser Script example interacts with a Microsoft slide ActiveX control that has been placed on a Siebel applet:

```javascript
// Get a reference to the control
var SlideCtrl = FindActiveXControl("SliderControl");

// Display some of the ActiveX Control's properties
theApplication().SWEAlert("element id = " + SlideCtrl.id);
theApplication().SWEAlert("Max ticks = " + SlideCtrl.Max);
SlideCtrl.SelStart = 2;  // Set a control property
SlideCtrl.Refresh();    // Call the control's Refresh method

var myCustomCtrl = FindActiveXControl("TestControl");
myCustomCtrl.TestProperty01 = "abc";
myCustomCtrl.Style.visible = "hidden"; // Use a Style sheet property
```

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

Applet.FindControl(controlName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlName</td>
<td>Literal string or string variable containing the name of the desired control</td>
</tr>
</tbody>
</table>

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. It also does not locate list columns on list applets.

Used With

Browser Script

Example

To use this Browser Script example, read the notes for the "SetLabelProperty Method" on page 297.

```
function Applet_PreInvokeMethod (name, inputPropSet)
{
    // Code to change the Font Size of the "Location" label
    if (name == "fontsize")
    {
        // Use FindControl() to get a reference to the control
        var ctl = this.FindControl("Location");

        // Set the font size
        ctl.SetLabelProperty("FontSize", "22");
    }
    return ('CancelOperation');
}
```

InvokeMethod Method

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

Browser Script Syntax

Applet.InvokeMethod(methodName, methodArgs_PropSet);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>methodName</td>
<td>The name of the method</td>
</tr>
<tr>
<td>methodArgs_PropSet</td>
<td>Property set containing the method arguments</td>
</tr>
</tbody>
</table>
Server Script Syntax

`Applet.InvokeMethod(methodName, methodArgs);`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>methodName</td>
<td>The name of the method</td>
</tr>
<tr>
<td>methArg1, methArg2, ..., methArgN</td>
<td>One or more strings containing arguments to <code>methodName</code></td>
</tr>
</tbody>
</table>

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 only be used with documented methods. Oracle does not support calling methods with `InvokeMethod`, unless they are listed in this book. Calling `InvokeMethod` with an undocumented method is not supported. Undocumented methods may be modified or obsoleted without notice. Use of undocumented methods is entirely at your own risk.

Used With

Browser Script, Server Script

Example

The following example is in Siebel eScript:

```javascript
function WebApplet_PreInvokeMethod (MethodName)
{
    //Invoke a Siebel SmartScript from a custom button
    //using the applet.InvokeMethod method
    //Note the InvokeSScriptFromButton is from a custom
    //method added to a button
    if (MethodName == "InvokeSScriptFromButton")
    {
        var iReturn = ContinueOperation;
        var sArgs = new Array(3);
        sArgs[0] = "Demo Opportunity Profile";
        sArgs[1] = "";
        sArgs[2] = "";
        this.InvokeMethod("RunCallScript", sArgs);
        iReturn = CancelOperation;
    }
    else
    {
        iReturn = ContinueOperation;
    }
```
Name Method

The Name method returns the name of the applet.

Syntax

Applet.Name()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns

A string containing the applet object name.

Used With

Browser Script, Server Script

Example

The following example is in Browser Script:

```javascript
function Applet_Load ()
{
    // Display the name of the applet when the applet loads using the
    // applet.Name() method to obtain the name of the applet
    var appletName;
    appletName = this.Name();
    theApplication().SWEAlert("The name of the applet is: " + appletName);
}
```

The following example is in Siebel eScript:

```javascript
function WebApplet_Load ()
{
    // Display the name of the applet when the applet loads using the
    // applet.Name() method to obtain the name of the applet
    var appletName;
    appletName = this.Name();
    TheApplication().RaiseErrorText("The name of the applet is: " + appletName);
}
```

The following example is in Siebel VB:
Sub WebApplet_Load
' Display the name of the applet when the applet loads using the
' applet.Name() method to obtain the name of the applet
Dim appletName As String
appletName = Me.Name
TheApplication.RaiseErrorText 'The name of the applet is: ' & appletName
End Sub

Applet Events
The following topics describe applet events:

- "Applet_ChangeFieldValue Event" on page 103
- "Applet_ChangeRecord Event" on page 105
- "Applet_InvokeMethod Event" on page 106
- "Applet_Load Event" on page 107
- "Applet_PreInvokeMethod Event" on page 108
- "WebApplet_InvokeMethod Event" on page 109
- "WebApplet_Load Event" on page 110
- "WebApplet_PreCanInvokeMethod Event" on page 112
- "WebApplet_PreInvokeMethod Event" on page 113
- "WebApplet_ShowControl Event" on page 114
- "WebApplet_ShowListColumn Event" on page 116

Applet_ChangeFieldValue Event
The ChangeFieldValue event fires after the data in a field changes through the applet in the user interface.

Syntax
Applet_ChangefieldValue(fieldname, fieldValue)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>A string representing the name of the field whose value changed</td>
</tr>
<tr>
<td>fieldValue</td>
<td>A string representing the new value assigned to FieldName</td>
</tr>
</tbody>
</table>

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.

**NOTE:** This event does not trigger for changes made in pick applets or popup applets.

**Used With**

Browser Script

**Example**

The following example is in Browser Script:

```javascript
function Applet_ChangeFieldValue (field, value)
{
  try
  {
    switch (field)
    {
      case "Primary Revenue Committed Flag":
        if (value == "Y")
        {
          var thisBC = this.BusComp();
          var sRev = thisBC.GetFieldValue("Primary Revenue Amount");
          var sUpside = thisBC.GetFieldValue("Primary Revenue Upside Amount");
          var total = sRev + sUpside;
          if (total < 500000)
          {
            thisBC.SetFieldValue("Primary Revenue Committed Flag", "N");
            theApplication().SWEAlert("Changing the Committed Flag to NO as $500,000 in Revenue and Upside amount is required");
          }
          break;
        }
    }
  }
  catch(e)
  {
    theApplication().SWEAlert("Error in ChangeFieldValue and error is " + e.toString() + " + e.errText()");
  }
}
```

**Related Topic**

"Applet_ChangeRecord Event"
Applet_ChangeRecord Event

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

Syntax
Applet_ChangeRecord()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns
Not applicable

Used With
Browser Script

Example
The following example is in Browser Script:

```javascript
function Applet_ChangeRecord ()
{
    try
    {
        var thisBC = this.BusComp();
        var sFlag = thisBC.GetFieldValue("Primary Revenue Committed Flag");
        if (sFlag == "Y")
        {
            theApplication().SWEAlert("This record cannot be updated because it has been commited");
        }
    }
    catch(e)
    {
        theApplication().SWEAlert("Error in ChangeFieldValue and error is " + e.toString() + " + e.errText()");
    }
}
```

**NOTE:** To return the value of the field to which you are navigating in the Applet_ChangeRecord event, use the BusComp.GetFieldValue() method. The control.GetValue() method returns data from the record you are leaving.

Related Topic
"Applet_ChangeFieldValue Event” on page 103
Applet_InvokeMethod Event

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

Syntax
Applet_InvokeMethod(name, inputPropSet)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the method that is triggered.</td>
</tr>
<tr>
<td>inputPropSet</td>
<td>A property set containing arguments to be passed to the InvokeMethod event.</td>
</tr>
</tbody>
</table>

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

Example
Some special methods create, modify, or delete records. In some cases, events at the applet or business component level are triggered by these actions. If there is a requirement to perform a specific action before and after the method has been executed, these events can be used. In this example, code has been added to the PreInvokeMethod and InvokeMethod applet events to set and reset the flag and to the NewRecord server event to set the fields.

```javascript
function Applet_PreInvokeMethod (name, inputPropSet)
{
    if (name == "Quote")
    {
        // Add code that needs to be executed BEFORE the special method
        // Set flag to "1"
        TheApplication().SetProfileAttr("flag","1");
    }
    return "ContinueOperation";
}
```
function Applet_InvokeMethod (name, inputPropSet)
{
    if (name == "Quote")
    {
        // Add code that needs to be executed AFTER the special method
        // Reset the flag to '0'
        TheApplication().SetProfileAttr("flag","0");
    }
}

function BusComp_NewRecord ()
{
    if (TheApplication().GetProfileAttr("flag") == "1")
    {
        this.SetFieldValue("Field1", "Value1");
        this.SetFieldValue("Field2", "Value2");
        ........
    }
}

Related Topics
"Applet_PreInvokeMethod Event" on page 108
"Application_InvokeMethod Event" on page 176

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

Syntax
Applet_Load()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns
Not applicable
Usage
You can use this event with form applets 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.

NOTE: Do not use the SWEAlert or RaiseErrorText methods in this event to display a popup. This can cause the browser to fail if the application has not yet been fully rendered in the browser.

Used With
Browser Script

Example
Use this event to dynamically hide or manipulate controls or set properties on a control. The following controls can be dynamically modified: CheckBox, ComboBox, Label, TextArea, and TextBox.

NOTE: This example is only applicable to code on form applets.

```javascript
function Applet_Load ()
{
    // Get the control instance.
    var ctrl = this.FindControl("FirstName");

    // Hide the control
    ctrl.SetProperty("Visible", "false");

    // Hide the label
    ctrl.SetLabelProperty("Visible", "hidden");
}
```

Applet_PreInvokeMethod Event
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)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputPropSet</td>
<td>A property set containing arguments to be passed to the PreInvokeMethod event</td>
</tr>
</tbody>
</table>

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 do not 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

**Example**

```javascript
function Applet_PreInvokeMethod (name, inputPropSet)
{
  if(name == 'NewRecord')
  {
    if(confirm("Are you sure you want to create a new record?"))
      return ("ContinueOperation");
    else
      return ("CancelOperation");
  return ("ContinueOperation");
}
```

**Related Topic**

“How Your Script Affects Program Flow” on page 68

**WebApplet_InvokeMethod Event**

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*)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>methodName</em></td>
<td>String variable or literal containing the name of the method invoked.</td>
</tr>
</tbody>
</table>
**Returns**
Not applicable

**Used With**
Server Script

**Example**
The following example is in Siebel eScript:

```javascript
switch (MethodName)
{
    case "NewQuery":
        TheApplication().SetSharedGlobal("EnableButton", "N"); break;
    case "ExecuteQuery":
        TheApplication().SetSharedGlobal("EnableButton", ""); break;
    case "UndoQuery":
        TheApplication().SetSharedGlobal("EnableButton", "");
        break;
}
```

The following example is in Siebel VB:

```vbnet
Select Case MethodName
Case "NewQuery"
    TheApplication.SetSharedGlobal "EnableButton", "N"
    break
Case "ExecuteQuery"
    TheApplication.SetSharedGlobal "EnableButton", ""
    break
Case "UndoQuery"
    TheApplication.SetSharedGlobal "EnableButton", ""
    break
End Select
```

**Related Topics**
"Applet_InvokeMethod Event” on page 106
"Application_InvokeMethod Event” on page 176
"WebApplet_PreCanInvokeMethod Event” on page 112

**WebApplet_Load Event**
The Load event is triggered just after an applet is loaded.
**Syntax**

WebApplet_Load()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**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

**Example**

The following example is in Siebel eScript:

```javascript
function WebApplet_Load ()
{
    try
    {
        var currBC = this.BusComp();
        with (currBC)
        {
            SetViewMode(OrganizationView);
            ClearToQuery();
            SetSearchSpec("Last Name", "A*");
            ExecuteQuery(ForwardBackward);
        }
    }
    catch (e)
    {
        TheApplication().RaiseErrorText(e.errText);
    }
}
```

The following example is in Siebel VB:

```vbscript
Sub WebApplet_Load
    Dim iReturn As Integer
    Dim currBC As BusComp
    Set currBC = Me.BusComp
    With currBC
        .SetViewMode OrganizationView
        .ClearToQuery
        .SetSearchSpec "Last Name", "A*"
    End With
End Sub
```
WebApplet_PreCanInvokeMethod Event

The PreCanInvokeMethod event is called under the following conditions, allowing the script to determine whether or not the user has the authority to invoke the applet method:

- Before the PreInvokeMethod event is called
- When the user steps to a different record
- When an applet is loaded
- When a different method, such as GetProfileAttr() or SetProfileAttr(), is called from Browser Script

**NOTE:** It is often easier to enable and disable methods at the applet level declaratively, using the CanInvokeMethod applet user property. For an example of this, see "Invoking Custom Methods with MiniButton Controls" on page 431. For more information on the CanInvokeMethod user property, see *Siebel Developer’s Reference*.

**Syntax**

WebApplet_PreCanInvokeMethod(MethodName, &CanInvoke)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MethodName</td>
<td>A string representing the name of the method to be executed.</td>
</tr>
<tr>
<td>&amp;CanInvoke</td>
<td>A string representing whether or not the Applet method can be invoked. Valid values are TRUE or FALSE.</td>
</tr>
</tbody>
</table>

**NOTE:** Using the FirstSelected business component method with the PreCanInvokeMethod event can cause unexpected behavior in pick applets invoked from the applet where this event is called.

**Returns**

CancelOperation or ContinueOperation

**Used With**

Server Script
Example
The following example is in Siebel eScript:

```javascript
function WebApplet_PreCanInvokeMethod (MethodName, &CanInvoke)
{
    if ( MethodName == "CustomMethod" )
    {
        CanInvoke = "TRUE";
        return( CancelOperation );
    } else
    return (ContinueOperation);
}
```

The following example is in Siebel VB:

```vbscript
Function WebApplet_PreCanInvokeMethod (MethodName As String, CanInvoke As String) As Integer
    Dim iReturn As Integer
    iReturn = ContinueOperation
    If MethodName = "Test" Then
        CanInvoke = "TRUE"
        iReturn = CancelOperation
    End If
    WebApplet_PreCanInvokeMethod = iReturn
End Function
```

WebApplet_PreInvokeMethod Event

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)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>methodName</code></td>
<td>String variable or literal containing the name of the method invoked</td>
</tr>
</tbody>
</table>

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 do not 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

**Example**
The following example is in Siebel eScript:

```javascript
function WebApplet_PreInvokeMethod (MethodName)
{
    switch (MethodName)
    {
        case "CustomMethod":
            var applet = this;
            var BC = applet.BusComp();
            var ConId = BC.GetFieldValue("Contact Id");
            var WshShell = COMCreateObject("WScript.Shell");
            WshShell.Popup("My Custom Method was called. Here is the ID " + ConId);
            return(CancelOperation);
            break;
    }
    return (ContinueOperation);
}
```

The following example is in Siebel VB:

```vbnet
Function WebApplet_PreInvokeMethod (MethodName As String) As Integer
    Dim iReturn As Integer
    iReturn = ContinueOperation
    Select Case MethodName
    Case "CustomMethod"
        Dim oBusComp As BusComp
        Set oBusComp = Me.BusComp
        Dim WshShell As Object
        ConId = oBusComp.GetFieldValue("Contact Id")
        Set WshShell = CreateObject("WScript.Shell")
        WshShell.Popup("My Custom Method was called. Here is the ID " & ConId)
        iReturn = CancelOperation
    End Select
    WebApplet_PreInvokeMethod = iReturn
End Function
```

**WebApplet_ShowControl Event**

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)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlName</td>
<td>A string indicating the name of the control to be rendered.</td>
</tr>
<tr>
<td>property</td>
<td>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.</td>
</tr>
<tr>
<td>mode</td>
<td>The mode of the applet that is being shown; possible modes are:</td>
</tr>
<tr>
<td></td>
<td>■ Base</td>
</tr>
<tr>
<td></td>
<td>■ Edit</td>
</tr>
<tr>
<td></td>
<td>■ New</td>
</tr>
<tr>
<td></td>
<td>■ Query</td>
</tr>
<tr>
<td></td>
<td>■ Sort</td>
</tr>
<tr>
<td>HTML</td>
<td>The HTML generated by the Siebel Web Engine for the swe:control or swe:this tag that triggers this event.</td>
</tr>
</tbody>
</table>

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 argument, 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:

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

■ The <swe:control> tag and <swe:this> tag are used to show a control:

```html
<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 an argument 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:

```javascript
function WebApplet_ShowControl (ControlName, Property, Mode, &HTML)
{
    var BC = this.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 Event**

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 standard interactivity mode.
Syntax
WebApplet_ShowListColumn (columnName, property, mode, HTML)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnName</td>
<td>A string indicating the name of the list column to be rendered</td>
</tr>
<tr>
<td>property</td>
<td>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.</td>
</tr>
<tr>
<td>mode</td>
<td>The mode of the applet that is being shown; possible modes are:</td>
</tr>
<tr>
<td></td>
<td>■ Base</td>
</tr>
<tr>
<td></td>
<td>■ Edit</td>
</tr>
<tr>
<td></td>
<td>■ New</td>
</tr>
<tr>
<td></td>
<td>■ Query</td>
</tr>
<tr>
<td></td>
<td>■ Sort</td>
</tr>
<tr>
<td>HTML</td>
<td>The HTML generated by the Siebel Web Engine for the swe:control or swe:this tag that triggers this event</td>
</tr>
</tbody>
</table>

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 argument, 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 an argument 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:

```vbnet
Sub WebApplet_ShowListColumn (ColumnName As String, Property As String, Mode As String, HTML As String)
    Dim amount as Double
    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
```

The following example is in Siebel eScript:

```javascript
function WebApplet_ShowListColumn (ColumnName, Property, Mode, &HTML)
{
    if ((ColumnName == 'Amount') & (Mode == 'Base') & (Property == 'FormattedHTML'))
    {
        var val = HTML.valueOf();
        if (val < 0)
            HTML = "<FONT Color=Red> " + HTML + " </FONT>";
    }
}
```
Application Methods

The following methods are built-in methods that return the current Siebel Application object instance:

- `TheApplication` when called from Siebel VB within Siebel Tools,
- `TheApplication()` (case-sensitive) when called from Siebel eScript within Siebel Tools
- `theApplication()` (case-sensitive) when called from Browser Script within Siebel Tools

If an Application method applies to one scripting language, then the Syntax definition in the method’s section includes `TheApplication`, `TheApplication()`, or `theApplication()` specifically.

If a method applies to external interfaces or to more than one scripting language, and thus to more than one syntax, then the Syntax definition includes `Application`, which denotes that:

- The applicable construct should be substituted for `Application` in Siebel VB, Siebel eScript, or Browser Script
- The name of an Application instance should be substituted for `Application` when you use external interfaces.

Examples of Application methods used by external interfaces frequently include `SiebelApplication` as the Application instance. You should understand that the examples assume that `SiebelApplication` is instantiated in the script, whether the instantiation statement is included in the example or not.

This section includes documentation for the following Application methods:

- “ActiveApplet Method” on page 120
- “ActiveBusComp Method” on page 121
- “ActiveBusObject Method” on page 122
- “ActiveViewName Method” on page 124
- “Attach Method” on page 125
- “CurrencyCode Method” on page 127
- “Detach Method” on page 128
- “EnableExceptions Method” on page 129
- “FindApplet Method” on page 131
- “GetBusObject Method” on page 131
- “GetDataSource Method” on page 143
- “GetLastErrCode Method” on page 133
- “Get_LastErrText Method” on page 134
- “GetProfileAttr Method” on page 135
- “GetService Method” on page 135
- “GetSharedGlobal Method” on page 138
ActiveApplet Method

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

```javascript
theApplication().ActiveApplet();
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**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

**Example**

```javascript
function Applet_PreInvokeMethod (name, inputPropSet)
{
    switch (name)
    {
    case "Drilldown":
        var activeapplet = theApplication().ActiveApplet();
        var activeappletname = activeapplet.Name();
        theApplication().SWEAlert("Here is the applet we are drilling down from " + activeappletname);
        break;
    }
    return ("ContinueOperation");
}
```

**ActiveBusComp Method**

ActiveBusComp returns the business component associated with the active applet.

**Syntax**

```javascript
theApplication().ActiveBusComp();
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>
**Returns**
The business component associated with the active applet

**Used With**
Browser Script

**Example**
```javascript
function Applet_Load ()
{
    var activeBC = theApplication().ActiveBusComp();
    activeBC = activeBC.Name();
    theApplication().SWEAlert(activeBC);
}
```

**ActiveBusObject Method**
ActiveBusObject returns the business object of the active view.

**Syntax**
`Application.ActiveBusObject`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**
The business object of the active view

**Usage**
Do not use ActiveBusObject in any event handler that may be initiated by the COM Data Server, COM Data Control, or Java Data Bean. If you use ActiveBusObj() you get the business object that exists already (if there is one). If you use GetBusObject() instead, any child Business components are ALWAYS new ones, even if you have some already.

**Used With**
Browser Script, Mobile Web Client Automation Server, Server Script

**Example**
The following example is in Browser Script:
```javascript
function Applet_Load ()
{
    var oBusObj;
```
The following samples show an example of server side script that could be invoked from a custom button on a child applet within a view. The script first checks to see if the Contact business object is active, and if so, retrieves the email address of the currently active parent Contact record. The custom 'SendEmail()' function is then invoked using the Contact's email address. Note that the objects are not destroyed at the end of the script, as they are the ones that are currently active in the user interface.

The following example is in Siebel eScript:

```javascript
function WebApplet_PreInvokeMethod (MethodName)
{
  if (MethodName == "Send Email")
  {
    var oBO = TheApplication().ActiveBusObject();
    if (oBO.Name() == "Contact")
    {
      var oBC = oBO.GetBusComp("Contact");
      var sEmail = oBC.GetFieldValue("Email Address");
      SendMail(sEmail);
      sEmail = ";
    }
    return (CancelOperation);
  }
  return (ContinueOperation);
}
```

The following example is in Siebel VB:

```vbnet
Function WebApplet_PreInvokeMethod (MethodName As String) As Integer
    Dim iRtn As Integer
    iRtn = ContinueOperation
    If MethodName = "Send Email" Then
        Dim oBO As BusObject
        Set oBO = TheApplication.ActiveBusObject()
        If oBO.Name() = "Contact" Then
            Dim oBC As BusComp
            Dim sEmail As String
            Set oBC = oBO.GetBusComp("Contact")
            sEmail = oBC.GetFieldValue("Email Address")
            SendEmail(sEmail)
        End If
    End If
End Function
```
ActiveViewName Method

ActiveViewName returns the name of the active view.

Syntax

\[ \text{Application}.\text{ActiveViewName} \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns

A string containing the active view name

Usage

Do not use the ActiveViewName method in any event handler that may be initiated by the COM Data Server, COM Data Control, or Java Data Bean.

Used With

Browser Script, Mobile Web Client Automation Server, Server Script

Example

The following example is in Siebel eScript:

```plaintext
function BusComp_PreSetFieldValue (FieldName, FieldValue)
{
    switch(FieldName)
    {
        case "Name":
        case "Location":
        case "Account Status":
        case "Alias":
        case "City":
        case "Country":
        case "Currency Code":
```

```plaintext
sEmail = *
End If
iRtn = CancelOperation
End If
WebApplet_PreInvokeMethod = iRtn
End Function
```
case "Current Volume":
case "DUNS Number":
case "Expertise":
case "Freight Terms":
case "Freight Terms Info":
case "Home Page":
case "Industry":
case "Location":
case "Main Phone Number":
case "Main Fax Number":
case "Sales Rep":
var sActiveViewName = TheApplication().ActiveViewName();
if (sActiveViewName == "All Accounts across Organizations")
{
    TheApplication().RaiseErrorText("You cannot update the " + FieldName +
    " on the " + sActiveViewName + " View");
}
break;
}
return (ContinueOperation);

**Attach Method**

The Attach method allows an external application to reconnect to an existing Siebel session.

**Syntax**

`Application.Attach(sessionString)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sessionString</td>
<td>A string containing the Siebel Session Id. The sessionString is typically the output of the Detach method.</td>
</tr>
</tbody>
</table>

**Returns**

Boolean indicating whether or not the method was successfully executed

**Used With**

COM Data Control, Java Data Bean

**Examples**

Each of these examples instantiates the first COM Data Control instance, logs in to a Siebel Server, detaches this instance, and then gains the session string. It then instantiates the second COM Data Control instance. It does not need to log in again, as it attaches to the existing session by using the session string. This reuses the connection created by the first instance.
The following example is for COM Data Control and is written in native Visual Basic:

```vbnet
Dim SiebelApplication_first As SiebelDataControl
Dim SiebelApplication_second As SiebelDataControl
Dim errCode As Integer
Dim sessionString As String
Dim attachResult As Boolean
Dim errText As String

' Instantiate the first instance
Set SiebelApplication_first = CreateObject("SiebelDataControl.SiebelDataControl.1")

' Login to Siebel
SiebelApplication_first.Login "host=""Siebel.tcpip.none.none://<virtual ip>:<port>/<enterprise>/<object manager>"", "<user id>", "<password>"

errCode = SiebelApplication_first.GetLastErrCode
If errCode <> 0 Then
    errText = SiebelApplication_first.GetLastErrText
    MsgBox errText
    Exit Sub
End If

' Detach this instance from Siebel and get session id
sessionString = SiebelApplication_first.Detach
MsgBox "The session string is: " & sessionString

' Instantiate the second instance
Set SiebelApplication_second = CreateObject("SiebelDataControl.SiebelDataControl.1")

' Attach the existing session to this instance
attachResult = SiebelApplication_second.Attach(sessionString)
If (attachResult = True) Then
    MsgBox "Session attached!"
Else
    MsgBox "Session attach failed"
End If

SiebelApplication_second.LogOff
Set SiebelApplication_second = Nothing
Set SiebelApplication_first = Nothing
```

The following example is for Java Data Bean:

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

public class JDBAttachDetachDemo
{
    private SiebelDataBean m_dataBean_first = null;
    private SiebelDataBean m_dataBean_second = null;
```
public static void main(String[] args)
{
    JDBAttachDetachDemo demo = new JDBAttachDetachDemo();
}
public JDBAttachDetachDemo()
{
    try
    {
        // Instantiate the Siebel Data Bean
        m_dataBean_first = new SiebelDataBean();

        // Login to the servers
        m_dataBean_first.login("siegel.tcpip.none.none:<virtual ip>:2320/" +
             "<enterprise>;<object manager name>;<user id>;<password>");
        System.out.println("Logged in to the Siebel server ");

        // Get the Detach Handle
        String detachHandle = m_dataBean_first.detach();
        System.out.println("The session id is: " + detachHandle);

        // Instantiate another Java Data Bean
        SiebelDataBean m_dataBean_second = new SiebelDataBean();

        // Do Attach
        System.out.println("Attaching in to the Siebel server ");
        m_dataBean_second.attach(detachHandle);
        System.out.println("Attach Done ");

        // Logoff
        m_dataBean_second.logoff();
    }
    catch (SiebelException e)
    {
        System.out.println(e.getErrorMessage());
    }
}

**CurrencyCode Method**

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

*Application.CurrencyCode*

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

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, Web Client Automation Server, Server Script

Example

The following example is in Siebel eScript:

```javascript
function WebApplet_Load ()
{
    var currencycode;
    currencycode = TheApplication().CurrencyCode();
    var WshShell = COMCreateObject("WScript.Shell");
    WshShell.Popup(currencycode);
}
```

Detach Method

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

Syntax

*Application.Detach*

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns

String containing the Siebel session Id.

Usage

The string returned by the Detach method should only be used with the Attach method.
**Used With**
COM Data Control, Java Data Bean

**Examples**
For a Java Data Bean sample and a native VB sample using COM Data Control, read “Attach Method” on page 125.

## EnableExceptions Method

The EnableExceptions method enables or disables native COM error handling.

**Syntax**

```
Application.EnableExceptions(bEnable)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnable</td>
<td>A Boolean: TRUE or FALSE</td>
</tr>
</tbody>
</table>

**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 Web Client Automation Server

**Examples**

This native Visual Basic script uses the Siebel ActiveX Data Control to connect to the Siebel Application and instantiate a business object. The script prompts the user to select whether the native error handling is to be enabled or not. If yes, the script throws the error immediately when it gets an error. If not, the script suppresses Siebel errors and errors are only detected by using GetLastErrCode method.

```vbnet
Dim SiebelApplication As SiebelDataControl
Dim errCode As Integer
Dim wrongBO As SiebelBusObject
Dim nativeHandle As String
Set SiebelApplication = CreateObject("SiebelDataControl.SiebelDataControl.1")
' Login to Siebel
```
SiebelApplication_first.Login "host="Siebel.tcpip.none.none://<virtual ip>:<port>/<enterprise>/<object manager>"", "<user id>", "<password>"

nativeHandle = InputBox("Use native error handling?", ", ", "Yes")

If nativeHandle = "Yes" Then
   SiebelApplication.EnableExceptions (True)
Else
   SiebelApplication.EnableExceptions (False)
End If

Set wrongBO = SiebelApplication.GetBusObject("No Such One") 'intended to create an error at this line by instantiating a non-existing Business Object

errCode = SiebelApplication.GetLastErrCode()
If errCode <> 0 Then 'if native error handle is disabled, this block detects it
   ErrText = SiebelApplication.GetLastErrText
   MsgBox ErrText
   Exit Sub
End If

This Visual Basic sample code uses the Siebel Mobile Automation Server to connect to the Siebel Application and instantiate a business object. The program prompts the user to select whether the native error handling is to be enabled or not. If yes, the script throws the error immediately when it gets an error. If not, the script suppresses Siebel errors and errors are only detected by using GetLastErrCode method.

Dim SiebelApp As SiebelWebApplication
Dim errCode As Integer
Dim wrongBO As SiebelBusObject

Set SiebelApp = CreateObject("TWSiebel.SiebelWebApplication.1")

Dim nativeHandle As String
nativeHandle = InputBox("Use native error handle?", ", ", "Yes")

If nativeHandle = "Yes" Then
   SiebelApp.EnableExceptions (True)
Else
   SiebelApp.EnableExceptions (False)
End If

Set wrongBO = SiebelApp.GetBusObject("No Such One") 'intended to create an error at this line by instantiating a non-existing Business Object

errCode = SiebelApp.GetLastErrCode()
If errCode <> 0 Then 'if native error handle is disabled, this block detects it
   ErrText = SiebelApp.GetLastErrText
   MsgBox ErrText
   Exit Sub
End If
**FindApplet Method**

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

**Syntax**

```javascript
theApplication().FindApplet(appletName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>appletName</em></td>
<td>String variable or literal containing the name of the desired applet.</td>
</tr>
</tbody>
</table>

**Returns**

The applet identified in *appletName*

**Usage**

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

**Used With**

Browser Script

**Example**

The following example is in Browser Script:

```javascript
function Applet_ChangeFieldValue(field, value)
{
    if (theApplication().ActiveViewName() == 'Account List View')
    {
        var newapplet = theApplication().FindApplet('Account Entry Applet');
        var entryappletcontrol = newapplet.FindControl('Name');
        var entryappletvalue = entryappletcontrol.GetValue();
        theApplication().SWEAlert(entryappletvalue);
    }
}
```

---

**GetBusObject Method**

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

*Application.GetBusObject(busObject*Name*)

**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. If you use *ActiveBusObj*() you get the business object that exists already (if there is one). If you use *GetBusObject*() instead, any child business components are ALWAYS new ones, even if you have some already.

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

**Examples**
The following examples always instantiate and return a new instance of the business object specified in the argument, which is the Account business object.

The following example is in Siebel eScript:

```javascript
var oBusObject = TheApplication().GetBusObject("Account");
var oBusComp = oBusObject.GetBusComp("Account");

[ Your code here ]
oBusComp = null;
oBusObject = null;
```

The following example is in Siebel VB:

```vbscript
Dim AcntBO as BusObject
Dim AcntBC as BusComp
Dim AddrBC as BusComp
Set AcntBO = TheApplication.GetBusObject("Account")
Set AcntBC = AcntBO.GetBusComp("Account")

[ your code here]
Set AcntBO = Nothing
Set AcntBC = Nothing
```
The following examples instantiate and return a new instance of the business object as did the previous example. However, the difference is that the business object returned could vary depending on the location from which the code is invoked, such as a Web applet event. This is useful when you want to refer to the currently active business object.

The following example is for Java Data Bean:

```java
private SiebelDataBean m_dataBean = null;
private SiebelBusObject m_busObject = null;
m_busObject = m_dataBean.getBusObject("Opportunity");
```

The following example is in Siebel eScript:

```javascript
var oBO = TheApplication().GetBusObject(this.BusObject.Name);
```

The following example is in Siebel VB:

```vbnet
Dim oBO as BusObject
Dim oBC as BusComp
Set oBO = TheApplication.GetBusObject(Me.BusObject.Name)
```

## GetLastErrCode Method

The GetLastErrCode method returns the last error execution status.

**Syntax**

```
Application.GetLastErrCode
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**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 Web Client Automation Server, Web Client Automation Server

**Example**

The following example is for COM Data Control. `Siebel Application` is an Application instance.
errcode = SiebelApplication.GetLastErrCode
If errcode <> 0 Then
    ErrText = SiebelApplication.GetLastErrText
    MsgBox ErrText
    Exit Sub
End If

Related Topic
“GetLastErrText Method” on page 134

GetLastErrText Method
The GetLastErrText method returns the last error text message.

Syntax
Application.GetLastErrText

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns
The last error text message as a string

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

Example
The following example is for COM Data Control. SiebelApplication is an Application instance.

errcode = SiebelApplication.GetLastErrCode
If errcode <> 0 Then
    ErrText = SiebelApplication.GetLastErrText
    MsgBox ErrText
    Exit Sub
End If

Related Topic
“GetLastErrCode Method” on page 133
GetProfileAttr Method

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

Syntax

Application.GetProfileAttr(name)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>A string indicating the name of the attribute</td>
</tr>
</tbody>
</table>

Returns

The value of the attribute name

Usage

GetProfileAttr is used in personalization to retrieve values of attributes in a user profile. It cannot be used with system fields, except Id, because they are not explicitly defined in the Personalization Profile business component. For more information on profile attributes, see Siebel Personalization Administration Guide.

Used With

Browser Script, COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

Examples

The following example is in Browser Script:

```javascript
var myprofile = theApplication().GetProfileAttr("Hobby");
```

The following example is in Siebel eScript:

```javascript
var myprofile = TheApplication().GetProfileAttr("Hobby");
```

The following example is in Siebel VB:

```vbnet
Dim myprofile As String
myprofile = TheApplication.GetProfileAttr("Hobby")
```

Related Topic

“SetProfileAttr Method” on page 162

GetService Method

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

Application.GetService(serviceName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>The name of the service to start</td>
</tr>
</tbody>
</table>

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 business 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, is cleared by setting it 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 Siebel Tools as an application user property. This prevents Service Not Found errors.

To register a business service

1. In Siebel Tools, select the Application object in the Object Explorer.
2. Select the desired application, for example Siebel Universal Agent, in the Object List Editor.
3. In the Object Explorer, expand the Application object, and then choose Application User Prop.
4. In the Object List Editor, create new application user property records as needed:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClientBusinessService0</td>
<td>XML Converter</td>
</tr>
<tr>
<td>ClientBusinessService1</td>
<td>My Business Service</td>
</tr>
</tbody>
</table>

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

Used With


Examples

The following examples instantiate a business service named Workflow Process Manager.
The following example is in Browser Script:

```javascript
function Applet_PreInvokeMethod (name, inputPropSet)
{
    if (name == 'MyCustomMethod')
    {
        var oBS;
        var inpPS;
        var outPS;
        inpPS = theApplication().NewPropertySet();
        outPS = theApplication().NewPropertySet();
        oBS = theApplication().GetService("Workflow Process Manager");
        outPS = oBS.InvokeMethod("RunProcess", inpPS);
        inpPS = null;
        outPS = null;
        return ('CancelOperation');
    }
    else
    {
        return ('ContinueOperation');
    }
}
```

The following example is in Siebel eScript:

```javascript
function WebApplet_PreInvokeMethod (MethodName)
{
    if (MethodName == "MyCustomMethod")
    {
        var oBS;
        var inpPS;
        var outPS;
        inpPS = TheApplication().NewPropertySet();
        outPS = TheApplication().NewPropertySet();
        oBS = TheApplication().GetService("Workflow Process Manager");
        oBS.InvokeMethod("RunProcess", inpPS, outPS);
        inpPS = null;
        outPS = null;
        oBS = null;
        return (CancelOperation);
    }
    else
    {
        return (ContinueOperation);
    }
}
```

The following example is in Siebel VB:

```vbnet
Function WebApplet_PreInvokeMethod (MethodName As String) As Integer
If MethodName = "MyCustomMethod" Then
    Dim oBS As Service
    Dim inpPS As PropertySet
    Dim outPS As PropertySet
    Set inpPS = TheApplication.NewPropertySet
```

---

**Siebel Object Interfaces Reference** Version 8.0, Rev. B 137
Set outPS = TheApplication.NewPropertySet
Set oBS = TheApplication.GetService("Workflow Process Manager")
oBS.InvokeMethod "RunProcess", inpPS, outPS
Set inpPS = Nothing
Set outPS = Nothing
Set oBS = Nothing
WebApplet_PreInvokeMethod = CancelOperation
Else
WebApplet_PreInvokeMethod = ContinueOperation
End If
End Function

GetSharedGlobal Method

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)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>varName</td>
<td>String literal or variable containing the name of the global variable</td>
</tr>
</tbody>
</table>

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, Mobile Web Client Automation Server, Server Script

Examples

In the following examples, the GetSharedGlobal method is called to get a global variable called myGlobalVar. The global variable was originally set using the SetSharedGlobal method in the Application_Start event. The global variable can be accessed from any event. For these examples, in the BusComp_WriteRecord event, the GetSharedGlobal method is called to retrieve myGlobalVar.

The following example is for COM. SiebelApplication is an Application instance.
Dim sReturn As String
oleVar = SiebelApplication.GetSharedGlobal("myGlobalVar", errCode)
SiebelApplication.SetSharedGlobal("myGlobalVar", "helloworld", errCode)

The following example is in Siebel eScript:

```javascript
function Application_Start(CommandLine)
{
TheApplication().SetSharedGlobal("myGlobalVar", "helloworld");
}

function BusComp_WriteRecord()
{
var myVar;
myVar = TheApplication().GetSharedGlobal("myGlobalVar");
}
```

The following example is in Siebel VB:

```vb
Sub Application_Start(CommandLine As String)
    TheApplication.SetSharedGlobal "myGlobalVar", "helloworld"
End Sub

Sub BusComp_WriteRecord
    Dim myVar As String
    myVar = TheApplication.GetSharedGlobal("myGlobalVar")
End Sub
```

**Related Topic**

“SetSharedGlobal Method” on page 164

---

**GotoView Method**

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])`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViewName</td>
<td>The name of the view for the Siebel application to display</td>
</tr>
<tr>
<td>BusinessObjectName</td>
<td>An optional argument 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.</td>
</tr>
</tbody>
</table>
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 following events: Application_Navigate, Application_PreNavigate, Application_Start, Navigate, PreNavigate, and WebApplet_Load.

The following Siebel VB script uses GotoView to programmatically navigate to the Opportunity List view.

TheApplication.GotoView "Opportunity List View", Nothing

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

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.

The Control property “Show Popup” should not be set to TRUE on a button if there is underlying script that uses GotoView. If Show Popup is set to TRUE and GotoView is used, the view is opened in a new browser window. The Siebel client UI does not support a Multiple Document Interface (MDI) architecture, so this combined configuration and scripted call to GotoView is not supported.

Used With
Server Script

Example
The following examples show how to use GoToView with and without the optional business object parameter.

The following example is in Siebel eScript:

function BusComp_WriteRecord ()
{
  var leadQuality;
  var actName;
  var actBO;
  var actBC;

  // Get the lead quality for this opportunity
  leadQuality = this.GetFieldValue("Quality");
  if(leadQuality == "1-Excellent")
  {

}
// If it is an excellent lead,
// go to the account for this opportunity
actName = this.GetFieldValue("Account");
actBO = TheApplication().GetBusObject("Account");
actBC = actBO.GetBusComp("Account");

with (actBC)
{
    SetUpViewMode(AllView);
    ClearToQuery();
    SetSearchSpec("Name", actName);
    ExecuteQuery(ForwardBackward);
}

TheApplication().GotoView("All Account List View", actBO);

else
{
    TheApplication().GotoView("Opportunity Detail - Activities View");
}

actBC = null;
actBO = null;

The following example is in Siebel VB:

Sub BusComp_WriteRecord
    Dim leadQuality As String
    Dim actName As String
    Dim actBO As BusObject
    Dim actBC As BusComp

    ' Get the lead quality for this opportunity
    leadQuality = Me.GetFieldValue("Quality")
    If (leadQuality = "1-Excellent") Then

        ' If it is an excellent lead
        ' go to the account for this opportunity
        actName = Me.GetFieldValue("Account")
        Set actBO = TheApplication.GetBusObject("Account")
        Set actBC = actBO.GetBusComp("Account")

        With actBC
            SetUpViewMode(AllView)
            ClearToQuery
            SetSearchSpec "Name", actName
            ExecuteQuery
        End With

        TheApplication().GotoView "All Account List View", actBO
    End If

End Sub
Else
    TheApplication.GotoView "Opportunity Detail - Activities View"
End If

Set actBC = Nothing
Set actBO = Nothing

End Sub

InvokeMethod Method
InvokeMethod calls a specialized method or user-defined method specified by its argument.

**Browser Script Syntax**
theApplication().InvokeMethod(methodName, methodArgs);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>methodName</td>
<td>The name of the method.</td>
</tr>
<tr>
<td>methodArgs</td>
<td>One or more strings containing arguments to methodName.</td>
</tr>
</tbody>
</table>

**Server Script Syntax**
Application.InvokeMethod(methodName, methodArgs);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>methodName</td>
<td>The name of the method.</td>
</tr>
<tr>
<td>methodArg1, methodArg2, ..., methodArgN</td>
<td>One or more strings containing arguments to methodName.</td>
</tr>
</tbody>
</table>

**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. Oracle 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, Java Data Bean, Mobile Web Client Automation Server, Server Script

Example
For an example, read “InvokeMethod Method” on page 100.

InvokeMethod Methods for the Application Object
The following methods are supported for use with InvokeMethod:

- “GetDataSource Method” on page 143
- “IsViewReadOnly Method” on page 144
- “Language Method” on page 145
- “LookupValue Method” on page 145

GetDataSource Method
Returns the name of the data source, as defined in the DataSource server parameter, that is being used for the session. (The default is ServerDataSrc.)

Syntax
dataSrc = Application.InvokeMethod("GetDataSource")

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

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

Used With
This method is supported by Application.InvokeMethod() calls in COM Data Control, Java Data Bean, Mobile Web Client Automation Server, and Server Script.

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

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

The following is the same example in Siebel VB.
Dim dataSrc As String
dataSrc = TheApplication.InvokeMethod("GetDataSource")
TheApplication.RaiseErrorText(dataSrc)

**IsViewReadOnly Method**
You can use the IsViewReadOnly method to test whether a view is read-only.

**Syntax**
Application.InvokeMethod("IsViewReadOnly",viewName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>viewName</td>
<td>The name of a view, as defined in Siebel Tools, in double quotes or a variable containing the name of a view.</td>
</tr>
</tbody>
</table>

**Returns**
Returns TRUE if the view is read-only, else it returns FALSE. If neither of these values is returned, then an error has occurred. Your script should provide a handler if neither TRUE nor FALSE is returned.

**Usage**
You can set a view as read-only for particular responsibilities in the Responsibility Administration view. One use of the IsViewReadOnly method is to determine whether such a view is read-only for the current responsibility before attempting to edit a field.

Buttons are not automatically set to read-only on views that are read-only due to the view-responsibility association, so another application is to set buttons to read-only in views for which IsViewReadOnly returns TRUE.

**Used With**
This method is supported by Application.InvokeMethod() calls in Browser Script, COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, and Server Script.

**Example**
The following example for Siebel eScript determines whether the active view is read only:

```javascript
function ShowViewROStatus()
{
    var sActive = TheApplication().ActiveViewName();
    if (TheApplication().InvokeMethod("IsViewReadOnly",sActive) == "TRUE")
        TheApplication().RaiseErrorText(sActive + " is read only.");
}
```
else
    TheApplication().RaiseErrorText(sActive + " is not read only.");
}

**Language Method**
Retrieves the language code of the language in which the active application is running.

**Syntax**
Application.InvokeMethod(“Language”);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

**Returns**
The language code of the active application, for example ENU

**Used With**
This method is supported by Application.InvokeMethod() calls in Server Script.

**Example**
Siebel VB:
```vbnet
dim curLang as string
    curLang = TheApplication.InvokeMethod("Language")
```

Siebel eScript:
```javascript
var curLang;
    curLang = TheApplication().InvokeMethod("Language");
```

**LookupValue Method**
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 = Application.InvokeMethod("LookupValue", type, lang_ind_cd)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Type as specified in the List of Values administration view.</td>
</tr>
<tr>
<td>lang_ind_cd</td>
<td>Language independent code value as specified in the List of Values administration view.</td>
</tr>
</tbody>
</table>

Returns
Returns a string containing the display value (the VAL column) for the row. LookupValue tries to find the display value for a given language independent code. If the display value is not found, LookupValue returns the language independent code itself as the value.

Used With
This method is supported by Application.InvokeMethod() calls in COM Data Control, Java Data Bean, Mobile Web Client Automation Server, and Server Script.

Example
The following eScript example 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.

```javascript
var LOVText = TheApplication().InvokeMethod("LookupValue", "SR_AREA", "Network");
```

LoadObjects Method

The LoadObjects method is used to start the COM Data Server object. This method must be the first call to the COM Data Server.
Syntax
Application.LoadObjects(absoluteCFGfileName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>absoluteCFGfileName</td>
<td>The complete path and name of the CFG file to open. For example: “C:\siebel\bin\uagent.cfg”</td>
</tr>
<tr>
<td></td>
<td>You can optionally identify the data source in the argument to the LoadObjects method by appending to the CFG file string, separated by a comma. For example: “D:\Server\siebsrvr\bin\ENU\siebel.cfg,ServerDataSrc”</td>
</tr>
<tr>
<td></td>
<td>When the data source is not specified, the LoadObjects method assumes “Local” as the data source.</td>
</tr>
</tbody>
</table>

Returns
Nothing if successful, else throws an error

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

When using COM Data Server, the COM client cannot create multiple connections to the COM Server. For example, a second attempt at calling LoadObjects() causes the error message: “The object definition manager has already been initialized.” The COM client must be restarted before another connection attempt can be successful. Use COM Data Control instead.

Used With
COM Data Server

Example
The following example is for COM Data Server. SiebelApplication is an Application instance.

```vbnet
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.GetLastErrorText
End Sub
```
LoadUserAttributes Method

The LoadUserAttributes method loads a user profile into the session.

Syntax

LoadUserAttributes(row_id)

Argument | Description
--- | ---
row_id | The row ID of the user whose profile needs to be loaded

Returns

Not applicable

Usage

This function has only one argument: the row ID of the user whose profile needs to be loaded. This loaded profile can be accessed as the “You” profile from personalization rules, with one exception: if the row ID is that of the current user, the profile will be loaded into the “Me” profile.

If this function is called with no argument, it unloads the loaded user profile.

For more information on user profiles, see Siebel Personalization Administration Guide.

Used With

Server Script

Example

The following Siebel 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
```

The following Siebel VB example unloads the loaded user profile:

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

Login Method

The Login method allows external applications to log in to the COM Data Server, COM Data Control, or Java Data Bean, and to 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

\texttt{Application>Login([connectString,] \textit{userName}, \textit{password})}

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{connectString}</td>
<td>Token-based connect string</td>
</tr>
<tr>
<td>\textit{userName}</td>
<td>Username for login</td>
</tr>
<tr>
<td>\textit{password}</td>
<td>User password for login</td>
</tr>
</tbody>
</table>

Returns

A string containing the error code

Usage

Verify that the \texttt{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 \texttt{EnableOLEAutomation} to \texttt{TRUE} in your \texttt{CFG} file (this is the default value for the argument).

For information on formatting the connect string, read “Connect String” on page 74.

Used With

COM Data Control, COM Data Server, Mobile Web Client Automation Server, Java Data Bean

Example

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

\begin{verbatim}
  host = "Siebel://my_computer/SIEBEL/objsrvr/my_computer" lang = "ENU"
\end{verbatim}

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

To use the COM Data Control in Visual Basic:

\begin{verbatim}
  m_dataBean.login("siebel.tcpip.none.none://gateway:gatewayport/enterpriseserver/SCCObjMgr", "username", "password");
\end{verbatim}

To use the COM Data Control in C++:

\begin{verbatim}
  Login(\"host=\"siebel://my_computer/SIEBEL/objsrvr/my_computer\" lang = \"ENU\"\", \"user\", \"password\");
\end{verbatim}

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

\begin{verbatim}
  Call SiebelAppControl>Login("host="siebel://gtwy/enterprise/ObjMgr**, "SADM1N", "SADM1N")

  //Check for errors
  If SiebelAppControl.GetLastErrorCode <> 0 Then
    frmMain.txtStatus.Text = SiebelAppControl.GetLastErrText
\end{verbatim}
Else
    frmMain.txtStatus.Text = "Connected successfully..."
End If

The following is a Java Data Bean example that logs into a Siebel Server and then logs off:

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

public class JDBLoginLogoffDemo
{
    private SiebelDataBean m_dataBean = null;
    public static void main(String[] args)
    {
        JDBLoginLogoffDemo demo = new JDBLoginLogoffDemo();
    }

    public JDBLoginLogoffDemo()
    {
        try
        {

            // instantiate the Siebel Data Bean
            m_dataBean = new SiebelDataBean();

            // login to the servers
            m_dataBean.login("siebel.tcpip.none.none://<gateway>:<port>/<enterprise>/<object manager>","<userid>","<password>");
            System.out.println("Logged in to the Siebel server ");

            //perform function code

            //release the business object

            // logoff
            m_dataBean.logoff();
            System.out.println("Logged off the Siebel server ");
        }
        catch (SiebelException e)
        {
            System.out.println(e.getMessage());
        }
    }
}
```

**LoginId Method**

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

Application>LoginId

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns

A string containing the login ID

Usage

The login ID is the value of the ROW_ID column in the user’s login record in the S_USER table. When obtained, the login ID can be conveniently used as a search specification.

Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client, Automation Server, Server Script

Example

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

```vbnet
Function BusComp_PreSetFieldValue (FieldName As String, FieldValue As String) As Integer
    Select Case FieldName
        Case "Account Status"
            If Me.GetFieldValue("Created By") <> Application.LoginId Then
                Application.RaiseErrorText("You cannot change Account Status because you did not create the record.")
            End If
    End Select
    BusComp_PreSetFieldValue = ContinueOperation
End Function
```

LoginName Method

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*

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

A string containing the user’s login name

**Used With**

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

**Example**

For examples, read "ExecuteQuery Method" on page 193 and "TheApplication Method" on page 322.

**Related Topic**

“Login Method” on page 148

---

**Logoff Method**

The Logoff method disconnects the client from the server.

**Syntax**

*Application.Logoff*

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**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 called automatically if you destroy the main object.
LookUpMessage Method

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

Syntax

Application.LookUpMessage(category, key, [arg1], [arg2],....., [argN])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>category</td>
<td>Name of the Message Category object, as defined in Siebel Tools, that is the parent of Key value.</td>
</tr>
<tr>
<td>key</td>
<td>Name of the Message object, as defined in Siebel Tools, whose text contains the value to be investigated.</td>
</tr>
<tr>
<td>arg1, arg2, ..., argN</td>
<td>Optional arguments used to format the error message if it contains any substitution arguments (%1, %2).</td>
</tr>
</tbody>
</table>

Returns

A string containing the localized message text.

Usage

Useful for retrieving locale specific custom error messages.

Used With

Server Script

Example

The following eScript example returns the text “Account Title should be entered before Stepping off.” To test this under the “User Defined Errors” message category, create a new record with the following text: “%1 should be entered before Stepping Off.” The parameter that is substituted in place of %1 is “Account Title”, which is present in the message test.

```javascript
var sVal = TheApplication().LookUpMessage("User Defined Errors", "Test", "Account Title");
```
Name Method
The Name method returns name of the application.

Syntax
Application.Name

Returns
A string containing the name of the application

Used With
Browser Script, Web Client Automation Server

NewPropertySet Method
The NewPropertySet method constructs a new property set object.

Syntax
Application.NewPropertySet

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
Example

This method constructs a new property set object.

The following example is in Browser Script:

```javascript
function Applet_PreInvokeMethod (name, inputPropSet)
{
    if (name == "MyCustomMethod") {
        var oBS;  
        var inpPS;  
        var outPS;  
        inpPS = theApplication().NewPropertySet();  
        outPS = theApplication().NewPropertySet();  
        oBS = theApplication().GetService("New Value Business Service");  
        outPS = oBS.InvokeMethod("New Value Method", inpPS);  
        inpPS = null;  
        outPS = null;  
        oBS = null;  
        return ("CancelOperation");
    } else {
        return ("ContinueOperation");
    }
}
```

The following example is for COM. SiebelApplication is an Application instance.

```javascript
Dim oBS As SiebelService
Dim inpPS As SiebelPropertySet
Dim outPS As SiebelPropertySet
Dim errCode as integer
Set inpPS = SiebelApplication.NewPropertySet errCode
Set outPS = SiebelApplication.NewPropertySet errCode
Set oBS = SiebelApplication.GetService("New Value Business Service", errCode)
oBS.InvokeMethod "New Value Method", inpPS, outPS, errCode
Set inpPS = Nothing
Set outPS = Nothing
Set oBS = Nothing
```

The following example is in Siebel eScript:

```javascript
function WebApplet_PreInvokeMethod (MethodName)
{
    if (MethodName == "MyCustomMethod") {
        var oBS;  
        var inpPS;  
        var outPS;  
        inpPS = TheApplication().NewPropertySet();  
        outPS = TheApplication().NewPropertySet();  
        oBS = TheApplication().GetService("New Value Business Service");
    }
```
The following example is in Siebel VB:

```vbnet
Function WebApplet_PreInvokeMethod (MethodName As String) As Integer
    If MethodName = "MyCustomMethod" Then
        Dim oBS As Service
        Dim inpPS As PropertySet
        Dim outPS As PropertySet
        Set inpPS = TheApplication.NewPropertySet
        Set outPS = TheApplication.NewPropertySet
        Set oBS = TheApplication.GetService("New Value Business Service")
        oBS.InvokeMethod("New Value Method", inpPS, outPS)
        Set inpPS = Nothing
        Set outPS = Nothing
        Set oBS = Nothing
        WebApplet_PreInvokeMethod = CancelOperation
    Else
        WebApplet_PreInvokeMethod = ContinueOperation
    End If
End Function
```

**PositionId Method**

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`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

A string row ID
**PositionName Method**

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
```

**Returns**

A string containing the user’s position

**Used With**

COM Data Control, COM Data Server, Java Data Bean, Mobile 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.

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

Dim sPosName As String sMsgText As String
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 ((sPosName="Manager") OR (sPosName="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.")
        End If
    End If

BusComp_PreSetFieldValue = ContinueOperation
End Select

End Function
```
# RaiseError Method

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 arguments are used to format the string if it contains any substitution arguments (%1, %2).

## Syntax

```javascript
Application.RaiseError(key, [arg1], [arg2],..., [argN])
```

## Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>key</code></td>
<td>Name of the Message object, as defined in Siebel Tools, whose text contains the value to be used.</td>
</tr>
<tr>
<td><code>arg1, arg2, ..., argN</code></td>
<td>Optional arguments used to format the error message if it contains any substitution arguments (%1, %2).</td>
</tr>
</tbody>
</table>

## Returns

Not applicable

## Usage

When invoked, the RaiseError method causes execution of the script to terminate, and sends a notification to the browser. Therefore, CancelOperation is not required after RaiseError.

Internally, the RaiseError/RaiseErrorText methods raise a Server Script exception. If you have implemented error handling in your scripts, the error handling can suppress RaiseError and RaiseErrorText functionality.

If you have implemented error handling in Siebel VB, when 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.

**CAUTION:** Be careful when using RaiseError, because it cancels operations. For example, if it is used in BusComp_PreWriteRecord, the user or code will not be able to step off the current record until the condition causing the RaiseError method to be invoked is addressed.

## Used With

Server Script

## Example

In the following eScript example, the RaiseError results in a scripting exception 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.GetFieldValuel"Account Status";

        if (status == "Gold") {
            TheApplication().RaiseError(<user defined error name>);
        }
        else {
            return (ContinueOperation);
        }
    }
    catch (e) {
        throw e;
    }
}

The following eScript example raises the error message "This user-defined test error is used in PreDelete, as an example for RaiseError Method" when deleting an opportunity with the “Pipeline” revenue class. Note that the key "user-defined test error1" is pre-defined as "This user-defined test error is used in %1, as an example for %2". When the script runs, 'PreDelete' is substituted for %1 and 'Raise Error Method' is substituted for %2.

function BusComp_PreDeleteRecord ()
{
    try {
        var revClass = this.GetFieldValue("Primary Revenue Class");
        if (revClass == "1-Pipeline")
        {
            TheApplication().RaiseError("user-defined test error1", "PreDelete", "RaiseError Method");
        }
        else {
            return (ContinueOperation);
        }
    }
    catch (e) {
        throw e;
    }
}

RaiseErrorText Method

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

```
Application.RaiseErrorText(value, [arg1], [arg2],..., [argN])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>value</code></td>
<td>The error text message.</td>
</tr>
<tr>
<td><code>arg1, arg2, ..., argN</code></td>
<td>Optional arguments used to format the error message if it contains any substitution arguments (%1, %2).</td>
</tr>
</tbody>
</table>

Returns

Not applicable

Usage

When invoked, the RaiseErrorText method stops execution of the script. Therefore, CancelOperation is not required after RaiseErrorText.

Internally, the RaiseError and RaiseErrorText methods raise a Server Script exception. Therefore, if you have implemented error handling in your scripts, the error handling can suppress RaiseError and RaiseErrorText functionality.

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.

**NOTE:** Do not use the %s and %n formatting literals with the RaiseErrorText method. This causes unpredictable results.

**CAUTION:** Be careful when using RaiseErrorText, because it cancels operations. For example, if it is used in BusComp_PreWriteRecord, the user or code will not be able to step off the current record until the condition causing the RaiseErrorText method to be invoked is addressed.

Used With

Server Script

Example

In the following eScript example, the RaiseErrorText results in a scripting exception being raised, transferring control to the catch statement. For the error message to be displayed, the error must be thrown, using the throw statement.

```javascript
function BusComp_PreDeleteRecord ()
{
  try {
    var status = this.getFieldValue("Account Status");
    if (status == "Gold") {
      TheApplication().RaiseErrorText("Unable to delete Gold Account");
    }
    else {
```
The following eScript example raises an error when deleting an opportunity with the “Pipeline” revenue class.

```javascript
function BusComp_PreDeleteRecord ()
{
    try
    {
        var revClass = this.GetFieldValue("Primary Revenue Class");
        if (revClass == "1-Pipeline")
        {
            TheApplication().RaiseErrorText("Exception occurred in %1. Unable to delete Opportunity with %2 revenue class.", "PreDeleteRecord", revClass);
        }
        else
        {
            return (ContinueOperation);
        }
    }
    catch (e)
    {
        throw e;
    }
}
```

**SetPositionId Method**

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

**Syntax**

```javascript
Application.SetPositionId(positionId)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>positionId</td>
<td>A string containing the Position Id you would like to change to</td>
</tr>
</tbody>
</table>

**Returns**

A Boolean denoting whether or not the operation was successfully completed.
**Usage**
When invoking the `SetPositionId` method, the `positionId` argument must contain a Position Id that has already been associated with the current, logged-in user.

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

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

**Syntax**

```application
Application.SetPositionName(positionName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>positionName</code></td>
<td>A string containing the name of the position.</td>
</tr>
</tbody>
</table>

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

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

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

**SetProfileAttr Method**
`SetProfileAttr` is used in personalization to assign values to attributes in a user profile.
Syntax

```
Application.SetProfileAttr(name, value)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>A string indicating the name of the attribute</td>
</tr>
<tr>
<td>value</td>
<td>The value of name</td>
</tr>
</tbody>
</table>

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 argument string already exists, the corresponding persistent profile attribute in the application, defined in the Personalization Profile business component, is updated with the new value. If the profile attribute specified in the argument 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.

In Browser Script, using `SetProfileAttr()` triggers a round trip to the server and back, creating a performance overhead each time it is used.

**NOTE:** `SetProfileAttr()` cannot be used with system fields, which are not explicitly defined in the Personalization Profile business component. While `GetProfileAttr()` can be used with the Id field, `SetProfileAttr()` cannot be used with it, because attempting to change the ROW_ID column of a table will generate an error. For more information on system fields, see *Configuring Siebel Business Applications*.

Used With

Browser Script, COM Data Control, COM Data Server, Server Script, Java Data Bean, Mobile Web Client Automation Server

Example

The following example is in Browser Script:

```javascript
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";
    }
}
```
This following examples show how to exchange information between applet server scripts and applet browser scripts. In an applet server script, a customer profile attribute called MyProAttr is set to "Hello World" using the SetProfileAttr method. In applet browser scripts, you can retrieve the profile attribute using GetProfileAttr method.

The following example is in Siebel eScript:

```javascript
function WebApplet_PreInvokeMethod (MethodName)
{
    if (MethodName == "MyCustomMethod") {
        TheApplication().SetProfileAttr("MyProAttr", "Hello World eScript");
        return (CancelOperation);
    }
    return (ContinueOperation);
}
```

The following example is in Siebel VB:

```vbnet
Function WebApplet_PreInvokeMethod (MethodName As String) As Integer
If MethodName = "MyCustomMethod" Then
    TheApplication.SetProfileAttr "MyProAttr", "Hello World VB"
    WebApplet_PreInvokeMethod = CancelOperation
Else
    WebApplet_PreInvokeMethod = ContinueOperation
End If
End Function
```

Related Topics

"Name Method" on page 154

For more information on user profile attributes, read *Siebel Applications Administration Guide*.

### SetSharedGlobal Method

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 can be accessed using GetSharedGlobal.
Syntax

```
Application.SetSharedGlobal(varName, value)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>varName</td>
<td>String variable or literal containing the name of the shared global variable to set</td>
</tr>
<tr>
<td>value</td>
<td>String variable or literal containing the value to set the variable to set</td>
</tr>
</tbody>
</table>

Returns

Not applicable

Used With

COM Data Control, COM Data Server, Mobile Web Client Automation Server, Server Script

Example

The following example is for COM. SiebelApplication is an Application instance.

```java
comVar = SiebelApplication.GetSharedGlobal("myVar", errCode)
SiebelApplication.SetSharedGlobal("myVar", "BLAH", errCode)
```

The following example is in Siebel VB:

```vbnet
TheApplication.SetSharedGlobal("myVar", "FOO"
myVar = TheApplication.GetSharedGlobal("myVar")
```

In this example, the SetSharedGlobal method is called to set a global variable called myGlobalVar in Application_Start event. The global variable can be accessed from any event. For this example, in the BusComp_WriteRecord event, the GetSharedGlobal method is called to retrieve the global variable.

The following example is for COM. SiebelApplication is an Application instance.

```java
Dim sReturn as String
oleVar = SiebelApplication.GetSharedGlobal("myGlobalVar", errCode)
SiebelApplication.SetSharedGlobal("myGlobalVar", "helloworld", errCode)
```

The following example is in Siebel eScript:

```javascript
function Application_Start (CommandLine)
{
    TheApplication().SetSharedGlobal("myGlobalVar", "helloworld");
}

function BusComp_WriteRecord ()
{
    var myVar;
    myVar = TheApplication().GetSharedGlobal("myGlobalVar");
}
```

The following example is in Siebel VB:
Sub Application_Start (CommandLine As String)
    TheApplication.SetSharedGlobal "myGlobalVar", "helloworld"
End Sub

Sub BusComp_WriteRecord
    Dim myVar as String
    myVar = TheApplication.GetSharedGlobal("myGlobalVar")
End Sub

Related Topic
"GetLastErrCode Method" on page 133

ShowModalDialog Method
ShowModalDialog allows you to show a modal dialog box with the cursor maintained in its default state. This Application object method invokes Microsoft’s equivalent Window object method.

Syntax
theApplication().ShowModalDialog (url[, argin][, options])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>The URL of the document to load and display.</td>
</tr>
<tr>
<td>argin</td>
<td>This parameter is used to pass arguments to use when displaying the document. This argument can be a value of any type, including an array of values. For more information, refer to the window DOM object's window.dialogArguments property. See, for example:</td>
</tr>
</tbody>
</table>
**Returns**
The value of the returnValue property, as set by the window of the document specified by the *url* parameter.

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options</td>
<td>String that specifies the attributes of the window that displays the dialog box. This parameter may include one or more of the following semicolon-delimited values:</td>
</tr>
<tr>
<td>■ dialogHeight:<em>sHeight</em> sets the height of the dialog window, where <em>sHeight</em> can be an integer or floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). For consistent results, specify the dialogHeight and dialogWidth in pixels when designing modal dialog boxes. Default unit of measure is em. Minimum height is 100 pixels.</td>
<td></td>
</tr>
<tr>
<td>■ dialogLeft:<em>sXPos</em> sets the left position of the dialog window relative to the upper-left corner of the desktop.</td>
<td></td>
</tr>
<tr>
<td>■ dialogTop:<em>sYPos</em> sets the top position of the dialog window relative to the upper-left corner of the desktop.</td>
<td></td>
</tr>
<tr>
<td>■ dialogWidth:<em>sWidth</em> sets the width of the dialog window.</td>
<td></td>
</tr>
<tr>
<td>■ center:{ yes</td>
<td>no</td>
</tr>
<tr>
<td>■ dialogHide:{ yes</td>
<td>no</td>
</tr>
<tr>
<td>■ edge:{ sunken</td>
<td>raised } specifies the edge style of the dialog window. The default is raised.</td>
</tr>
<tr>
<td>■ help:{ yes</td>
<td>no</td>
</tr>
<tr>
<td>■ resizable:{ yes</td>
<td>no</td>
</tr>
<tr>
<td>■ scroll:{ yes</td>
<td>no</td>
</tr>
<tr>
<td>■ status:{ yes</td>
<td>no</td>
</tr>
<tr>
<td>■ unadorned:{ yes</td>
<td>no</td>
</tr>
</tbody>
</table>
Example
This example shows how this method can be used in browser script to bring up a modal dialog box with a specified URL.

```javascript
function Applet_Load () {
  var sOptions = "dialogHeight: 1000px; edge: sunken; resizable; yes";
  theApplication().ShowModalDialog("http://www.yahoo.com", ",", sOptions)
}
```

SWEAlert Method
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(), pop-up 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 background.

Example
The following browser script example displays a status message to the user.

```javascript
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
```
**Trace Method**

The Trace method appends a message to the trace file. Trace is useful for debugging SQL query execution and the allocation of the objects. This tracing is not the same as the tracing that can be activated in the application’s CFG file. For more information, read “Tracing Scripts” on page 22.

**NOTE:** This method and the TraceOn Method on page 172 are meant for debugging purposes and are not recommended for use in production environments.

**Syntax**

```plaintext
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, Java Data Bean, Mobile Web Client Automation Server, Server Script

**Example**

The following example is for COM Data Server. `SiebelApplication` is an Application instance.

```plaintext
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
```

The following example is in 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

The following is 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

The following is sample output of an 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.CREATION_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 = :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

Related Topics
  "TraceOff Method"
  "TraceOn Method" on page 172
TraceOff Method

TraceOff turns off the tracing started by the TraceOn method.

**Syntax**

```vbscript
Application.TraceOff
```

**Returns**

Not applicable

**Used With**

COM Data Control, COM Data Server, Java Data Bean, Mobile 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:

```vbscript
Sub BusComp_NewRecord
    TheApplication.TraceOn "C:\lvpick.doc", 'SQL', ""
    Dim oBC as BusComp
    set oBC = me.GetPickListBusComp("Sales Stage")
    With oBC
        .SetViewMode AllView
        .ActivateField "Sales Stage Order"
        .ClearToQuery
        .SetSortSpec "Sales Stage Order"
        .ExecuteQuery ForwardOnly
        if .FirstRecord then
            .Pick
            end if
    End With
    set oBC = Nothing
    TheApplication.TraceOff
End Sub
```
**TraceOn Method**

TraceOn turns on the tracking of allocations and deallocations of Siebel objects and SQL statements generated by the Siebel application.

**Syntax**

\[ \text{Application.} \text{TraceOn}(\text{filename, type, selection}) \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
</table>
| filename | Output filename for the trace messages. If this argument is not specified, tracing information is logged to the Object Manager log file for that user session. The filename argument can take two additional inline arguments: $p and $t. The $p argument substitutes the process id to the filename, and $t substitutes the thread id to the file name. For example: 
\[
\text{TheApplication().TraceOn("d:\temp\trace_$p$\_t.txt", "Allocation", "All")};
\]

would log trace files to d:\temp\trace\trace_1496_1412.txt. Place a separator between the $p and $t arguments to make sure that the filename argument 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 
\[
d:\temp\trace\trace_112.txt
\]

If user B had a process id of 11, and a thread id of 2, the tracing file would be 
\[
d:\temp\trace\trace_112.txt
\]

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. 
\[
d:\temp\trace\trace_1_12.txt
d:\temp\trace\trace_11_2.txt
\]

<table>
<thead>
<tr>
<th>type</th>
<th>Specifies the type of tracing to start. This can have the following values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Allocation. Traces allocations and deallocations of Siebel objects. This option is useful if you suspect memory leaks in your code.</td>
<td></td>
</tr>
<tr>
<td>• SQL. Traces SQL statements generated by the Siebel application.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>selection</th>
<th>Indicates which Siebel objects should be traced for the Allocation trace type. This argument should be &quot;&quot; if the trace type is SQL.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Script. Traces VB and eScript objects.</td>
<td></td>
</tr>
<tr>
<td>• OLE. Traces allocations for data server or automation server programs.</td>
<td></td>
</tr>
<tr>
<td>• 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.</td>
<td></td>
</tr>
</tbody>
</table>
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. The old file is left open (locked). You can issue multiple TraceOn statements to the same trace file.

NOTE: This method and the Trace Method on page 169 are meant for debugging purposes and are not recommended for use in production environments.

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

Example
The following example is for COM Data Server. Siebel Application is an Application instance:

```vbscript
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
```

The following example is in Siebel eScript:

```javascript
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);
}
```

The following example is in Siebel VB:

```vbscript
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
```

The following is sample output of an Allocation trace section:
The following is sample output of an 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
```

The following examples show the use of Trace, Traceoff, and TraceOn methods to generate a trace file with SQL statements issues by the scripting query.

The following example is in Siebel eScript:

```javascript
function BusComp_NewRecord ()
{
    TheApplication().TraceOn("C:\trace_output.txt", "SQL", "");
    TheApplication().Trace("Start of tracing!");
    var oBC = this.GetPickListBusComp("Sales Stage");

    with (oBC)
    {
        SetViewMode(AllView);
        ClearToQuery();
        SetSortSpec("Sales Stage Order(ASCENDING)");
        ExecuteQuery(ForwardOnly);
        if (FirstRecord())
```

---

*The following is sample output of an 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
```

*The following examples show the use of Trace, Traceoff, and TraceOn methods to generate a trace file with SQL statements issues by the scripting query.*
The following example is in Siebel VB:

```vbnet
Sub BusComp_NewRecord
    TheApplication.TraceOn "C:\trace_output.txt", "SQL", ""
    TheApplication.Trace "Start of tracing!"
    Dim oBC as BusComp
    Set oBC = Me.GetPickListBusComp("Sales Stage")
    With oBC
        .SetViewMode AllView
        .ClearToQuery
        .SetSortSpec "Sales Stage Order(ASCENDING)"
        .ExecuteQuery ForwardOnly
        If .FirstRecord Then
            .Pick
        End If
    End With
    Set oBC = Nothing
    TheApplication.Trace "End of tracing!"
    TheApplication.TraceOff
End Sub
```

Related Topics

"Trace Method" on page 169
"TraceOff Method" on page 171

**Application Events**

The following topics describe application events:

- "Application_Close Event"
- "Application_InvokeMethod Event" on page 176
- "Application_Navigate Event" on page 177
- "Application_PreviewMethod Event" on page 177
- "Application_PreviewNavigate Event" on page 179
- "Application_Start Event" on page 180
**Application_Close Event**

The Close event is called before the application exits. This allows 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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

**Returns**

Not applicable

**Used With**

Server Script

*NOTE: Siebel Business Processes invokes this event. For more information, read Siebel Business Process Framework: Workflow Guide.*

**Application_InvokeMethod Event**

The Application_InvokeMethod event is called after a specialized method is invoked.

**Server Script Syntax**

Application_InvokeMethod(methodName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>methodName</td>
<td>Name of the method invoked</td>
</tr>
</tbody>
</table>

**Browser Script Syntax**

Application_InvokeMethod(name, inputPropSet)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputPropSet</td>
<td>A property set containing arguments to be passed to the InvokeMethod event.</td>
</tr>
</tbody>
</table>

**Returns**

Returns TRUE if the call succeeds or FALSE if it does not succeed.
**Usage**
The InvokeMethod event is called just after a specialized or user-defined method is invoked on the application.

The Browser script implementation does not return a property set.

**Used With**
Browser Script, Server Script

**Related Topics**
"How Your Script Affects Program Flow" on page 68
"Application_PreInvokeMethod Event"

**Application_Navigate Event**
The Application_Navigate event is called after the client has navigated to a view.

**Syntax**
Application_Navigate

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**
Not applicable

**Used With**
Server Script

**Application_PreInvokeMethod Event**
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)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>methodName</td>
<td>String variable or literal containing the name of the method invoked</td>
</tr>
</tbody>
</table>
Browser Script Syntax
Application_PreInvokeMethod (methodName, inputPropSet)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>methodName</td>
<td>String variable or literal containing the name of the method invoked.</td>
</tr>
<tr>
<td>inputPropSet</td>
<td>A property set containing arguments to be passed to the event.</td>
</tr>
</tbody>
</table>

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.

```vbnet
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
The following example is in Siebel VB and shows an implementation of the PreInvokeMethod:

```vbnet
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
    End Select
End Function
```
Case "LaunchExcel"
       i = Shell("C:\Program Files\Microsoft Office\Office\EXCEL.EXE",1)
       iReturn = CancelOperation
   End Select

   Application_PreInvokeMethod = iReturn

   End Function

The following 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":"C:\Program Files\Microsoft Office\Office\WINWORD.EXE",1);
       iReturn = CancelOperation;
       break;

       case "LaunchExcel":"C:\Program Files\Microsoft Office\Office\EXCEL.EXE",1);
       iReturn = CancelOperation;
       }

       return (iReturn);
   }

Related Topic
"How Your Script Affects Program Flow” on page 68

Application_PreNavigate Event

The Application_PreNavigate event is called before the client navigates to a view.

Syntax
Application_PreNavigate(DestViewName, DestBusObjName As String) As Integer

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DestViewName</td>
<td>Name of the view to which the user is navigating</td>
</tr>
<tr>
<td>DestBusObjName</td>
<td>Business object of the destination view</td>
</tr>
</tbody>
</table>

Returns
CancelOperation or ContinueOperation
**Used With**
Server Script

**Example**
In the following eScript code sample the script checks for the current business object (contact) and sets the current contact id as global variable (can be used for keeping context):

```javascript
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 Event**
The Start event is called when the client starts and again when the user interface is first displayed.

**Syntax**
Application_Start(commandline)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>commandline</td>
<td>Text of the command line with which the Siebel application was started.</td>
</tr>
</tbody>
</table>

**NOTE:** Siebel Business Processes invoke this event. For more information, read *Siebel Business Process Framework: Workflow 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:

```vbscript
Sub Application_Start(CommandLine As String)
Dim oEmpBusObj as BusObject
Dim oEmpBusComp as BusComp
Dim oEmpBusComp as BusComp
Dim sLoginName as String
Dim sUserName as String
sLoginName = TheApplication.LoginName
Set oEmpBusObj = TheApplication.GetBusObject("Employee")
Set oEmpBusComp = oEmpBusObj.GetBusComp("Employee")
With oEmpBusComp
  .ActivateField "First Name"
  .ActivateField "Last Name"
  .ClearToQuery
  .SetSearchSpec "Login Name", sLoginName
  .ExecuteQuery
  If (.FirstRecord = 1) Then
    sUserName = .GetFieldValue("First Name")
    sUserName = sUserName + " " + .GetFieldValue("Last Name")
  End If
End With
Set oEmpBusComp = Nothing
Set oEmpBusObj = Nothing
End Sub
```

**CAUTION:** Do not use the RaiseErrorText() method in the Application_Start event. That method does not work in this event, and can cause the Application Object Manager to abort.

### Business Component Methods

In the methods described in this section, the placeholders `oBusComp` and `BusComp` refer to a business component instance:

- "ActivateField Method” on page 183
- "ActivateMultipleFields Method” on page 184
- "Associate Method” on page 186
- "BusObject Method” on page 188
- "ClearToQuery Method” on page 189
- "DeactivateFields Method” on page 191
- "DeleteRecord Method” on page 193
- "ExecuteQuery Method” on page 193
- "ExecuteQuery2 Method” on page 195
- "FirstRecord Method” on page 196
■ “FirstSelected Method” on page 198
■ “GetAssocBusComp Method” on page 200
■ “GetFieldValue Method” on page 201
■ “GetFormattedFieldValue Method” on page 203
■ “GetLastErrCode Method” on page 205
■ “GetLastErrText Method” on page 206
■ “GetMultipleFieldValues Method” on page 206
■ “GetMVGBusComp Method” on page 207
■ “GetNamedSearch Method” on page 208
■ “GetPicklistBusComp Method” on page 209
■ “GetSearchExpr Method” on page 211
■ “GetSearchSpec Method” on page 212
■ “GetSortSpec Method” on page 212
■ “GetUserProperty Method” on page 213
■ “GetViewMode Method” on page 214
■ “InvokeMethod Method” on page 215
■ “LastRecord Method” on page 224
■ “Name Method” on page 225
■ “NewRecord Method” on page 225
■ “NextRecord Method” on page 227
■ “NextSelected Method” on page 228
■ “ParentBusComp Method” on page 228
■ “Pick Method” on page 229
■ “PreviousRecord Method” on page 231
■ “RefineQuery Method” on page 232
■ “Release Method” on page 233
■ “SetFieldValue Method” on page 235
■ “SetFormattedFieldValue Method” on page 237
■ “SetMultipleFieldValues Method” on page 238
■ “SetNamedSearch Method” on page 240
■ “SetSearchExpr Method” on page 242
■ “SetSearchSpec Method” on page 244
■ “SetSortSpec Method” on page 248
ActivateField Method

ActivateField allows queries to retrieve data for the argument-specified field.

Syntax

`BusComp.ActivateField(FieldName)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>String variable or literal containing the name of the field to activate</td>
</tr>
</tbody>
</table>

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 before 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:

- The business component is the instance on which the applet is based and the fields are displayed on the applet or, for fields in list applets, the Show In List list column property is TRUE.
- The fields are System fields (which include Id, Created, Created By, Updated, and Updated By).
- The fields’ Force Active property is set to TRUE.
- The ActivateField method has been invoked on the fields and an ExecuteQuery method has been executed afterwards.
- The fields 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 requerie 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.
The ActivateField method forces the specified field to be included in the SQL statement that is initiated by an ExecuteQuery method that follows. ActivateField should always be followed by ExecuteQuery. If a field is activated and then referenced by a GetFieldValue or SetFieldValue statement prior to an ExecuteQuery statement, the activation has no effect. The activated field is not retrieved through a query, so it contains an empty value.

If a field is not activated prior to a WriteRecord, the data is written to the database, but corruption issues may arise when mobile users synchronize. An ActivateField call prior to an ExecuteQuery call, followed by a WriteRecord, makes sure that the field is written correctly to the transaction log so that changes made by mobile users are saved back to the server database correctly at synchronization time.

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

**Example**
The following example is in Siebel VB. For an equivalent Siebel eScript example, read “ClearToQuery Method” on page 189.

```vbnet
Dim oEmpBusObj As BusObject
Dim oEmpBusComp As BusComp
Dim sLoginName As String

Set oEmpBusObj = TheApplication.ActiveBusObject
Set oEmpBusComp = oEmpBusObj.GetBusComp("Employee")
oEmpBusComp.SetViewMode AllView
oEmpBusComp.ClearToQuery
oEmpBusComp.SetSearchSpec "Login Name", sLoginName
oEmpBusComp.ExecuteQuery
Set oEmpBusComp = Nothing
Set oEmpBusObj = Nothing
```

**Related Topic**
"DeactivateFields Method” on page 191

**ActivateMultipleFields Method**
Use ActivateMultipleFields to activate data for the fields specified in the property set.
Syntax

\texttt{BusComp.ActivateMultipleFields(SiebelPropertySet \textit{sps})}

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SiebelPropertySet</td>
<td>Property set containing a collection of properties representing the fields that are to be activated</td>
</tr>
</tbody>
</table>

Returns

TRUE if success; FALSE if failure

Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

Example

The following example is for Java Data Bean:

```java
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 \textit{ps} = new mdata_bean.NewPropertySet();
\textit{ps}.setProperty("Account Products","");
\textit{ps}.setProperty("Agreement Name","");
\textit{ps}.setProperty("Project Name","");
\textit{ps}.setProperty("Description","");
\textit{ps}.setProperty("Name","");
siebBusComp.ActivateMultipleFields(\textit{ps});
...
```

The following Siebel eScript example queries the Contact business component and retrieves the First Name and Last Name of the first contact found:

```javascript
var ContactBO = TheApplication().GetBusObject("Contact");
var ContactBC = ContactBO.GetBusComp("Contact");
with (ContactBC)
{
    SetViewMode(AllView);
    var fieldsPS = TheApplication().NewPropertySet();
    var valuesPS = TheApplication().NewPropertySet();
    fieldsPS. SetProperty("Last Name","");
    fieldsPS. SetProperty("First Name","");
    ```
ActivateMultipleFields(fieldsPS);
ClearToQuery();
ExecuteQuery(ForwardBackward);
if (FirstRecord())
{
    GetMultipleFieldValues(fieldsPS, valuesPS);
    var slName = valuesPS.GetProperty("Last Name");
    var sfName = valuesPS.GetProperty("First Name");
}

**Related Topics**

"SetMultipleFieldValues Method" on page 238
"GetMultipleFieldValues Method" on page 206

**Associate Method**

The Associate method creates a new many-to-many relationship for the parent object through an association business component (see GetAssocBusComp).

**Syntax**

\[BusComp.Associate(\textit{whereIndicator})\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{whereIndicator}</td>
<td>This argument should be one of the following predefined constants: NewBefore or NewAfter, as in NewRecord.</td>
</tr>
</tbody>
</table>

**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**

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

**Example**

The following VB example updates the Opportunity Assignment Type field. The parent business component can be any business component that includes the Sales Rep multi-value group.
The following Siebel eScript example finds a contact with the Last Name = "Abanilla", and adds a new organization named "CKS Software" to its Organization MVG.

```javascript
var ok = 0;
var ContactBO = TheApplication().GetBusObject("Contact");
var ContactBC = ContactBO.GetBusComp("Contact");
with (ContactBC)
{
    ClearToQuery();
    SetViewMode(AllView);

    // Searches by Last Name
    SetSearchSpec("Last Name", "Abanilla");
    ExecuteQuery(ForwardOnly);
    if (FirstRecord())
    {
        // Instantiates Organization MVG
        var oMvgBC = GetMVGBusComp("Organization");
        var oAssocBC = oMvgBC.GetAssocBusComp();
        oAssocBC.ClearToQuery();
        oAssocBC.SetSearchSpec("Name", "CKS Software");
        oAssocBC.ExecuteQuery();

        // Checks if the Organization was found
        if (oAssocBC.FirstRecord())
        {
            // Organization was found
            try
            {
                oAssocBC.Associate(NewAfter);
                ok = 1;
            }
            catch (e)
            {
                ok = 0; // Catch error
            }
        }
    }
}
```

Dim oParentBC as BusComp
Dim oMvgBC as BusComp
Dim oAssocBC as BusComp

Set oParentBC = Action.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"
oMvgBC.WriteRecord
Set oAssocBC = Nothing
Set oMvgBC = Nothing
Set oParentBC = Nothing
catch (e)
{
    ok = 0;
    TheApplication().RaiseErrorText("Error Associating new Organization");
}
} // if oAssocBC.FirstRecord
} // if FirstRecord
oAssocBC = null;
oMvgBC = null;
} // With ContactBC
ContactBC = null;
ContactBO = null;

Related Topics
"NewRecord Method" on page 225
"FirstSelected Method" on page 198
"GetMVGBusComp Method" on page 207

**BusObject Method**
The BusObject method returns the business object that contains the business component.

**Syntax**
`BusComp.BusObject`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

**Returns**
The business object that contains the business component

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

**Example**
For an example, read "SetViewMode Method" on page 251.
ClearToQuery Method

The ClearToQuery method clears the current query but does not clear sort specifications on the BusComp.

Syntax

`BusComp.ClearToQuery`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns

Not applicable

Usage

Any fields to be queried must be activated before ClearToQuery. For more information, read "ActivateField Method" on page 183.

Search and sort specifications sent to the business component are cumulative; the business component retains and logically performs an AND operation on 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

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

Example

The following example is in Siebel eScript. For Siebel VB examples, read "Applet_PreInvokeMethod Event" on page 108, "ActivateField Method" on page 183, and "ExecuteQuery Method" on page 193. For another eScript example, read "GotoView Method" on page 139.

```javascript
var oEmpBusObj = TheApplication().ActiveBusObject();
var oEmpBusComp = oEmpBusObj().GetBusComp("Employee");
var sLoginName;

oEmpBusComp.ClearToQuery();
oEmpBusComp.SetSearchSpec("Login Name", sLoginName);
oEmpBusComp.ExecuteQuery(ForwardBackward);
```
Interfaces Reference ▶ Business Component Methods

```
oEmpBusComp = null;
oEmpBusObj = null;
```

**Related Topic**
“RefineQuery Method” on page 232

## CountRecords Method
CountRecords uses database aggregation to count the records returned by the last ExecuteQuery() call.

**Syntax**

```
BusComp.CountRecords()
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**
An integer indicating the number of records returned by the last ExecuteQuery() call.

**Used With**
Server Script

**Examples**
The following example is in Siebel eScript:

```javascript
function Service_PreInvokeMethod (MethodName, Inputs, Outputs)
{
    if (MethodName == "Call_eScript")
    {
        var bo = TheApplication().GetBusObject("Opportunity");
        var bc = bo.GetBusComp("Opportunity");
        with (bc)
        {
            ClearToQuery();
            SetSearchSpec ("Name", "A*");
            ExecuteQuery(ForwardBackward);
            var count = CountRecords();
        }
        // other code..

        bc = null;
        bo = null;
    }
```

190 Siebel Object Interfaces Reference Version 8.0, Rev. B
DeactivateFields Method

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

```java
BusComp.DeactivateFields
```

**Argument**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
</tr>
</tbody>
</table>

**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 Force Active property is set to TRUE.
- The ActivateField method has been invoked on them and an ExecuteQuery method has been executed afterwards.
- They have the Link Specification property set to TRUE.

After fields have been deactivated, the business component must be reexecuted or the application fails.

**Used With**

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script
Examples

The following example is for COM. **Siebel Application** is an Application instance.

```vbscript
Dim oBO As BusObject
Dim OBC As BusComp
Dim errCode

Set oBO = SiebelApplication.GetBusObject("Account", errCode)
Set oBC = oBO.GetBusComp("Account", errCode)
oBC.DeactivateFields errCode
oBC.ActivateField "Name", errCode
oBC.ActivateField "Location", errCode
oBC.ClearToQuery errCode
oBC.ExecuteQuery ForwardOnly, errCode
Set oBC = Nothing
Set oBO = Nothing
```

The following example is in Siebel eScript:

```javascript
var oBC;
var oBO;

oBO = TheApplication().GetBusObject("Account");
oBC = oBO.GetBusComp("Account");
oBC.DeactivateFields();
oBC.ActivateField("Name");
oBC.ActivateField("Location");
oBC.ClearToQuery();
oBC.ExecuteQuery(ForwardOnly);
oBC = null;
oBO = null;
```

The following example is in Siebel VB:

```vbscript
Dim oBO As BusObject
Dim oBC As BusComp

Set oBO = TheApplication.GetBusObject("Account")
Set oBC = oBO.GetBusComp("Account")
oBC.DeactivateFields
oBC.ActivateField "Name"
oBC.ActivateField "Location"
oBC.ClearToQuery
oBC.ExecuteQuery ForwardOnly
Set oBC = Nothing
Set oBO = Nothing
```

Related Topic

"ActivateField Method” on page 183
DeleteRecord Method

DeleteRecord removes the current record from the business component.

Syntax

BusComp.DeleteRecord

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns

Not applicable

Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

Example

This Siebel VB example illustrates how to delete accounts with a status of Inactive:

```vba
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. Do not use NextRecord after DeleteRecord in a loop because this causes the deletion of the last record in the loop to be skipped. If you use DeleteRecord on the last record, the cursor points to nothing.

ExecuteQuery Method

ExecuteQuery returns a set of business component records using the criteria established with methods such as SetSearchSpec.
Syntax

`BusComp.ExecuteQuery ([cursorMode])`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cursorMode</code></td>
<td>An integer. An optional argument that must be one of the following constants (provided in Siebel VB as well as COM Servers):</td>
</tr>
<tr>
<td></td>
<td>■ <strong>ForwardBackward.</strong> Selected records can be processed from first to last or from last to first. This is the default if no value is specified.</td>
</tr>
<tr>
<td></td>
<td>■ <strong>ForwardOnly.</strong> Selected records can be processed only from the first record to the last record. Focus cannot return to a record.</td>
</tr>
</tbody>
</table>

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 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.

**NOTE:** You must activate fields by using the `ActivateField` method before executing a query for a business component. 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.

Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

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 Event” on page 108, “GotoView Method” on page 139, and “ClearToQuery Method” on page 189.
Option Explicit

Function BusComp_PreSetFieldValue (FieldName As String,
FieldValue As String) As Integer
Dim i As Integer
Dim iFoundP As Integer ' 1 = found (TRUE), 0 = not found (FALSE)
Dim oMVGBC As BusComp

iFoundP = FALSE
Select Case FieldName
Case "SSA Primary Field"
    Set oMVGBC = me.ParentBusComp.GetMVGBusComp("Sales Rep")
    With oMVGBC ' this is the position BC
        .ActivateField "Active Login Name"
        .ActivateField "SSA Primary Field"
        .ClearToQuery
        .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
                    TheApplication.RaiseErrorText("You cannot change the Primary address
because you are not the Primary on the Account Team")
                End If
            End If
            i = .NextRecord
        Loop
    End With
End Select

Set oMVGBC = Nothing
BusComp_PreSetFieldValue = ContinueOperation

End Function

Related Topics
"ActivateField Method” on page 183
"ClearToQuery Method” on page 189
"SetSearchSpec Method” on page 244

ExecuteQuery2 Method

ExecuteQuery2 returns a set of business component records using the criteria established with methods such as SetSearchSpec.
Syntax

BusComp.ExecuteQuery2 ([cursorMode], ignoreMaxCursorSize)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cursorMode</td>
<td>An integer. An optional argument that can be one of the following two constants (provided in Siebel VB as well as COM Servers):</td>
</tr>
<tr>
<td></td>
<td>■ ForwardBackward. Selected records may be processed from first to last or from last to first. This is the default if no value is specified.</td>
</tr>
<tr>
<td></td>
<td>■ ForwardOnly. Selected records can be processed only from the first record to the last record. Focus cannot return to a record.</td>
</tr>
<tr>
<td>ignoreMaxCursorSize</td>
<td>■ TRUE. Retrieves every row from a business component. This option may result in lower performance.</td>
</tr>
<tr>
<td></td>
<td>■ FALSE. Retrieves the number of rows specified by the MaxCursorSize argument in the CFG file.</td>
</tr>
</tbody>
</table>

Returns

Not applicable

Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

FirstRecord Method

FirstRecord moves the record pointer to the first record in a business component, making that record current and invoking any associated script events.

NOTE: When executing a query on a business component, SQL is generated for any active child business component. Calling the FirstRecord method triggers the BusComp_ChangeRecord event and causes the same SQL for the child business component to execute again.

Syntax

BusComp.FirstRecord

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

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, and ActiveX.
Used With
COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

Examples
The following examples show how the FirstRecord method could be used to check whether an Account displayed in a child applet (for example, the Account List Applet - child applet in the Contact Detail - Accounts View) has any service requests associated to it. The outcome of this could then determine whether other code should be run against the Account record.

The following example is in Siebel eScript:

```plaintext
function BusComp_PreInvokeMethod (MethodName)
{
    // 'CheckSR' method invoked from a custom button on 'Account List Applet - child' applet.
    if (MethodName == "CheckSR")
    {
        var oBO = TheApplication().ActiveBusObject();
        var oBC = oBO.GetBusComp("Service Request");
        var strAccntId = this.GetFieldValue("Id");
        with (oBC)
        {
            SetViewMode(AllView);
            ClearToQuery();
            SetSearchSpec("Account Id", strAccntId);
            ExecuteQuery(ForwardOnly);
            if (FirstRecord())
            {
                // additional code placed here
            }
            else
            {
                TheApplication().RaiseErrorText("No Service Requests Associated To This Account.");
            }
        }
    }
    return (CancelOperation);
}
return (ContinueOperation);
```

The following example is in Siebel VB:

```plaintext
Function BusComp_PreInvokeMethod (MethodName As String) As Integer
    Dim iRtn As Integer
   ...
```

iRtn = ContinueOperation

' CheckSR method invoked from a custom button on 'Account List Applet - child' Applet.
If MethodName = "CheckSR" Then
    Dim oBO As BusObject
    Dim oBC As BusComp
    Dim strAccntId As String
    Set oBO = TheApplication.ActiveBusObject
    Set oBC = oBO.GetBusComp("Service Request")
    strAccntId = me.GetFieldValue("Id")
    With oBC
        .SetViewMode AllView
        .ClearToQuery
        .SetSearchSpec "Account Id", strAccntId
        .ExecuteQuery ForwardOnly
        If .FirstRecord Then
            ' [additional code placed here]
        Else
            TheApplication.RaiseErrorText("No Service Requests Associated To This Account.")
        End If
    End With
    Set oBC = Nothing
    Set oBO = Nothing
    iRtn = CancelOperation
    End If
End If
BusComp_PreInvokeMethod = iRtn
End Function

Related Topic
"NextRecord Method" on page 227

FirstSelected Method

FirstSelected moves the focus to the first record of the multiple selection in the business component, invoking any associated Basic events.

Syntax
BusComp.FirstSelected

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>
**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 ActiveX, COM, and Siebel eScript.

**Used With**
COM Data Server, Server Script

**Examples**
The following examples show how the FirstSelected method could be used in conjunction with the NextSelected method to provide custom multirecord deletion functionality. This code could be triggered in respect to the user invoking the Delete Selected custom method, when pressing a custom button on an applet.

The following example is in Siebel eScript:

```javascript
function BusComp_PreInvokeMethod (MethodName)
{
    if (MethodName == "Delete Selected")
    {
        with (this)
        {
            var iRecord = FirstSelected();
            while (iRecord)
            {
                DeleteRecord();
                iRecord = NextSelected();
            }
        }
        return (CancelOperation);
    }
    return (ContinueOperation);
}
```

The following example is in Siebel VB:

```vbnet
Function BusComp_PreInvokeMethod (MethodName As String) As Integer
    Dim iRtn As Integer
    iRtn = ContinueOperation
    If MethodName = "Delete Selected" Then
        With me
            Dim iRecord As Integer
            iRecord = .FirstSelected
```

---

**Interfaces Reference**

**Business Component Methods**

**Siebel Object Interfaces Reference** Version 8.0, Rev. B  **199**
GetAssocBusComp Method

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`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns

The association business component for a 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 add 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 M:M Link is used and the Child Applet has an Association Applet defined.

Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script
Example
The following example is in Siebel VB and uses GetAssocBusComp to add a new industry to an account record:

```vbnet
Dim oAssocBC As BusComp
Set oAssocBC = oMainBc.GetMVGBusComp("Industry").GetAssocBusComp
With oAssocBC
    .ClearToQuery
    .SetSearchExpr "[SIC Code] = ""5734"
    .ExecuteQuery ForwardOnly
    If .FirstRecord Then .Associate NewBefore
End With
Set oAssocBC = Nothing
```

The following is the equivalent Siebel eScript code:

```javascript
// get the business Object and the business component
var oAssocBC = oMainBc.GetMVGBusComp("Industry").GetAssocBusComp();
with (oAssocBC)
{
    ClearToQuery;
    SetSearchExpr("[SIC Code] = '5734'");
    ExecuteQuery(ForwardOnly)
    if (FirstRecord())
        Associate(NewBefore);
}
oAssocBC = null;
```

Related Topics
"GetMVGBusComp Method” on page 207
"GetPicklistBusComp Method” on page 209

GetFieldValue Method
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)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>String variable or literal containing the name of the field</td>
</tr>
</tbody>
</table>
Returns
A string containing the field value of the field identified in FieldName, an error message if the field is inactive, or an empty string if the field is empty.

NOTE: Date fields retrieved by GetFieldValue() are always returned using the format MM/DD/YYYY, no matter what your local date format is set to. Use GetFormattedFieldValue() to get the same date format you use in the client interface.

Usage
Only fields that were active at the time of the BusComp query contain values. For more information, read "ActivateField Method" on page 183. If this method is used on fields that are not active, an error message is returned. If this method is used on fields that are empty, an empty string is returned.

CAUTION: If a value from a business component that is a parent of the current business component is desired, the Link Specification property for that field must be set to TRUE in Siebel Tools. Otherwise, the child business component cannot access the value in the parent business component. For more information on the Link Specification property, see Siebel Object Types Reference.

The FieldName must be enclosed in double quotes and must be spelled exactly as the field name appears in Siebel Tools, 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 and for system fields.

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

Example
The following example is in Siebel VB. It shows an implementation of the PreSetFieldValue event to illustrate 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"
    End With
```

The following is the equivalent example in Siebel eScript.

```javascript
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;
end function
```

Related Topics

"ActivateField Method“ on page 183
"GetFormattedFieldValue Method“

GetFormattedFieldValue Method

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)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>String variable or literal containing the name of the field to obtain the value from</td>
</tr>
</tbody>
</table>

**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:
```cpp
phone = bc.GetFieldValue("Main Phone Number")
TheApplication.Trace "The number is " & phone
```
Result:
```
The number is 8869629123
```
Example 2:
```cpp
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 34 shows the standard formats used by GetFieldValue and SetFieldValue to return data.

<table>
<thead>
<tr>
<th>Type of Data</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dates</td>
<td>mm/dd/yyyy</td>
</tr>
<tr>
<td>Times</td>
<td>hh:nn:ss</td>
</tr>
<tr>
<td>Date-times</td>
<td>mm/dd/yyyy hh:nn:ss</td>
</tr>
</tbody>
</table>

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, COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script
Example
The following Siebel VB example demonstrates how to use the GetFormattedFieldValue function and how to calculate the number of days between two dates.

```vbnet
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
```

Related Topics
“ActivateField Method” on page 183
“GetFieldValue Method” on page 201
“SetFieldValue Method” on page 235
“SetFormattedFieldValue Method” on page 237

GetLastErrorCode Method
The GetLastErrorCode method returns the most recent error code on the business component level.

Syntax
`BusComp.GetLastErrorCode`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns
The last error code as a short integer. 0 indicates no error.

Usage
After execution of a method, the GetLastErrorCode can be invoked to check if any error was returned from the previous operation. The GetLastErrorText method can be invoked to retrieve the text of the error message. The text retrieved using GetLastErrorText also includes a Siebel error number that can be used to search Oracle MetaLink for additional information about the error.

Used With
COM Data Control, Mobile Web Client Automation Server
GetLastErrText Method

The GetLastErrText method returns the last error text message on the business component level.

Syntax

```
BusComp.GetLastErrText
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

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

COM Data Control, Mobile Web Client Automation Server

Related Topic

"GetLastErrCode Method"

GetMultipleFieldValues Method

GetMultipleFieldValues returns values for the fields specified in the property set.

Syntax

```
BusComp.GetMultipleFieldValues(SiebelPropertySet fieldNames, SiebelPropertySet fieldValues)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldNames</td>
<td>A property set containing a collection of properties representing the fields</td>
</tr>
<tr>
<td>fieldValues</td>
<td>A property set containing a collection of properties representing the values for the fields specified in the <code>fieldNames</code> argument</td>
</tr>
</tbody>
</table>

Returns

TRUE if success; FALSE if failure
**GetMVGBusComp Method**

The GetMVGBusComp method 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**

```plaintext
BusComp.GetMVGBusComp(FieldName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FieldName</code></td>
<td>Name of the field with a multi-value group attached, used to obtain the multi-value group business component</td>
</tr>
</tbody>
</table>

**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 multi-value field.

The GetMVGBusComp business component should be set to Nothing (Siebel VB) or null (Siebel eScript) 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 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**

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script
Example
The following sample Siebel VB code using GetMVGBusComp inserts a new address to the "Hong Kong Flower Shop" account record. For other examples, read "ExecuteQuery Method" on page 193 and "FirstSelected Method" on page 198.

```vbscript
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"
    .ClearToQuery
    .SetSearchSpec "Name", "Hong Kong Flower Shop"
    .ExecuteQuery
    If (.FirstRecord) Then Set AddrBC = .GetMVGBusComp "Street Address"
End With

With AddrBC
    .NewRecord "NewAfter"
    .SetFieldValue "City", "Denver"
    .SetFieldValue "Street Address", "123 Main Street"
    .WriteRecord
End With

Set AddrBC = Nothing
Set AccntBC = Nothing
Set AccntBO = Nothing
```

Related Topics
"FirstSelected Method" on page 198
"GetPicklistBusComp Method"

GetNamedSearch Method
GetNamedSearch returns the named search specification specified by searchName.

Syntax
`BusComp.GetNamedSearch(searchName)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>searchName</code></td>
<td>Name of the search specification that references the search string.</td>
</tr>
</tbody>
</table>

Returns
A string containing the value specified in the search specification identified in `searchName`
Usage
The search specification uses the same syntax as used in predefined queries.

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

Related Topics
"GetSearchExpr Method” on page 211  
"GetSearchSpec Method” on page 212  
“SetNamedSearch Method” on page 240

GetPicklistBusComp Method
GetPicklistBusComp returns the pick business component associated with the specified field in the current business component.

Syntax
BusComp.GetPicklistBusComp(FieldName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>Name of the field with a picklist specified; used to obtain the pick business component</td>
</tr>
</tbody>
</table>

Returns
The pick business component of the current business component and identified field. If there is no picklist associated with that field, the function returns an error.

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 (Siebel VB) or null function (eScript or Browser Script).

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**
COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

**Example**
The following example is 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;
}
```

The following example is for Java Data Bean. It selects a product from a picklist.
```
Sieb_busObject = Sieb_dataBean.getBusObject("Service Request");
Sieb_busComp = Sieb_busObject.getBusComp("Service Request");
Sieb_busComp.newRecord(false);

SiebelBusComp productBusComp = Sieb_busComp.getPicklistBusComp("Product");
productBusComp.clearToQuery();
productBusComp.setSearchSpec("Name", "ATM Card");
productBusComp.executeQuery(false);
isRecord = productBusComp.firstRecord();
try
{
    if (isRecord)
        productBusComp.pick();
    Sieb_busComp.writeRecord();
}
catch (SiebelException e)
{
    System.out.println("Error in Pick " + e.getMessage());
}
```

The following example is in Siebel VB:
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

Related Topics
"FirstSelected Method" on page 198
"GetMVGBusComp Method" on page 207

GetSearchExpr Method
GetSearchExpr returns the current search expression for the business component.

Syntax
BusComp.GetSearchExpr

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns
A string containing the current search expression. An example of a returned search expression string is the following:

"[Revenue] > 10000 AND [Probability] > .5"

Usage
GetSearchSpec retrieves the business component state, not the values. The business component state does not change until the query is executed. Note that it may never change to the original value if the user input is invalid.

When using GetSearchExpr in a browser script and the Applet_PreInvokeMethod, GetSearchExpr returns a null value even if a query filter has been added.

Used With
Browser Script, COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script
**GetSearchSpec Method**

GetSearchSpec returns the search specification for the field specified by the `FieldName` argument.

**Syntax**

```java
BusComp.GetSearchSpec(FieldName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FieldName</code></td>
<td>Contains the name of the field from which to obtain the associated search specification.</td>
</tr>
</tbody>
</table>

**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, COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

**Related Topics**

"GetNamedSearch Method“ on page 208
"GetSearchExpr Method”
"SetSearchExpr Method“ on page 242

---

**GetSortSpec Method**

GetSortSpec returns the active sort specification of the object that has context.

**Syntax**

```javascript
this.GetSortSpec();
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>
## getUserProperty Method

GetUserProperty returns the value of a named user property.

### Syntax

```plaintext
BusComp.GetUserProperty(propertyName)
```

### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>propertyName</td>
<td>Contains the name of the user property to obtain.</td>
</tr>
</tbody>
</table>

### 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

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

### Related Topic

“`SetUserProperty Method` on page 250”
GetViewMode Method

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, see “SetViewMode Method” on page 251.

Syntax

BusComp.GetViewMode

Returns

An integer constant that identifies a visibility mode

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>Where mode is a Siebel ViewMode constant or its corresponding integer value. The constants shown are defined in three environments. For details on each Siebel ViewMode constant, read “SetViewMode Method” on page 251.</td>
</tr>
<tr>
<td></td>
<td>- SalesRepView (0)</td>
</tr>
<tr>
<td></td>
<td>- ManagerView (1)</td>
</tr>
<tr>
<td></td>
<td>- PersonalView (2)</td>
</tr>
<tr>
<td></td>
<td>- AllView (3)</td>
</tr>
<tr>
<td></td>
<td>- OrganizationView (5)</td>
</tr>
<tr>
<td></td>
<td>- GroupView (7)</td>
</tr>
<tr>
<td></td>
<td>- CatalogView (8)</td>
</tr>
<tr>
<td></td>
<td>- SubOrganizationView (9)</td>
</tr>
</tbody>
</table>

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 the most restrictive visibility mode defined for that business component.

Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script
Related Topic
"SetViewMode Method” on page 251

InvokeMethod Method
InvokeMethod calls the specialized method or user-created method named in the argument.

VB Syntax
BusComp.InvokeMethod methodName, methodArgs

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>methodName</td>
<td>The name of the method. For more information on the available methods, read</td>
</tr>
<tr>
<td></td>
<td>&quot;InvokeMethod Methods for the Business Component Object” on page 216.</td>
</tr>
<tr>
<td>methodArgs</td>
<td>A single string or a string array (object interfaces) containing arguments to</td>
</tr>
<tr>
<td></td>
<td>methodName.</td>
</tr>
</tbody>
</table>

eScript Syntax
BusComp.InvokeMethod(methodName, methArg1, methArg2, ..., methArgn);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>methodName</td>
<td>The name of the method</td>
</tr>
<tr>
<td>methArg1,</td>
<td>One or more strings containing arguments to methodName</td>
</tr>
<tr>
<td>methArg2,</td>
<td></td>
</tr>
<tr>
<td>..., methArgn</td>
<td></td>
</tr>
</tbody>
</table>

Returns
A string containing the result of the method

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. Oracle does not support calling specialized methods with InvokeMethod, unless they are listed in this book.

Used With
COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script
Example
For examples of the usage of InvokeMethod, see “ClearLOVCache Method” on page 216, “CreateFile Method” on page 218, “GetFile Method” on page 220, and “PutFile Method” on page 221.

InvokeMethod Methods for the Business Component Object
Siebel applications provide multiple methods for performing useful operations, such as manipulating files stored in the Siebel File System and refreshing records and business components. These methods can be invoked using server script (Siebel VB, eScript) or using one of the programmatic interfaces (Mobile Web Client Automation Server – connected mode only, COM Data Control, Java Data Bean).

The following methods are available for use with InvokeMethod:

- "ClearLOVCache Method” on page 216
- “CreateFile Method” on page 218
- “GenerateProposal Method” on page 219
- “GetFile Method” on page 220
- “PutFile Method” on page 221
- “RefreshBusComp Method” on page 222
- “RefreshRecord Method” on page 223
- “SetAdminMode Method” on page 223

The methods available for manipulating the file system (CreateFile, GetFile, and PutFile) always store the file to or retrieve the file from the file system local to the server on which the script is being executed. 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 COM Data Control, Java Data Bean, Mobile Web Client Automation Server, and Server Script.

ClearLOVCache Method
This method clears the object manager list of values (LOV) cache, functioning similarly to the Clear Cache button in the Administration - Data > List of Values view.

NOTE: ClearLOVCache clears only the object manager cache, not the session cache in the High Interactivity client.
Syntax
BusComp.InvokeMethod("ClearLOVCache")

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>

Returns
Not Applicable

Used With
This method is supported by BusComp.InvokeMethod() calls in Browser Script, COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, and Server Script.

Example
The following Siebel eScript example is for Server Script:

```javascript
function WebApplet_PreInvokeMethod (MethodName)
{
    if (MethodName == "TestMethod") {
        var lov_bo = TheApplication().GetBusObject("List Of Values");
        var lov_bc = lov_bo.GetBusComp("List Of Values");
        lov_bc.NewRecord(NewAfter);
        lov_bc.SetFieldValue("Type", "ACCOUNT_STATUS");
        lov_bc.SetFieldValue("Name", "Hello");
        lov_bc.SetFieldValue("Value", "Hello");
        lov_bc.SetFieldValue("Order By", "12");
        lov_bc.SetFieldValue("Translate", "Y");
        lov_bc.WriteRecord();
        lov_bc.InvokeMethod("ClearLOVCache");
        lov_bc = null;
        lov_bo = null;
        return (CancelOperation);
    }
    return(ContinueOperation);
}
```
CreateFile Method
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
BusComp.InvokeMethod("CreateFile", SrcFilePath, KeyFieldName, KeepLink)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SrcFilePath</td>
<td>The fully qualified path of the file on the Siebel Server or Mobile Web Client.</td>
</tr>
<tr>
<td>KeyFieldName</td>
<td>The name of the field in the business component that contains the File Name, for example, AccntFileName in the Account Attachment business component.</td>
</tr>
<tr>
<td>KeepLink</td>
<td>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.</td>
</tr>
</tbody>
</table>

Returns
A string containing the value “Success” or “Error” depending on whether or not the operation succeeded.

Used With
This method is supported by BusComp.InvokeMethod() calls in COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, and Server Script.

Example
The following example is in Siebel VB:
```vb
Dim RetValue as String
Dim fileBC as BusComp

' Instantiate fileBC as the appropriate attachment business component
fileBC.NewRecord NewAfter
RetValue = fileBC.InvokeMethod("CreateFile", "c:\Demo\Image.bmp", "AccntFileName", "Y")
fileBC.WriteRecord
```

The following example is in Siebel eScript:
```javascript
var fileBC;

// Instantiate fileBC as the appropriate attachment business component
```
fileBC.NewRecord(NewAfter);
RetVal = fileBC.InvokeMethod("CreateFile", "C:\Demo\Image.bmp",
"AccntFileName", "Y");
fileBC.WriteRecord();

The following example is in COM Data Control:

Dim errCode as Integer
Dim Args(2) as String
Dim RetValue as String
Dim fileBC as BusComp

' Instantiate fileBC as the appropriate attachment business component
Args(0) = "C:\Demo\Image.bmp"
Args(1) = "AccntFileName"
Args(2) = "Y"

fileBC.NewRecord NewAfter, errCode
RetVal = fileBC.InvokeMethod("CreateFile", Args, errCode)
fileBC.WriteRecord

GenerateProposal Method

GenerateProposal creates a new proposal record. The DocServer handles the work of generating the actual proposal.

Syntax
To specify a template:
BusComp.InvokeMethod("GenerateProposal", RecordExists, Replace, TemplateFile)

To use the default proposal template:
BusComp.InvokeMethod("GenerateProposal", RecordExists, Replace)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordExists</td>
<td>If FALSE, then a new record is created and used to create a new proposal.</td>
</tr>
<tr>
<td></td>
<td>If TRUE, the current selected proposal is used.</td>
</tr>
<tr>
<td>Replace</td>
<td>If TRUE, the template file is copied from the template into the proposal (as a draft file). You should typically call this method with this argument set to FALSE.</td>
</tr>
<tr>
<td>TemplateFile</td>
<td>(Optional) The default value of this argument is NULL.</td>
</tr>
<tr>
<td></td>
<td>A string that specifies the name of the template to use. When a string is passed into this argument, the proposal searches for the first template record whose name contains the string passed rather than using the default template.</td>
</tr>
</tbody>
</table>

Used With

This method is supported by BusComp.InvokeMethod() calls in Browser Script, COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, and Server Script.
GetFile Method
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
`BusComp.InvokeMethod("GetFile", KeyFieldName)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>KeyFieldName</code></td>
<td>The name of the field in the business component that contains the File Name, for example, <code>AccntFileName</code> in the Account Attachment business component.</td>
</tr>
</tbody>
</table>

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
This method is supported by `BusComp.InvokeMethod()` calls in COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, and Server Script.

Example
The following example uses Siebel VB:
```
Dim RetValue as String
Dim fileBC as BusComp

' Instantiate fileBC as the appropriate attachment business component
' Query for the desired attachment record
RetValue = fileBC.InvokeMethod("GetFile", "AccntFileName")
```

The following example uses Siebel eScript:
```
var RetValue;
var fileBC;

// Instantiate fileBC as the appropriate attachment business component
// Query for the desired attachment record
var RetValue = fileBC.InvokeMethod("GetFile", "AccntFileName");
```

The following example uses COM Data Control:
Dim errCode as Integer
Dim Args as String
Dim RetValue as String
Dim fileBC as BusComp

' Instantiate fileBC as the appropriate attachment business component

' Query for the desired attachment record

Args = "AccontFileName"
RetValue = fileBC.InvokeMethod("GetFile", Args, errCode)

**PutFile Method**
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**

\[ BusComp.InvokeMethod("PutFile", SrcFilePath, KeyFieldName) \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SrcFilePath</strong></td>
<td>This is the fully qualified path of the file on the Siebel Server or Mobile Web Client.</td>
</tr>
<tr>
<td><strong>KeyFieldName</strong></td>
<td>This is the name of the field in the business component that contains the File Name. For example: AccontFileName field in the Account Attachment business component.</td>
</tr>
</tbody>
</table>

**Returns**
A string containing the values of “Success” or “Error” depending on whether or not the operation succeeded.

**Usage**
After using PutFile to save a file attachment the updated attachment is not visible in the user interface until you call the WriteRecord method. For more information about WriteRecord, read “WriteRecord Method” on page 255.

**Used With**
This method is supported by BusComp.InvokeMethod() calls in COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, and Server Script.

**Example**
The following example uses Siebel VB:

```vba
Dim RetValue as String
Dim fileBC as BusComp
```
The following example uses Siebel eScript:

```javascript
var RetValue;
var fileBC;

// Instantiate fileBC to the appropriate attachment business component
// Query for the attachment record to be updated
RetValue = fileBC.InvokeMethod("PutFile", "c:\Demo\Image.bmp", "AccntFileName");
fileBC.WriteRecord();
```

The following example uses COM Data Control:

```vbscript
Dim errCode as Integer
Dim Args(1) as String
Dim RetValue as String
Dim fileBC as BusComp

' Instantiate fileBC to the appropriate attachment business component
' Query for the attachment record to be updated
Arg(0) = "C:\Demo\Image.bmp"
Arg(1) = "AccntFileName"
RetValue = fileBC.InvokeMethod("PutFile", Arg, errCode)
fileBC.WriteRecord()
```

### RefreshBusComp Method

This method re-executes the current query for the business component and places the focus back onto the record that was previously highlighted. The user sees that the data is refreshed but the same record is still highlighted in the same position in the list applet as before the RefreshBusComp method was invoked.

**Syntax**

`BusComp.InvokeMethod("RefreshBusComp")`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

Not Applicable
Interfaces Reference  ■ Business Component Methods

RefreshRecord Method
This method refreshes the currently highlighted record, which triggers an update of the business component fields in the client display and positions the cursor on the context record. Other records currently exposed in the user interface are not refreshed.

Syntax
retVal = BusComp.InvokeMethod("RefreshRecord")

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>

Returns
Not Applicable

SetAdminMode Method
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
BusComp.InvokeMethod("SetAdminMode", flag)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flag</td>
<td>&quot;TRUE&quot; or &quot;FALSE&quot;. Flag to specify whether the business component should be executed in Admin mode.</td>
</tr>
</tbody>
</table>

Returns
Not Applicable

NOTE: This method only works with business components that are derived from CSSBCBase.
Used With
This method is supported by BusComp_InvokeMethod() calls in COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, and Server Script.

LastRecord Method
LastRecord moves the record pointer to the last record in the business component.

Syntax
BusComp_LastRecord

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

Returns
An integer in Siebel VB; a Boolean in ActiveX, COM, Java Data Bean, Siebel eScript.

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

Example
The following example is for Mobile Web Client Automation Server. SiebelApplication is an Application instance:

```vbs
Private Sub LastRecord_Click()
    Dim errCode As Integer
    Dim oBusComp as SiebelBusComp
    FieldValue.Text = ""
    oBusComp.ClearToQuery
    oBusComp.ExecuteQueryForwardBackward
    oBusComp.LastRecord errCode
    If errCode = 0 Then
        FieldValue.Text = oBusComp.GetFieldValue(FieldName.Text, _
            errCode)
    End If
    Status.Text = SiebelApplication.GetLastErrText
End Sub
```

Related Topics
"FirstRecord Method" on page 196
"NextRecord Method" on page 227
Name Method
The Name property contains the name of the business component.

Syntax
BusComp.Name()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

Returns
A string containing the business component name

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

Example
The following example is in Browser Script:

```javascript
function BusComp_PreSetFieldValue (fieldName, value)
{
  theApplication().SWEAlert(this.Name());
}
```

NewRecord Method
NewRecord adds a new record (row) to the business component.
Syntax

_busComp_.NewRecord(\texttt{whereIndicator})

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{whereIndicator}</td>
<td>Predefined constant indicating where the new row is added. This value should be one of the following:</td>
</tr>
<tr>
<td></td>
<td>■ NewBefore</td>
</tr>
<tr>
<td></td>
<td>■ NewAfter</td>
</tr>
<tr>
<td></td>
<td>■ NewBeforeCopy</td>
</tr>
<tr>
<td></td>
<td>■ NewAfterCopy</td>
</tr>
<tr>
<td>With Java Data Bean the values are:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ FALSE (equivalent to NewBefore)</td>
</tr>
<tr>
<td></td>
<td>■ TRUE (equivalent to NewAfter)</td>
</tr>
</tbody>
</table>

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.

\begin{verbatim}
Object.NewRecord NewBeforeCopy
\end{verbatim}

To place the copy after the original record, use the following command.

\begin{verbatim}
Object.NewRecord NewAfterCopy
\end{verbatim}

In certain cases, using the NewRecord method in a server script results in slow performance for this method. There is no error message shown and the new record is created, but the response time is not optimal. This is due to the expected behavior of the application when creating new records.

Before an application inserts a new record into the database, it must obtain the cursor for the record set to position the new record in the record set. This requires the record set to be populated before creating the new record. In the context of a script, a query must be run on the business component before the NewRecord method is called. If the query is not explicitly run in the script, the application will force a query to run to bring back the cursor. The application will normally run a full table query, which results in the performance issue.

For more information on performance issues with the NewRecord method, see Doc ID 477556.1 on OracleMetaLink 3.
Used With
COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

Example
The following example is in Siebel VB:

```vbnet
dim oBusObj as BusObject
dim oBC as BusComp

set oBusObj = TheApplication.ActiveBusObject
do while oBusObj is true
    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 Method

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

```vbnet
BusComp.NextRecord
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns

An integer in Siebel VB; a Boolean in Siebel eScript and COM: 1 if the record pointer was moved to the next record, 0 if the current record was already the last record.

Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

Example

The following example is in Siebel eScript. For a similar Siebel VB example, see “FirstRecord Method” on page 196.

```eScript
var isRecord;
```
```java
with (this)
{
    ClearToQuery();
    SetSearchSpec("Name", "A*" quả);
    ExecuteQuery(ForwardBackward);
    isRecord = FirstRecord();
    while (isRecord)
    {
        // do some record manipulation
        isRecord = NextRecord();
    }
}
```

**Related Topic**
"FirstRecord Method” on page 196

### NextSelected Method

NextSelected moves the focus to the next record of the current multiple selection.

**Syntax**

```
BusComp.NextSelected
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

An integer: 1 if there is another record in the multiple selection, 0 otherwise.

**Used With**

Server Script

**Example**

For examples, read "FirstSelected Method” on page 198.

### ParentBusComp Method

ParentBusComp returns the parent (master) business component when given the child (detail) business component of a Link.
Syntax

BusComp.ParentBusComp

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

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.

CAUTION: If a value from a business component that is a parent of the current business component is desired, the Link Specification property for that field must be set to TRUE in Siebel Tools. Otherwise, the child business component cannot access the value in the parent business component. For more information on the Link Specification property, see Siebel Object Types Reference.

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

Example
The following example is in Siebel VB. For another example, read “ExecuteQuery Method” on page 193.

```vbscript
Dim strParentName as String
...
strParentName = Me.ParentBusComp.GetFieldValue("Name")
```

Pick Method

The Pick method places the currently selected record in a picklist business component into the appropriate fields of the parent business component.

NOTE: Pick cannot be used to change the record in a read-only picklist field.

Syntax

BusComp.Pick

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>
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
COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

Example
This Siebel VB example sorts the values in the Sales Stage field:

```vbscript
Sub BusComp_NewRecord
  Dim oBC as BusComp
  set oBC = me.GetPickListBusComp("Sales Stage")
  With oBC
    .ClearToQuery
    .SetSearchSpec "Sales Stage", "2 - Qualified"
    .ExecuteQuery ForwardOnly
    if .FirstRecord then .Pick
  End With
  set oBC = Nothing
End Sub
```

The following is the equivalent example in Siebel eScript:

```javascript
function BusComp_NewRecord ()
{
  var oBC = this.GetPickListBusComp("Sales Stage");
  with (oBC)
  {
    ClearToQuery();
    SetSearchSpec("Sales Stage", "2 - Qualified");
    ExecuteQuery(ForwardOnly);
    if (FirstRecord())
      Pick();
  }
  oBC = null;
}
```

Related Topic
"GetPicklistBusComp Method“ on page 209
PreviousRecord Method

PreviousRecord moves the record pointer to the next record in the business component, making that the current record and invoking any associated script events.

Syntax

`BusComp.PreviousRecord`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns

An integer in Siebel VB; a Boolean in Siebel eScript, COM, and ActiveX.

Usage

PreviousRecord may be used only on a business component that has been queried using the ForwardBackward CursorMode.

Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

Example

The following Siebel eScript example finds the next-to-last record in a query and does some manipulation with it:

```javascript
with (this) {
    ActivateField("Name")
    ClearToQuery();
    SetSearchSpec("Name", "A*");
    ExecuteQuery(ForwardBackward);
    isRecord = FirstRecord();
    while (isRecord)
    {
        // do some record manipulation
        isRecord = NextRecord();
    }
```
Related Topic
“ExecuteQuery Method” on page 193

RefineQuery Method
This method refines a query after the query has been executed.

Syntax
`BusComp.RefineQuery`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

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
COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

Example
The following Siebel VB code fragment shows how RefineQuery might be used:
me.SetSearchSpec 'Status', 'Open'
me.ClearToQuery
me.ExecuteQuery
me.RefineQuery
me.SetSearchSpec 'Substatus', 'Assigned'
me.ExecuteQuery

Related Topics
"ClearToQuery Method" on page 189
"GetNamedSearch Method" on page 208

**Release Method**

The Release() method enables the release of the business component and its resources on the Siebel Server.

**Syntax**

```
BusComp.release()
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

**Returns**

Not applicable

**Used With**

Java Data Bean

**Example**

The following example is for Java Data Bean:

```java
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.
}\)

The following example logs in to a Siebel Server. It then instantiates a business object, a business component, and a business service. Then, it releases them in reverse order.

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

public class JDBReleaseDemo
{
    private SiebelDataBean m_dataBean = null;
    private SiebelBusObject m_busObject = null;
    private SiebelBusComp   m_busComp = null;
    private SiebelService  m_busServ = null;

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

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

            // login to the servers
            m_dataBean.login("siebel.tcpip.none.none://<gateway>:<port>/<enterprise>/<
            object manager>","<user id>","<password>");
            System.out.println("Logged in to the Siebel server ");

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

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

            // get the business service
            m_busServ = m_dataBean.getService("Workflow Process Manager");

            // release the business service
            m_busServ.release();
            System.out.println("BS released ");

            // release the business component
            m_busComp.release();
            System.out.println("BC released ");
        }
    }
```
// release the business object
m_busObject.release();
System.out.println("BO released ");

// logoff
m_dataBean.logoff();
System.out.println("Logged off the Siebel server ");
}
catch (SiebelException e)
{
    System.out.println(e.getErrorMessage());
}
}

Related Topic
"Logoff Method" on page 152

SetFieldValue Method

SetFieldValue assigns the new value to the named field for the current row of the business component.

Syntax

BusComp.SetFieldValue FieldName, FieldValue

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>String containing the name of the field to assign the value to</td>
</tr>
<tr>
<td>FieldValue</td>
<td>String containing the value to assign</td>
</tr>
</tbody>
</table>

Returns

Not applicable

Usage

This method can be used only on fields that are active. For details, read "ActivateField Method" on page 183. 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 the Siebel Repository (not in the status line of the application or the list column header), 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.

To set a field to null, follow this example:

```vbscript
SetFieldValue 'Name', ""
```

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 the Pick method to emulate the user picking the record.

**NOTE:** SetFieldValue cannot be used with calculated fields and cannot be used recursively.

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

**Example**
The following example is in Siebel VB:

```vbscript
If Val(Me.GetFieldValue("Rep %")) < 75 Then
    Me.SetFieldValue "Rep %", "75"
    Me.WriteRecord
End If
```

The following is the equivalent example in Siebel eScript:

```vbscript
if (ToInteger(this.GetFieldValue("Rep %")) < 75)
{
    this.SetFieldValue("Rep %", "75");
    this.WriteRecord();
}
```

**Related Topics**
“ActivateField Method” on page 183
“SetFormattedFieldValue Method”
“Pick Method” on page 229
“GetPicklistBusComp Method” on page 209
SetFormattedFieldValue Method

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

```
BusComp.SetFormattedFieldValue FieldName, FieldValue
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>String containing the name of the field to assign the value to.</td>
</tr>
<tr>
<td>FieldValue</td>
<td>String containing the value to assign.</td>
</tr>
</tbody>
</table>

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, COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

Example

This Siebel VB example is a fragment from a program designed to track the velocity of an opportunity through its sales stages:

```vbnet
Function BusComp_PreWriteRecord As Integer
    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 OpportunityBO = TheApplication.GetBusObject ("Opportunity")
    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
```

If (.FirstRecord) Then
    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 NewAfter
        ' 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

Set SalesStageBC = Nothing
Set SalesStageBO = Nothing
Set StageBC = Nothing
Set OpportunityBO = Nothing

End Function

Related Topics
"ActivateField Method" on page 183
"SetFieldValue Method" on page 235

SetMultipleFieldValues Method

SetMultipleFieldValues assigns a new value to the fields specified in the property set for the current row of the business component.
Syntax

\`BusComp.SetMultipleFieldValues oPropertySet\`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oPropertySet</td>
<td>Property set containing a collection of properties representing the fields to be set, and their values</td>
</tr>
</tbody>
</table>

Returns
Not applicable

Usage
This method can be used only on fields that are active. The FieldName argument 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".

**NOTE:** Do not use the SetMultipleFieldValues method on a field that has a pick list.

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

Examples
The following example is in Siebel eScript:

```javascript
var bo = TheApplication().GetBusObject("Opportunity");
var bc = bo.GetBusComp("Opportunity");
var ps = TheApplication().NewPropertySet();

with (ps)
{
    SetProperty ("Name", "Call Center Opportunity");
    SetProperty ("Account", "Marriott International");
    SetProperty ("Sales Stage", "2-Qualified");
}

bc.ActivateMultipleFields(ps);
bc.NewRecord(NewBefore);
bc.SetMultipleFieldValues(ps);
bc.WriteRecord;

ps = null;
bc = null;
bo = null;
```

The following Java Data Bean example sets multiple fields using SetMultipleFieldValues:
try {
    Sieb_dataBean = new SiebelDataBean();
    ...
    Sieb_busObject = Sieb_dataBean.getBusObject("Account");
    Sieb_busComp = Sieb_busObject.getBusComp("Account");
    ps = Sieb_dataBean.newPropertySet();
    with(ps) {
        setProperty("Name", "Frank Williams Inc");
        setProperty("Location", "10 Main St");
        setProperty("Account Status", "Active");
        setProperty("Type", "Customer");
    }
    Sieb_busComp.activateField("Name");
    Sieb_busComp.activateField("Location");
    Sieb_busComp.activateField("Account Status");
    Sieb_busComp.activateField("Type");
    Sieb_busComp.newRecord(true);
    Sieb_busComp.setMultipleFieldValues(ps);
    Sieb_busComp.writeRecord();
}

catch (SiebelException e) {
    System.out.println("Error : " + e.getErrorMessage());
}

ps.release();
Sieb_busComp.release();
Sieb_busObject.release();
Sieb_dataBean.release();

Related Topics
"ActivateMultipleFields Method" on page 184
"GetMultipleFieldValues Method" on page 206

SetNamedSearch Method

SetNamedSearch sets a named search specification on the business component. A named search specification is identified by the searchName argument.
Syntax

*BusComp.SetNamedSearch searchName, searchSpec*

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchName</td>
<td>String containing the name of the named search specification</td>
</tr>
<tr>
<td>searchSpec</td>
<td>String containing the search specification string corresponding to the name</td>
</tr>
</tbody>
</table>

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. When 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 Method*" and "*SetSearchSpec Method*" on page 244.

Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

Examples

This example shows how to set a named search to a business component depending on the position of the current user.

The following example is in Siebel eScript:

```javascript
function BusComp_PreQuery ()
{
  if (TheApplication().GetProfileAttr("Position") == "Siebel Administrator"){
    this.SetNamedSearch ("Candidates", "[Status] LIKE 'Candidate'
  }
  return (ContinueOperation);
}
```

The following example is in Siebel VB:
Function BusComp_PreQuery () As Integer
    If TheApplication.GetProfileAttr("Position") = "Siebel Administrator" Then
        Me.SetNamedSearch "Candidates", 
        
    End If

    BusComp_PreQuery = ContinueOperation
    End Function

Note that defining searches using the SetNamedSearch method does not create a PDQ entry, this is a search specified in script only. To retrieve this search specification, use GetNamedSearch method. GetProfileAttr is used in personalization to retrieve values of an attribute in a user profile.

Related Topics
“GetNamedSearch Method” on page 208
“SetSearchSpec Method” on page 244

SetSearchExpr Method

SetSearchExpr sets an entire search expression on the business component, rather than setting one search specification for each field. Syntax is similar to that on the Predefined Queries screen.

Syntax
BusComp.SetSearchExpr searchSpec

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchSpec</td>
<td>Search specification string field</td>
</tr>
</tbody>
</table>

Returns
Not applicable

Usage
Call this method after ClearToQuery and before ExecuteQuery. It is not necessary to use ActivateField on a field that is used in SetSearchExpr.

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, you would use the following VB syntax to avoid an SQL error:

```vbnet
substatus = GetFieldValue("Sub-Status")
searchst = "[Value] = " & substatus & ""
BC.SetSearchExpr searchst
```

The following VB syntax generates an SQL error:

```vbnet
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.

Any dates used with SetSearchExpr must use the format MM/DD/YYYY, regardless of the Regional control panel settings of the server or client computer.

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

**Example**
The following example in Siebel eScript demonstrates how to log the current search specification to a file for debugging purposes:

```javascript
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(ForwardBackward);
```

**Related Topics**
"ClearToQuery Method" on page 189
"ExecuteQuery Method" on page 193
"SetSearchSpec Method" on page 244
"SetSortSpec Method" on page 248
SetSearchSpec Method

SetSearchSpec sets the search specification for a particular field. This method must be called before ExecuteQuery.

Syntax

```
BusComp.SetSearchSpec FieldName, searchSpec
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>String containing the name of the field on which to set the search specification.</td>
</tr>
<tr>
<td>searchSpec</td>
<td>String containing the search specification.</td>
</tr>
</tbody>
</table>

Returns

Not applicable

Usage

To avoid an unpredictable compound search specification on a business component, it is recommended to call ClearToQuery before calling SetSearchSpec. It is not necessary to use ActivateField on a field that is used in SetSearchSpec.

If multiple calls are made to SetSearchSpec for a business component, then the multiple search specifications are handled as follows:

- If the existing search specification is on the same field as the new search specification, then the new search specification replaces the existing search specification. For example:

  ```
  myBusComp.SetSearchSpec("Status", '<> 'Renewal');
  myBusComp.SetSearchSpec("Status", '<> 'Dropped');
  ```

  results in the following WHERE clause:

  ```
  WHERE Status <> 'Dropped'
  ```

- If the existing search specification is not on the same field as the new search specification, then the resultant search specification is a logical AND of the existing and the new search specifications. For example:

  ```
  myBusComp.SetSearchSpec("Type", '<> 'Renewal');
  myBusComp.SetSearchSpec("Status", '<> 'Sold' AND [Status] <> 'Cancelled' AND [Status] <> 'Renewed');
  ```

  results in the following WHERE clause:

  ```
  WHERE Type <> 'Renewal' AND [Status] <> 'Sold' AND Status <> 'Cancelled' AND Status <> 'Renewed'
  ```

- If the existing search specification includes one or more of the same fields as the new search specification, then the new search specification on those common fields only replaces the existing search specification on the common fields. For example, if:
myBusComp.SetSearchSpec("Status", "<> 'In Progress'")

is subsequently applied to the result of the previous example, then the following WHERE clause results:

WHERE Type <> 'Renewal' AND Status <> 'In Progress'

Only the search specification on Status is replaced in the compound WHERE clause.

■ If a search specification is set declaratively in Siebel Tools, and another search specification is set with script using SetSearchSpec(), then the resultant search specification is a logical AND of the existing Tools-created specification and the scripted specification. For example:

myBusComp.SetSearchSpec("Status", "<> 'Cancelled'")

is applied to the following existing search specification created declaratively in Tools:

[Type] <> 'Renewal' AND [Status] <> 'Sold'

Then the following WHERE clause results:

WHERE Type <> 'Renewal' AND (Status <> 'Sold' AND Status <> 'Cancelled')

**NOTE:** When an existing Tools-created search specification includes the same field as a subsequent search specification set with SetSearchSpec(), the behavior is not like the replacement behavior that results when both specifications are set by using SetSearchSpec().

The maximum length of a predefined query is 2000 characters.

**CAUTION:** Do not use SetSearchExpr and SetSearchSpec together because 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. For example, in VB:

```vbnet
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 VB 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 in VB 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", ""

Any dates used with SetSearchSpec must use the format MM/DD/YYYY, regardless of the Regional
control panel settings of the server or client computer.

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

**Example**
For Siebel VB examples, read “FirstRecord Method” on page 196, “SetFormattedFieldValue Method” on
page 237, and ”BusComp_PreQuery Event” on page 266. For a Siebel eScript example, read
“ClearToQuery Method” on page 189.

**Example**
This Siebel VB 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 = Me.BusComp
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.ClearToQuery
NewComp.SetSearchSpec "Last Name", RecId1
NewComp.SetSearchSpec "First Name", RecId2
NewComp.SetSearchSpec "Account Id", RecId3
NewComp.ExecuteQuery ForwardBackward
TheApplication.GotoView TargetView , NewBusObj
Set NewComp = Nothing
Set NewBusObj = Nothing
Set theCurrComp = Nothing
End Sub

The following example is in Siebel eScript:

var oAccntBO = TheApplication().GetBusObject("Account");
var oAccntBC = oAccntBO.GetBusComp("Account");
var oAddrBC;

with (oAccntBC)
{
    SetViewMode(SalesRepView);
    ClearToQuery();
    SetSearchSpec("Name", "Hong Kong Flower Shop");
    ExecuteQuery(ForwardBackward);
    if (FirstRecord())
```java
oAddrBC = GetMVGBusComp("Street Address");
with (oAddrBC)
{
    NewRecord(NewAfter);
    SetFieldValue("City", "Denver");
    SetFieldValue("Street Address", "123 Main Street");
    WriteRecord();
}
```

```java
oAddrBC = null;
oAccntBC = null;
oAccntBO = null;
```

**Related Topics**
- “ExecuteQuery Method” on page 193
- “ClearToQuery Method” on page 189
- “SetSearchExpr Method” on page 242
- “SetSortSpec Method”

**SetSortSpec Method**

SetSortSpec sets the sort specification for a query.

**Syntax**

```java
BusComp.SetSortSpec sortSpec
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sortSpec</td>
<td>String containing the sort specification</td>
</tr>
</tbody>
</table>

**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)'
```

Any dates used with SetSortSpec must use the format MM/DD/YYYY, regardless of the Regional control panel settings of the server or client computer.

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

**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 ForwardBackward
    End With
End Function
```

The following is the equivalent example in Siebel eScript:

```
function BusComp_PreQuery {
    with (this)
    {
        ActivateField("Account");
        ActivateField("Account Location");
        ClearToQuery();
        SetSortSpec("Account(DESCENDING), Account Location(ASCENDING)");
        ExecuteQuery(ForwardBackward);
    }
}
```
SetUserProperty Method

Sets the value of a named business component user property. The user properties are similar to instance variables of a BusComp.

Syntax

`BusComp.SetUserProperty propertyName, newValue`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>propertyName</code></td>
<td>String containing the name of the user property to set</td>
</tr>
<tr>
<td><code>newValue</code></td>
<td>String containing the property value</td>
</tr>
</tbody>
</table>

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

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

Example

The following example is in Siebel VB:

```vbnet
Sub BusComp_SetFieldValue (FieldName As String)
  Select Case FieldName
  Case "Committed"
    me.SetUserProperty "Flagged", "Y"
  End Select
End Sub
```
The following is the equivalent example in Siebel eScript:

```javascript
function BusComp_SetFieldValue (FieldName)
{
    switch (FieldName)
    {
        case "Committed":
            this.SetUserProperty("Flagged", 'Y');
            break;
    }
}
```

**Related Topic**

"GetUserProperty Method" on page 213

### SetViewMode Method

SetViewMode sets the visibility type for the business component. This is used prior to a query.

**Syntax**

```javascript
BusComp.SetViewMode mode
```

where `mode` is a Siebel ViewMode constant or its corresponding integer value. The constants shown are defined in three environments.

**NOTE:** It is recommended that you use constants, rather than integer values, because integer values are subject to change.
Siebel ViewMode constants correspond to applet visibility types. For more information about applet visibility types, see *Siebel Security Guide*.

<table>
<thead>
<tr>
<th>Siebel ViewMode Constant</th>
<th>Integer Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SalesRepView</td>
<td>0</td>
<td>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.</td>
</tr>
<tr>
<td>ManagerView</td>
<td>1</td>
<td>Displays records that the user and the user’s direct reports have access to. Example: My Team’s Accounts. Typically used by managers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If a user’s position has no subordinate positions, then no data is displayed, not even the user’s own data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To use this visibility applet type, the business component must have a view mode with an Owner Type of Position.</td>
</tr>
<tr>
<td>PersonalView</td>
<td>2</td>
<td>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.</td>
</tr>
<tr>
<td>AllView</td>
<td>3</td>
<td>Displays all records for which there is a valid owner. Example: All Accounts Across Organizations.</td>
</tr>
<tr>
<td>OrganizationView</td>
<td>5</td>
<td>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 and the user’s position is associated with the organization. Example: All Accounts List View.</td>
</tr>
</tbody>
</table>
### Siebel ViewMode Constants

<table>
<thead>
<tr>
<th>Siebel ViewMode Constant</th>
<th>Integer Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>GroupView</td>
<td>7</td>
<td>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.</td>
</tr>
<tr>
<td>CatalogView</td>
<td>8</td>
<td>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.</td>
</tr>
<tr>
<td>SubOrganizationView</td>
<td>9</td>
<td>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.</td>
</tr>
</tbody>
</table>

### Returns

Not applicable

### Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

### Related Topic

“GetViewMode Method” on page 214

### Example

The following example is in Siebel VB. For another example, see “BusComp_PreDeleteRecord Event” on page 263.
```vbscript
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
  SetSearchSpec "Name", Me.GetFieldValue("Name")
  SetSearchSpec "Id", "<> " & Me.GetFieldValue("Id")
  ExecuteQuery ForwardOnly
  If .FirstRecord Then
    TheApplication.Trace"Entry for name " & Me.GetFieldValue("Name") & " exists."
  End If
End With

Set oBC = Nothing
Set oBO = Nothing
```

The following is the equivalent example in Siebel eScript:

```javascript
var oBO = TheApplication().GetBusObject(this.BusObject().Name());
var oBC = oBO.GetBusComp(this.Name);

TheApplication().TraceOn("c:\\trace.txt","Allocation","All");
with (oBC)
{
  SetViewMode(SalesRepView);
  ClearToQuery();
  SetSearchSpec("Name", this.GetFieldValue("Name");
  SetSearchSpec("Id", "<> " + this.GetFieldValue("Id");
  ExecuteQuery(ForwardOnly);
  if (FirstRecord())
    TheApplication().Trace("Entry for name " + this.GetFieldValue("Name") + " exists."");
}

TheApplication().TraceOff();
oBC = null;
oBO = null;
```

### UndoRecord Method

**UndoRecord** reverses any uncommitted changes made to the record. This includes reversing uncommitted modifications to fields, as well as deleting an active record that has not yet been committed to the database.
Syntax

`BusComp.UndoRecord`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

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

COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

Related Topic

"NewRecord Method" on page 225

WriteRecord Method

Commits to the database any changes made to the current record.

Syntax

`oBusComp.WriteRecord`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

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, COM Data Control, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

**Example**
This Siebel VB example inserts an activity if the Sales Stage field is set to 02. For other examples, see “GetMVGBusComp Method” on page 207 and “NewRecord Method” on page 225.

```vbnet
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
                    .SetField Value “Type”, “Event”
                    .SetField Value “Description”, “THRU SVB, Stage changed to 02”
                    .SetField Value “Done”, Format(Now(), “mm/dd/yyyy hh:mm:ss”)
                    .SetField Value “Status”, “Done”
                    .WriteRecord
                End With
            End If
    End Select
End Sub
```

**Business Component Events**
The following topics describe business component events:

- “BusComp_Associate Event” on page 257
- “BusComp_ChangeRecord Event” on page 258
- “BusComp_CopyRecord Event” on page 259
BusComp_Associate Event

The Associate event is called after a record is added to a business component to create an association.

Syntax
BusComp_Associate

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns
Not applicable

Usage
The semantics are the same as for BusComp_NewRecord.

Used With
Server Script
Related Topic
"BusComp_NewRecord Event" on page 261

BusComp_ChangeRecord Event

The ChangeRecord event is called after a record becomes the current row in the business component.

Syntax
BusComp_ChangeRecord

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

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.

```vbnet
(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 Event

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>
**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 Event**

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**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.

**NOTE:** The BusComp_PreDeleteRecord and BusComp_DeleteRecord events do not fire for child records that are deleted due to the Cascade Delete property on a link. Such deletes happen directly from the data layer for performance reasons, while script events are triggered from the object layer and are therefore not executed.

**Returns**
Not applicable

**Used With**
Server Script
**BusComp_InvokeMethod Event**

The InvokeMethod event is called when the InvokeMethod method is called on a business component.

**Syntax**

BusComp_InvokeMethod(methodName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>methodName</td>
<td>String containing the name of the method that was invoked</td>
</tr>
</tbody>
</table>

**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 Event**

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**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 Method” on page 229.

### BusComp_PreAssociate Event

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**

```plaintext
BusComp_PreAssociate
```

**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 Event

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**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 Event**

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**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.

**NOTE:** The BusComp_PreDeleteRecord and BusComp_DeleteRecord events do not fire for child records that are deleted due to the Cascade Delete property on a link. Such deletes happen directly from the data layer, mainly for performance reasons, while script events are triggered from the object layer and are therefore not executed.

**Used With**
Server Script

**Example**
This Siebel VB example prevents the deletion of an account that has associated opportunities:

```vbnet
Option Explicit

Function BusComp_PreDeleteRecord As Integer
    Dim oBC as BusComp
    Dim oBO as BusObject
    Dim sAcctRowId as string

    sAcctRowId = me.GetFieldValue("Id")
    set oBO = TheApplication.GetBusObject("Opportunity")
    set oBC = oBO.GetBusComp("Opportunity")

    With oBC
        .SetViewMode AllView
        .ClearToQuery
        .SetSearchSpec "Account Id", sAcctRowId
        .ExecuteQuery ForwardOnly
        If (.FirstRecord = 1) Then
            RaiseErrorText("Opportunities exist for the Account - Delete is not allowed")
        End If
    End With

    BusComp_PreDeleteRecord = ContinueOperation

    Set oBC = Nothing
    Set oBO = Nothing
End Function
```

**BusComp_PreGetFieldValue Event**
The PreGetFieldValue event is called when the value of a business component field is accessed.
Syntax
BusComp_PreGetFieldValue(FieldName, FieldValue)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>String containing the name of the field accessed</td>
</tr>
<tr>
<td>FieldValue</td>
<td>String containing the value accessed</td>
</tr>
</tbody>
</table>

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.

Even empty scripts are invoked by the framework and thus cause a performance impact. To remove an existing script from BusComp_PreInvokeMethod to improve performance, you must open the script in Siebel Tools and delete its entire contents, including the `Function` and `End Function` (Siebel VB) or `function { } and } (Siebel eScript) statements.

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 Event**

The PreInvokeMethod event is called before a specialized method is invoked on the business component.

Syntax
BusComp_PreInvokeMethod(methodName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>methodName</td>
<td>String containing the name of the method invoked</td>
</tr>
</tbody>
</table>
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 Event
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

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 Event
The PreQuery event is called before query execution.
Syntax
BusComp_PreQuery

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

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
The following example is in Siebel VB:

```vb
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 Event

The PreSetFieldValue event is called after a value is changed in the user interface (when the user attempts to leave the field) or through a call to SetFieldValue, but before any field-level validation is performed. This event is intended to allow the developer to introduce complex validation before any repository validation is applied.
Syntax
BusComp_PreSetFieldValue(FieldName, FieldValue)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>String containing the name of the changed field</td>
</tr>
<tr>
<td>FieldValue</td>
<td>String containing the changed value</td>
</tr>
</tbody>
</table>

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, you have an applet that you use to populate Last Name, First Name, and Contact ID. Therefore, PreSetFieldValue fires 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, PreSetFieldValue still fires for the other two fields populated by the picklist.

NOTE: To prevent infinite recursions, if the PreSetFieldValue event is running it does not run again for the same business component instance, even if used on a different field in the business component.

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 Method” on page 150, and “ExecuteQuery Method” on page 193.

```vbnet
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

The following is the equivalent example in Siebel eScript:

```javascript
function BusComp_PreSetFieldValue (FieldName, FieldValue)
{
    var msgtext = "Discounts greater than 20% must be approved";
    if (FieldName == "Discount")
    {
        if (FieldValue > 20)
        {
            TheApplication().RaiseErrorText(msgtext);
        }
        else
        {
            return (ContinueOperation);
        }
    }
    else
    {
        return (ContinueOperation);
    }
}
```

### BusComp_PreWriteRecord Event

The PreWriteRecord event is called before a row is written to the database. The event can perform any final validation necessary before any internal record-level validation is performed.

#### Syntax

```javascript
BusComp_PreWriteRecord
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

#### 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, or when a record is deleted. When a record is deleted, **PreWriteRecord** is called to delete the implied join records to the initial record.

When associating a multivalue group record (based on an M:M relationship) with the business component that invokes the association, the **PreWriteRecord** and **WriteRecord** events execute. These events execute even if no fields on the base or invoking business component are updated by the association. The **PreWriteRecord** and **WriteRecord** events are executed to acknowledge the update to the intersection table.

**CAUTION:** Be careful when using the **Raise Error** and **RaiseErrorText** methods in **BusComp_PreWriteRecord**, because they cancel operations. For example, if **RaiseErrorText** is used in **BusComp_PreWriteRecord**, the user or code will not be able to step off the current record until the condition causing the **RaiseErrorText** method to be invoked is addressed.

**Used With**
Server Script

**Example**
```vbnet
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** Event

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**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

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.

```vbs
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 Event

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)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>String containing the name of the affected field</td>
</tr>
</tbody>
</table>

Returns

Not applicable

Used With

Server Script

Example

This Siebel VB example shows how to invoke methods on an existing business component when the SetFieldValue event is triggered:

```vbnet
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
```

The following is the equivalent example in Siebel eScript:

```javascript
function BusComp_SetFieldValue (FieldName)
{
    if (FieldName == "Type" && GetFieldValue(FieldName) == "Account")
    {
        SetFieldValue("Description", "Record is of Type 'Account'." );
    }
}
```

BusComp_WriteRecord Event

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 in a PreWriteRecord event (explained in "BusComp_PreWriteRecord Event" on page 269).

**Syntax**

`BusComp_WriteRecord`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

Not applicable

**Usage**

When associating a multi-value group record (based on an M:M relationship) with the business component that invokes the association, the PreWriteRecord and WriteRecord events execute. These events execute even if no fields on the base or invoking business component are updated by the association. The PreWriteRecord and WriteRecord events are executed to acknowledge the update to the intersection table.

**CAUTION:** Be careful when using the Raise Error and RaiseErrorText methods in `BusComp_WriteRecord`, because they cancel operations. For example, if RaiseErrorText is used in `BusComp_WriteRecord`, the user or code will not be able to step off the current record until the condition causing the RaiseErrorText method to be invoked is addressed.

**Used With**

Server Script

## Business Object Methods

In the method descriptions, the term `oBusObj` indicates a variable containing a BusObject:

- "GetBusObject Method" on page 131
- "GetLastErrCode Method" on page 275
- "GetLastErrText Method" on page 275
- "Name Method" on page 276
- "Release Method" on page 276

### GetBusComp Method

The GetBusComp method returns the specified business component.
Syntax

`oBusObj.GetBusComp(BusCompName)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BusCompName</code></td>
<td>String containing the desired business component in the business object</td>
</tr>
</tbody>
</table>

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.

If you already have a BusComp but you want to create a new one (without getting any existing ones), use `GetBusObject()` first. This creates a new BusComp() that is not the same as the one already existing (for example in an applet). Then use the new business object to do a `GetBusComp()` to create new business components. If you use the business object that already exists you pick up any child business components that already exist, even if you use `GetBusComp()` to get them.

Use the Nothing (Siebel VB) or null (eScript or Browser Script) keyword 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, COM Data Control, COM Data Server, Java Data Bean, Server Script

Examples

The following examples are in Siebel eScript:

- To access a business component in the UI context:
  ```javascript
  var ActiveBO = TheApplication().ActiveBusObject();
  var ConBC = ActiveBO.GetBusComp("Contact");
  ```

- To access a business component in the non-UI context:
  ```javascript
  var BO = TheApplication().GetBusObject("Account");
  var ConBC = BO.GetBusComp("Contact");
  ```
GetLastErrorErrorCode Method
The GetLastErrorErrorCode method returns the last error code.

Syntax
```
GetLastErrorErrorCode
```

Returns
The last error code as a short integer; 0 indicates no error.

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

Used With
COM Data Control, Mobile Web Client Automation Server

Related Topic
"GetLastErrorText Method” on page 275

GetLastErrorText Method
The GetLastErrorText method returns the last error text.

Syntax
```
GetLastErrorText
```

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**
COM Data Control, Mobile Web Client Automation Server

**Related Topic**
"GetLastErrCode Method" on page 275

**Name Method**
The Name method returns the name of the business object.

**Syntax**
```java
oBusObj.Name
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**
A string containing the business object name

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

**Example**
For an example, read "Name Method" on page 225.

**Release Method**
The Release() method enables the release of the Business Object and its resources on the Siebel Server.
Syntax

`oBusObj.release()`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns

Not applicable

Used With

Java Data Bean

Example

The following example is for Java Data Bean:

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

{
    ...

    // create Siebel Data Bean
    SiebelDataBean Sieb_dataBean = null;
    Sieb_dataBean = new SiebelDataBean();

    // login into Siebel Data Bean
    ...

    // Create Siebel Bus Object.
    // get the Bus Object from SiebelDataBean
    SiebelBusObject busObj = null;
    busObj = Sieb_dataBean.getBusObject("Account");

    ...

    // Use the business Object
    // Release the business object resources

    ...

    busObj.release();
}
```
Business Service Methods

In the method descriptions, the placeholder `oService` represents a business service instance:

- “GetFirstProperty Method”
- “GetNextProperty Method” on page 280
- “GetProperty Method” on page 281
- “InvokeMethod Method” on page 282
- “Name Method” on page 283
- “PropertyExists Method” on page 284
- “Release Method” on page 284
- “RemoveProperty Method” on page 286
- “SetProperty Method” on page 287

GetFirstProperty Method

This method retrieves the name of the first property of a business service.

Syntax

```
oService.GetFirstProperty()
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

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, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script
Examples

This function returns the number of Property Sets that belong to the Business Service given in the parameter.

The following example is in Siebel eScript:

```javascript
function countPropSets(busService)
{
    var propSetName = busService.GetFirstProperty();
    var count = 0;
    while(propSetName != "")
    {
        count++;
        propSetName = busService.GetNextProperty();
    }
    return count;
}
```

The following example is in Java:

```java
public int countPropSets(SiebelService busService)
{
    int count = 0;
    try
    {
        String propSetName = busService.getFirstProperty();
        while(propSetName != "")
        {
            count++;
            propSetName = busService.getNextProperty();
        }
    }
    catch(SiebelException sExcept)
    {
        return 0;
    }
    return count;
}
```

Related Topics

“GetNextProperty Method” on page 280
“GetProperty Method” on page 281
“SetProperty Method” on page 287
GetNextProperty Method

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()\`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns

A string containing the name of the next property of a business service, or an empty 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 an empty string (" ") is returned. When property names have been retrieved, they can be used as arguments to GetProperty to retrieve the property value, or with SetProperty to assign property values.

Used With

Browser Script, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

Examples

This function returns the number of Property Sets that belong to the Business Service given in parameter.

The following example is in Siebel eScript:

```javascript
function countPropSets(busService)
{
    var propSetName = busService.GetFirstProperty();
    var count = 0;
    while (propSetName != "")
    {
        count++;
        propSetName = busService.GetNextProperty();
    }
    return count;
}
```

The following example is in Java:

```java
```
public int countPropSets(SiebelService busService)
{
    int count = 0;
    try
    {
        String propSetName = busService.getFirstProperty();
        while(propSetName != "")
        {
            count ++;
            propSetName = busService.getNextProperty();
        }
    }
    catch(SiebelException sExcept)
    {
        return 0;
    }
    return count;
}

Related Topics
"GetFirstProperty Method" on page 308
"GetProperty Method"
"SetProperty Method" on page 287

GetProperty Method
The GetProperty method returns the value of the property whose name is specified in its argument.

Syntax
oService.GetProperty(propName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>propName</td>
<td>The name of the property whose value is to be returned</td>
</tr>
</tbody>
</table>

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, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

Related Topics
“GetFirstProperty Method” on page 308
“GetNextProperty Method” on page 280
“SetProperty Method” on page 287

InvokeMethod Method
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>methodName</td>
<td>A string representing the name of the method to execute</td>
</tr>
<tr>
<td>InputArguments</td>
<td>A property set containing the arguments required by the method</td>
</tr>
<tr>
<td>OutputArguments</td>
<td>A property set containing the arguments to be returned by the method (passed by reference)</td>
</tr>
</tbody>
</table>

Browser Script Syntax
outputPropSet=Service.InvokeMethod(MethodName, inputPropSet)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>methodName</td>
<td>The name of the method</td>
</tr>
<tr>
<td>inputPropSet</td>
<td>A property set containing the method arguments</td>
</tr>
<tr>
<td>outputPropSet</td>
<td>A property set containing the output arguments of the Property Set</td>
</tr>
</tbody>
</table>

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. Oracle does not support calling specialized methods with InvokeMethod, unless they are listed in this book.

Used With

Related Topics
"Service_InvokeMethod Event" on page 287
"Service_PreInvokeMethod Event” on page 290

Name Method
The Name property contains the name of the service.

Syntax
```
oService.Name
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

Returns
A string containing the service name

Used With

Example
The following example is in Browser Script:
var svc = theApplication().GetService("Data Quality Manager");
theApplication().SWEAlert("The active service is " + svc.Name());

## PropertyExists Method

This method returns a Boolean value indicating whether a specified property exists.

### Syntax

```javascript
oService.PropertyExists(propName)
```

### Returns

In Siebel VB, an integer (0 for false, 1 for true); in other interfaces, a Boolean

### Usage

Because GetProperty returns an empty 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, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

## Release Method

The Release method() enables the release of the Business Service and its resources on the Siebel Server.

### Syntax

```javascript
oBusSvc.release()
```

### Returns

Not applicable

---

284 Siebel Object Interfaces Reference Version 8.0, Rev. B
Used With
Java Data Bean

Example
The following example logs in to a Siebel Server. It then instantiates a business object, a business component, and a business service. Then, it releases them in reverse order.

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

public class JDBReleaseDemo
{
    private SiebelDataBean m_dataBean = null;
    private SiebelBusObject m_busObject = null;
    private SiebelBusComp m_busComp = null;
    private SiebelService m_busServ = null;

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

    public JDBReleaseDemo()
    {
        try
        {

            // instantiate the Siebel Data Bean
            m_dataBean = new SiebelDataBean();

            // login to the servers
            m_dataBean.login("siebel.tcpip.none.none://<gateway>:<port>/<enterprise>/<object manager>","<user id>","<password>");
            System.out.println("Logged in to the Siebel server ");

            // get the business object
            m_busObject = m_dataBean.busObject("Account");

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

            // get the business service
            m_busServ = m_dataBean.getService("Workflow Process Manager");

            // release the business service
            m_busServ.release();
            System.out.println("BS released ");

            // release the business component
            m_busComp.release();
            System.out.println("BC released ");
        }
    }
}
```
// release the business object
m_busObject.release();
System.out.println("BO released ");

// logoff
m_dataBean.logoff();
System.out.println("Logged off the Siebel server ");
}
catch (SiebelException e)
{
    System.out.println(e.getErrorMessage());
}
}

**RemoveProperty Method**

This method removes a property from a business service.

**Syntax**

`oService.RemoveProperty(propName)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>propName</td>
<td>A string indicating the name of the property to be removed</td>
</tr>
</tbody>
</table>

**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, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

**Related Topic**

“PropertyExists Method” on page 284
**SetProperty Method**
This method assigns a value to a property of a business service.

**Syntax**

```javascript
oService.SetProperty(propName, propValue)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>propName</td>
<td>A string indicating the name of the property whose value is to be set</td>
</tr>
<tr>
<td>propValue</td>
<td>A string containing the value to assign to the property indicated by propName</td>
</tr>
</tbody>
</table>

**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, COM Data Server, Java Data Bean, Mobile Web Client Automation Server, Server Script

**Example**
For an example, read “Service_PreInvokeMethod Event” on page 290.

**Related Topic**
“GetProperty Method” on page 281

---

**Business Service Events**
The following topics describe business service events:

- “Service_InvokeMethod Event”
- “Service_PreCanInvokeMethod Event” on page 289
- “Service_PreInvokeMethod Event” on page 290

**Service_InvokeMethod Event**
The InvokeMethod event is called after the InvokeMethod method is called on a business service.
Server Script Syntax
Service_InvokeMethod(MethodName, InputArguments, OutputArguments)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MethodName</td>
<td>A string representing the name of the method to execute</td>
</tr>
<tr>
<td>InputArguments</td>
<td>A property set containing the arguments required by the method</td>
</tr>
<tr>
<td>OutputArguments</td>
<td>A property set containing the arguments to be returned by the method</td>
</tr>
</tbody>
</table>

Browser Script Syntax
OutputArguments = oService.InvokeMethod(methodName, InputArguments)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>methodName</td>
<td>A string representing the name of the method to execute</td>
</tr>
<tr>
<td>InputArguments</td>
<td>A property set containing the arguments required by the method</td>
</tr>
<tr>
<td>OutputArguments</td>
<td>A property set containing the arguments to be returned by the method.</td>
</tr>
</tbody>
</table>

**NOTE:** In Browser Script, output property sets are not supported for this event.

**Returns**
Not applicable

**Usage**
Although this event does not return a value, in Server Script it can add properties to, or alter the values of the properties in, the property set `OutputArguments`. In Browser Script it cannot add to, store, or update the values of the properties in the output property set.

When you invoke business service methods through Browser Script, the business service can 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 can 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 can modify the properties in the `OutputArguments` property set.

**Used With**
Browser Script, Server Script
Example
This eScript example adds custom logic to a standard business service, Credit Card Transaction Service, for handling transactions that are not approved.

```javascript
function Service_InvokeMethod (MethodName, Inputs, Outputs)
if (Outputs.GetProperty("SiebelResponseMessage") != "Approved")
{
    // special handling for failed txns here
}
```

Related Topic
“Service_PreInvokeMethod Event” on page 290

Service_PreCanInvokeMethod Event
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 Script Syntax**
Service_PreCanInvokeMethod(MethodName, &CanInvoke)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MethodName</td>
<td>A string representing the name of the method to be executed</td>
</tr>
<tr>
<td>&amp;CanInvoke</td>
<td>A string representing whether or not the business service method can be invoked. Valid values are TRUE and FALSE.</td>
</tr>
</tbody>
</table>

**Browser Script Syntax**
Service_PreCanInvokeMethod(MethodName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MethodName</td>
<td>A string representing the name of the method to be executed</td>
</tr>
</tbody>
</table>

**Returns**
CancelOperation or ContinueOperation

**Used With**
Browser Script, Server Script
Service_PreInvokeMethod Event

The PreInvokeMethod event is called before a specialized method on the business service is invoked.

Server Script Syntax
Service_PreInvokeMethod(MethodName, InputArguments, OutputArguments)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MethodName</td>
<td>A string representing the name of the method to execute</td>
</tr>
<tr>
<td>InputArguments</td>
<td>A property set containing the arguments required by the method</td>
</tr>
<tr>
<td>OutputArguments</td>
<td>A property set containing the arguments to be returned by the method</td>
</tr>
</tbody>
</table>

Browser Script Syntax
Service_PreInvokeMethod(name, inputPropSet)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>A string representing the name of the method to execute</td>
</tr>
<tr>
<td>inputPropSet</td>
<td>A property set containing the arguments required by the method</td>
</tr>
</tbody>
</table>

NOTE: In Browser Script, output property sets are not supported for this event.

Returns
“ContinueOperation” or “CancelOperation”

Usage
The Server Script version of this event is used for the following:
- Performing business logic
- Setting any outputs in the output property set
- Returning CancelOperation (assuming a custom business service)

The Browser Script version is used for the following:
- User interaction, such as asking for input data
- Setting input properties
- Canceling user operations, for example “Are you sure you want to do this?”

NOTE: The Browser Script version is not intended to perform business logic, and does not return an output property set.
Figure 7 illustrates the differences in how standard and custom business service methods are handled.

### Standard Method

**Business Service Method Invoked**

```javascript
var svc = TheApplication().GetService(Business Svc);
var psInputs = TheApplication().NewPropertySet();
var psOutputs = TheApplication().NewPropertySet();
...
svc.InvokeMethod ("Method", psInputs, psOutputs);
...
```

**Service_PreInvokeMethod Invoked**

```javascript
Service_PreInvokeMethod (MethodName, Inputs, Outputs) {
    switch (MethodName)
    {
        case "Method":
            // custom handling here
            break;
        // other cases
        ...
    }
}
```

**Method Handled By C++ Code**

C++ class knows how to address Method

**Service_InvokeMethod Invoked**

```javascript
Service_InvokeMethod (MethodName, Inputs, Outputs) {
    switch (MethodName)
    {
        case "Method":
            // custom handling here
            break;
        // other cases
        ...
    }
}
```

### Custom Method

**Business Service Method Invoked**

```javascript
var svc = TheApplication().GetService(Business Svc);
var psInputs = TheApplication().NewPropertySet();
var psOutputs = TheApplication().NewPropertySet();
...
svc.InvokeMethod ("Method", psInputs, psOutputs);
...
```

**Service_PreInvokeMethod Invoked**

```javascript
Service_PreInvokeMethod (MethodName, Inputs, Outputs) {
    switch (MethodName)
    {
        case "Method":
            // custom handling here
            return (CancelOperation);
            break;
        // other cases
    }
}
```

**Method Not Handled By C++ Code**

C++ class cannot address Method, as it is not a known method

**Service_InvokeMethod Not Invoked**

```javascript
Service_InvokeMethod (MethodName, Inputs, Outputs) {
    // code here not executed
}
```

Figure 7. Differences in Handling Standard and Custom Business Service Methods

With a standard method, the script can intercept Method in the Service_PreInvokeMethod event and take any necessary custom actions before the C++ code is executed. The C++ code then executes, setting values in the outputs as defined by the service code.
If the C++ code executes successfully, the Service_InvokeMethod event can be used to inspect the outputs, modify them if necessary, or perform other tasks dependent on the successful completion of the C++ code. At that point, the calling function regains control of the script execution.

With a custom method, the script can intercept Method in the Service_PreInvokeMethod event and take any necessary custom actions.

The script must return CancelOperation. CancelOperation tells the Siebel application to cancel the remaining operations associated with the event. If not canceled, the code flow would continue to the C++ code, which does not have the ability to handle the custom method, and would therefore throw an "Unknown method name" error (indicated by X in Figure 7 on page 291).

Because the method invocation is canceled, the Service_InvokeMethod event is not executed (indicated by X in Figure 7 on page 291).

**Used With**
Browser Script, Server Script

**Example**
This Siebel VB example sets properties in a new Shipping Engine business service:

```vbnet
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("Volume"))
        sZone = 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"
        End Select
```
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

Related Topic
"Service_InvokeMethod Event" on page 287

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 Method" on page 294
- "BusComp Method" on page 294
- "GetProperty Method" on page 295
- "GetValue Method" on page 295
- "Name Method" on page 296
- "SetLabelProperty Method" on page 297
- "SetProperty Method" on page 299
- "SetValue Method" on page 300
Applet Method
The Applet method returns the parent applet object for a control.

Syntax
`controlVar.Applet`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

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 Method
The BusComp method returns the corresponding business component for the control.

Syntax
`controlVar.BusComp`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns
The business component associated with the control’s parent applet.

Used With
Browser Script

For an example, read “Name Method” on page 225.
GetProperty Method

The GetProperty method returns the value of the property of a control.

Syntax

controlVar.GetProperty(propName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>propName</td>
<td>The name of the property to be retrieved</td>
</tr>
</tbody>
</table>

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, Read Only, Visible. For more information on these properties, see Table 35 on page 297.

If more than one property is to be retrieved, each must be retrieved in a separate statement.

Used With

Browser Script

Example

This code sample uses GetProperty to return values for FontSize, Background Color, Width, and Height:

```javascript
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 Method

The GetValue method returns the value of the control. The type of the return value depends on the specific control object.
**Syntax**

`controlVar.GetValue`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**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 Method**

The Name method returns the name of the object.

**Syntax**

`controlVar.Name`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

A string containing the object name

**Used With**

Browser Script

**Example**

For an example, read "Name Method" on page 225.
SetLabelProperty Method

The SetLabelProperty method sets visual properties of a label.

Syntax

```plaintext
controlVar.SetLabelProperty(propName, propValue)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>propName</td>
<td>The name of the property to be set, as described in the following table</td>
</tr>
<tr>
<td>propValue</td>
<td>The value to assign to the property, as described in the following table</td>
</tr>
</tbody>
</table>

Returns

Not applicable

Usage

If more than one property is to be set, each must be set in a separate statement.

Table 35 lists the properties that can be set for a label, and the values that can be assigned to them:

Table 35. Label Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BgColor</td>
<td>string</td>
<td>Determines background color for a label; for example, red is “#ff0000”, green is “#00ff00”, and blue is “#0000ff”</td>
</tr>
<tr>
<td>FontColor</td>
<td>string</td>
<td>Determines font color for a label; for example, green is “#00ff00”</td>
</tr>
<tr>
<td>FontType</td>
<td>string</td>
<td>Determines font type for a label; for example, “Times Roman”</td>
</tr>
<tr>
<td>FontSize</td>
<td>string</td>
<td>Determines font size for a label; for example, ”12 pt”</td>
</tr>
<tr>
<td>FontStyle</td>
<td>string</td>
<td>Determines font style for a label; for example, “italic”</td>
</tr>
<tr>
<td>FontWeight</td>
<td>string</td>
<td>Determines font weight for a label. Acceptable values are bold, bolder, lighter, normal, 100, 200, 300, 400 (equivalent to normal), 500, 600, 700 (equivalent to bold), 800, and 900. Default is normal.</td>
</tr>
<tr>
<td>Height</td>
<td>string</td>
<td>Determines height for a label, in pixels; for example, “5”</td>
</tr>
<tr>
<td>Visible</td>
<td>visible or hidden</td>
<td>Determines whether the label is visible. Defaults to repository definition unless explicitly modified by using SetLabelProperty.</td>
</tr>
<tr>
<td>Width</td>
<td>string</td>
<td>Determines width for a label, in pixels; for example, “80”</td>
</tr>
</tbody>
</table>

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”.

Siebel Object Interfaces Reference Version 8.0, Rev. B 297
Example

The following code shows the use of SetLabelProperty:

```javascript
function Applet_PreInvokeMethod (name, inputPropSet){

    switch (name) {

        // Example of changing the font size of the Location label
        case ("fontsize"):
            {
                var ctl = this.FindControl("Location");
                var fSize = prompt("Specify the desired label font size (numeric value only). ");
                ctl.SetLabelProperty("FontSize", fSize);
                return ("Cancel Operation");
            }

        // Example of changing the background color of the Location label
        case ("bgcolor"):
            {
                var ctl = this.FindControl("Location");
                var bgColor = prompt("Specify the background color of the label. Enter a valid six hexadecimal digit RGB value preceded by ").
                ctl.SetLabelProperty("BgColor", bgColor);
                return ("Cancel Operation");
            }

        // Example of changing the font type of the Location label
        case ("fonttype"):
            {
                var ctl = this.FindControl("Location");
                var fontType = prompt("Specify the font type for the label.");
                ctl.SetLabelProperty("FontType", fontType);
                return ("Cancel Operation");
            }

        // Example of changing the font color of the Location label
        case ("fontcolor"):
            {
                var ctl = this.FindControl("Location");
                var fontColor = prompt("Specify the font color of the label. Enter a valid six hexadecimal digit RGB value preceded by ").
                ctl.SetLabelProperty("FontColor", fontColor);
                return ("Cancel Operation");
            }
    }
}```
SetProperty Method

The SetProperty method sets visual properties of a control.

Syntax

controlVar.SetProperty(propName, propValue)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>propName</td>
<td>The name of the property to be set, as described in the following table</td>
</tr>
<tr>
<td>propValue</td>
<td>The value to assign to the property, as described in the following table</td>
</tr>
</tbody>
</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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BgColor</td>
<td>string</td>
<td>Determines background color for a label; for example, red is &quot;#ff0000&quot;, green is &quot;#00ff00&quot;, and blue is &quot;#0000ff&quot;</td>
</tr>
<tr>
<td>Enabled</td>
<td>TRUE or FALSE</td>
<td>Is the button active? (Unless explicitly modified by using SetProperty, default is TRUE.)</td>
</tr>
<tr>
<td>FontColor</td>
<td>string</td>
<td>Determines font color for a label; for example, green is &quot;#00ff00&quot;</td>
</tr>
<tr>
<td>FontType</td>
<td>string</td>
<td>Determines font type for a label; for example, &quot;Times Roman&quot;</td>
</tr>
<tr>
<td>FontSize</td>
<td>string</td>
<td>Determines font size for a label; for example, &quot;12 pt&quot;</td>
</tr>
<tr>
<td>FontStyle</td>
<td>string</td>
<td>Determines font style for a label; for example, &quot;italic&quot;</td>
</tr>
<tr>
<td>Height</td>
<td>string</td>
<td>Determines height for a control, in pixels; for example, &quot;5&quot;</td>
</tr>
</tbody>
</table>
Control Methods

Example
The following code shows the use of SetProperty:

```javascript
objCheckBox.SetProperty("FontColor", "#00ff00");
objCheckBox.SetProperty("FontStyle", "italic");
objCheckBox.SetProperty("FontType", "Verdana");
objCheckBox.SetProperty("FontSize", "14 pt");
objCheckBox.SetProperty("BgColor", "#00f000");
objCheckBox.SetProperty("Width", "100");
objCheckBox.SetProperty("Height", "100");
```

SetValue Method
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.
SetValue does not validate the format of the data. Data validation occurs at the time user commits the record by stepping off the field/record or saving the record. SetValue can also set the value for a read-only control. However, such value is lost when the record is committed. Also, these methods only work on form applets.

**Used With**
Browser Script

**Example**
The following code shows the use of GetValue and SetValue:

```javascript
function Applet_PreInvokeMethod (name, inputPropSet) {
  switch (name) {
    // Example of changing the value of the Abstract control to uppercase
    case ("SR Abstract"): {
      var ctlName = "Abstract";
      var ctl = this.FindControl(ctlName);
      var ctlVal = ctl.GetValue();
      ctl.SetValue(ctlVal.toUpperCase());
      ctl = null;
      return("CancelOperation");
    }
    // Example of changing the value of a checkbox control
    case ("SR Billable"): {
      var ctlName = "Billable Flag";
      var ctl = this.FindControl(ctlName);
      var ctlVal = ctl.GetValue();
      if (ctlVal == "Y")
        ctl.SetValue("N"); // clear the box
      else
        ctl.SetValue("Y"); // check the box
      ctl = null;
      return("CancelOperation");
    }
    // Example of changing the value of a date/time control
    case ("SR Commit time"): {
      var ctlName = "Agent Committed";
      var ctl = this.FindControl(ctlName);
      ctl.SetValue("12/1/2001 1:09:31 AM");
    }
  }
}
```
// format is not validated until user commits the record
ctl = null;
return("CancelOperation");
}
break;
}

Property Set Methods

In the method descriptions, the placeholder oPropSet refers to a variable containing a property set:

- "AddChild Method" on page 303
- "Copy Method" on page 304
- "GetByteValue Method" on page 305
- "GetChild Method" on page 306
- "GetChildCount Method" on page 307
- "GetFirstProperty Method" on page 308
- "GetLastErrCode Method" on page 309
- "GetLastErrText Method" on page 310
- "GetNextProperty Method" on page 310
- "GetProperty Method" on page 311
- "GetPropertyCount Method" on page 312
- "GetType Method" on page 312
- "GetValue Method" on page 313
- "InsertChildAt Method" on page 314
- "PropertyExists Method" on page 314
- "RemoveChild Method" on page 315
- "RemoveProperty Method" on page 316
- "Reset Method" on page 316
- "SetByteValue Method" on page 317
- "SetProperty Method" on page 318
- "SetType Method" on page 319
- "SetValue Method" on page 319
AddChild Method

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)
```

**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.

If a property set is instantiated within script and then added to a parent property set, the child property set is not released when the parent property set is released. This is because a reference to the child property set still exists independently.

**Used With**


**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);
```
Copy Method

This method returns a copy of a property set.

Syntax

\[ oPropSet.Copy() \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

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


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:

```vbnet
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
    Set message = Nothing
    Set Outputs = Nothing
    Set Inputs = Nothing
End Sub

GetByteValue Method
This method returns a byte array if a byte value has been set.

Syntax
oPropSet.getByteValue()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns
A byte array if a byte value has been set, null if a string value has been set.

Used With
Java Data Bean

Example
The following example takes a binary value and outputs binary.
```java
SiebelPropertySet input = new SiebelPropertySet();
SiebelPropertySet output = new SiebelPropertySet();

input.setProperty("ProcessName", "LMS3 Jason");

// XML to send
String str = "<?xml version="1.0" encoding="UTF8"
?><GetCommunicationDataInput><MemberID>20048963</MemberID></
GetCommunicationDataInput>");

// convert string to byte array
byte[] bvalue = new String(str).getBytes();

input.setByteValue(bvalue);
businessService.invokeMethod("RunProcess", input, output);

// Use getByteValue to retreive the value..and pop it in a String..for example
String out2 = new String(output.getByteValue());
System.out.println(out2);
```

**Related Topic**
"SetByteValue Method" on page 317

## GetChild Method

This method returns a specified child property set of a property set.

### Syntax

```
oPropSet.GetChild(index)
```

### Argument Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>index</code></td>
<td>An integer representing the index number of the child property set to be retrieved</td>
</tr>
</tbody>
</table>

**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 all property sets 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.
Used With

NOTE: When using the Web Client Automation Server, the child object retrieved is a copy of the actual object. Any update to the object retrieved will not update the originating object.

Example
This Siebel eScript example sets the Name property of child property sets to the same value:

```javascript
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');
    }
}
```

Related Topics
“AddChild Method” on page 303
“InsertChildAt Method” on page 314

GetChildCount Method
This method returns the number of child property sets attached to a parent property set.

Syntax
`oPropSet.GetChildCount()`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

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.

**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

Example
For an example, read “GetChild Method” on page 306.

### GetFirstProperty Method
This method returns the name of the first property in a property set.

Syntax

```
oPropSet.GetFirstProperty()
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

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
**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.

```javascript
function Service_PreInvokeMethod (MethodName, Inputs, Outputs)
{
    var propName = "";
    var propVal = "";

    propName = Inputs.GetFirstProperty();

    // stay in loop if the property name is not an empty string
    while (propName != "") {
        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);
}
```

**Related Topics**
"GetNextProperty Method"
"GetProperty Method" on page 311

**GetLastErrCode Method**
The GetLastErrCode method returns the most recent error code.

**Syntax**

```javascript
oPropSet.GetLastErrCode
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**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 Web Client Automation Server, Web Client Automation Server

Related Topic
"GetLastErrText Method”

GetLastErrText Method
The GetLastErrText method returns the last error text message.

Syntax
\( oPropSet.\text{GetLastErrText} \)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

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 Web Client Automation Server, Web Client Automation Server

Related Topic
"GetLastErrCode Method” on page 309

GetNextProperty Method
This method returns the next property in a property set.
Syntax

```javascript
oPropSet.GetNextProperty()
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**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 an empty string ("") is returned. When property names have been retrieved, they may be used as arguments to `GetProperty` to retrieve the property value, or with `SetProperty` to assign property values.

**Used With**

**Example**
For an example, read "GetFirstProperty Method" on page 308.

**Related Topics**
"GetFirstProperty Method" on page 308
"GetProperty Method"

---

**GetProperty Method**

This method returns the value of a property when given the property name.

**Syntax**

```javascript
oPropSet.GetProperty(propName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>propName</td>
<td>A string representing the name of a property as returned by <code>GetFirstProperty</code> or <code>GetNextProperty</code></td>
</tr>
</tbody>
</table>

**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**

**Example**
The following fragment of Siebel eScript code receives a set of input properties used with the Shipping Engine service described in “Service_PreInvokeMethod Event” on page 290:

```javascript
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"));
```

**Related Topics**
"GetFirstProperty Method” on page 308
"GetNextProperty Method” on page 310
"SetProperty Method” on page 318

**GetPropertyCount Method**
This method returns the number of properties attached to a property set.

**Syntax**
```javascript
oPropSet.GetPropertyCount
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**
The number of properties stored at the current level in the hierarchy, but not all properties throughout the entire property set hierarchy

**Used With**

**GetType Method**
This method retrieves the data value stored in the type attribute of a property set.
**Syntax**

\( oPropSet.GetType \)

**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**


**Related Topics**

“GetValue Method”

“SetType Method” on page 319

---

**GetValue Method**

This method retrieves the data value stored in the value attribute of a property set.

**Syntax**

\( oPropSet.GetValue \)

**Argument** | **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**

InsertChildAt Method

This method inserts a child property set into a parent property set at a specific location.

Syntax

```
oPropSet.InsertChildAt childObject, index
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>childObject</td>
<td>A property set to be made subsidiary to the property set indicated by oPropSet</td>
</tr>
<tr>
<td>index</td>
<td>An integer representing the position at which childObject is to be inserted</td>
</tr>
</tbody>
</table>

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


Related Topic

“AddChild Method” on page 303

PropertyExists Method

This method returns a Boolean value indicating whether a specified property exists in a property set.
Syntax

`oPropSet.PropertyExists(propName)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>propName</td>
<td>A string representing the name of the property to be found</td>
</tr>
</tbody>
</table>

Returns

In Siebel VB, an integer (0 for false, 1 for true); in other interfaces, a Boolean

Usage

Because `GetProperty` returns an empty string ("") for every nonexistent property, use `PropertyExists()` in an `if` statement to determine whether a specific property has been set.

Used With


Example

For an example, read “GetFirstProperty Method” on page 308.

RemoveChild Method

This method removes a child property set from a parent property set.

Syntax

`oPropSet.RemoveChild index`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>An integer representing the index number of the child property set to be removed</td>
</tr>
</tbody>
</table>

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**

**Example**
The following Siebel VB code fragment removes every child property set of a property set:

```vbscript
Dim i As Integer
for i = 0 to outputs.GetChildCount()
    outputs.RemoveChild(0)
Next i
```

**Related Topics**
“AddChild Method” on page 303
“InsertChildAt Method” on page 314

**RemoveProperty Method**
This method removes a property from a property set.

**Syntax**
`oPropSet.RemoveProperty propName`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>propName</td>
<td>The name of the property to be removed</td>
</tr>
</tbody>
</table>

**Returns**
Not applicable

**Usage**
This method removes the property `propName` from the property set `oPropSet`.

**Used With**

**Reset Method**
This method removes all properties and children from a property set.
**Reset Method**

Syntax

\[ \text{oPropSet.Reset()} \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

Not applicable

**Usage**

This method removes all properties and children from a property set, allowing the property set to be reused with new properties.

**Used With**


**SetByteValue Method**

This method sets the value portion of a property set.

Syntax

\[ \text{oPropSet.setByteValue(value)} \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>The byte array containing the value to be set</td>
</tr>
</tbody>
</table>

**Returns**

Not applicable

**Used With**

Java Data Bean

**Example**

The following example takes a binary value and outputs binary.

```java
SiebelPropertySet input = new SiebelPropertySet();
SiebelPropertySet output = new SiebelPropertySet();
input.setProperty("ProcessName", "LMS3 Jason");
```
// XML to send
String str = "<?xml version="1.0" encoding="UTF8"?>
<GetCommunicationDataInput><MemberID>20048963</MemberID></GetCommunicationDataInput>";

// convert string to byte array
byte [] bvalue = new String(str).getBytes();
input.setByteValue(bvalue);
businessService.invokeMethod("RunProcess", input, output);

// use getByteValue to retrieve the value and put it into a String
String out2 = new String(output.getByteValue());
System.out.println(out2);

Related Topic
"GetByteValue Method" on page 305

SetProperty Method
This method assigns a data value to a property in a property set.

Syntax
oPropSet.SetProperty propName, propValue

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>propName</td>
<td>A string representing the name of a property</td>
</tr>
<tr>
<td>propValue</td>
<td>A string representing the value to be assigned to propName</td>
</tr>
</tbody>
</table>

Returns
Not applicable

Used With

Example
This Siebel VB fragment makes use of the business service "Shipping Engine," which is illustrated in "Service_PreInvokeMethod Event" on page 290:

    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

Related Topic
"GetProperty Method" on page 311

SetType Method
This method assigns a data value to the type attribute of a property set.

Syntax
oPropSet.SetType type

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>A string representing data to be stored in the type attribute</td>
</tr>
</tbody>
</table>

Returns
Not applicable

Usage
Type, like value, is a special storage location for a data value.

Used With

Related Topics
"GetType Method" on page 312
"SetValue Method" on page 319

SetValue Method
This method assigns a data value to the value attribute of a property set.
**Syntax**

`oPropSet.SetValue value`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>A string representing data to be stored in the value attribute</td>
</tr>
</tbody>
</table>

**Returns**

Not applicable

**Usage**

Values, like properties and types, are storage locations for a data value.

**Used With**


**Related Topics**

"GetValue Method" on page 313
"SetProperty Method" on page 318

---

**Miscellaneous Methods**

The following methods do not belong to any other category:

- "GetErrorCode Method" on page 320
- "GetErrorMessage Method" on page 321
- "TheApplication Method" on page 322

**GetErrorCode Method**

This method is used with the Java Data Bean to display numeric error codes.

**Syntax**

`public int getErrorCode()`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>
Returns
A numeric error code

Used With
Java Data Bean

Example
This example for the Siebel Java Data Bean retrieves the first record in the Account business component. If an error occurs during execution, the script displays the error code and error message.

```java
try {
    // Instantiate the Siebel Data Bean
    Sieb_dataBean = new SiebelDataBean();
    String Cstr = "GatewayServer, EntServer, FINSObjMgr";
    Sieb_dataBean.login(Cstr, "SADMIN", "SADMIN");
    SiebelBusObject m_busObject = Sieb_dataBean.getBusObject("Account");
    SiebelBusComp m_busComp = m_busObject.getBusComp("Account");
    m_busComp.activateField("Name");
    m_busComp.executeQuery(true);
    m_busComp.firstRecord();
    Name = m_busComp.getFieldValue("Name");
    System.out.println("Account Name : " + Name);
    m_busComp.release();
    m_busComp = null;
    m_busObject.release();
    m_busObject = null;
    Sieb_dataBean.logoff();
    Sieb_dataBean = null;
} catch (SiebelException e) {
    ErrorText = "Code: " + e.getErrorCode() + "\n" + "Description: " + e.getErrorMessage();
    System.out.println("Error Occurred\n " + ErrorText);
}

Related Topic
"GetErrorMessage Method"

GetErrorMessage Method
This method is used with the Java Data Bean to display error messages.
Syntax
public string getErrorMessage()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns
A string containing an error message

Used With
Java Data Bean

Related Topic
"GetErrorCode Method"

TheApplication Method

The Application 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()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

Returns
Application, an object for use in finding other objects or invoking methods
Usage
For example, when using Siebel eScript to determine whether you are logged in to a server database or local database, use TheApplication.InvokeMethod("GetDataSource").

Used With
Browser Script, Server Script

Example
The following example is in Siebel VB. It retrieves the login name from the application object and creates the Employee business object.

```vbnet
Dim oEmpBusObj as BusObject
Dim sLoginName as String

sLoginName = TheApplication>LoginName
Set oEmpBusObj = TheApplication.GetBusObject("Employee")

...
Set oEmpBusObj = Nothing
```
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. The following topics show to build and test real-time interfaces to Siebel using C++ for integration purposes:

- "Building the Siebel COM Client in C++" on page 325
- "Testing Your Program" on page 329

Building the Siebel COM Client in C++

The following procedure explains how to build the Siebel COM Client in C++.

To build the Siebel COM client in C++

2. Select the MFC AppWizard (exe) project type.
3. In the Project name field, enter Siebel COM, and then click OK.
   The MFC AppWizard starts.
4. Select 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 the following illustration.

9 Choose View > ClassWizard > Automation.

10 Click Add Class > From a type library.

11 Navigate to the SIEBSRVR_ROOT\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.
   c Double-click the SiebelCOMDlg.h file.
      The code window opens.
   d Add the following code highlighted in bold to the SiebelCOMDlg.h file:

```cpp
#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

   // Other code...

```
//Construction
public:
    CSiebelCOMDlg(CWnd* pParent = NULL); //standard constructor
virtual ~CSiebelCOMDlg();

Choose File > Open, and then select the SiebelCOMDlg.cpp file. Add the following code highlighted in bold to the OnInitDialog procedure:

CDialog::OnInitDialog();
...
// 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
...

In the same file, add the following code highlighted in bold to the OnOK 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 correct file
    sApp.LoadObjects(C:\\siebel\\bin\\siebel.cfg", &sErr);
    if(sErr)
    {
        AfxMessageBox("LoadObject failed.");
        return;
    } else
    {
        AfxMessageBox("CFG file loaded.");
    }

    // Log in as SADMIN
    sApp.Login("SADMIN", "SADMIN", &sErr);
    if(sErr)
    {
        AfxMessageBox("Login failed.");
        return;
    } else

```cpp
{  
    AfxMessageBox("Logged into Siebel database.");
}

// Get Account business object  
LPDISPATCH lpdBo;
lpdBo = sApp.GetBusObject("Account", &sErr);
if(sErr)
{  
    AfxMessageBox("GetBusObject failed.");  
    return;
} else
{  
    AfxMessageBox("Account business object retrieved.");
}  
SiebelBusObject Bo(lpdBo);

// Get Account business component  
LPDISPATCH lpdBc;
lpdBc = Bo.GetBusComp("Account", &sErr);
if(sErr)
{  
    AfxMessageBox("GetBusComp failed.");  
    return;
} else
{  
    AfxMessageBox("Account business component retrieved.");
}
SiebelBusComp Bc(lpdBc);

// Get the name of the first account  
if (sErr) return;
Bc.ClearToQuery(&sErr);
if (sErr) return;
Bc.SetSearchSpec("Name", "+", &sErr);
if (sErr) return;
Bc.ExecuteQuery(ForwardOnly, &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);

Bc = null;
lpdBc = null;
Bo = null;
lpdBo = null;
return;

if (CanExit())
    CDiaglog::OnOK();
```
When you have finished creating your program, test it to make sure it works properly.

Testing Your Program

Use the following procedure to test your program.

To test your program

1. Start your Siebel client application using the same CFG file and login arguments 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 business object retrieved.
   - Account business component retrieved.

   The application displays the name of the first account in the All Accounts view.
This chapter provides a quick reference for Siebel COM Data Control methods. It has the following topics:

- “Application Methods for COM Data Control”
- “Business Component Methods for COM Data Control” on page 334
- “Business Object Methods for COM Data Control” on page 338
- “Business Service Methods for COM Data Control” on page 338
- “Property Set Methods for COM Data Control” on page 339

### Application Methods for COM Data Control

Table 36 lists a summary of the application methods’ syntax.

Table 36 does not include methods that are not invoked directly from an Application object instance. For information on methods that are called with InvokeMethod on the Application object, see “InvokeMethod Methods for the Application Object” on page 143.

#### Table 36. Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| Attach Method        | Allows an external application to reconnect to an existing Siebel session.  | Dim application as SiebelDataControl  
|                      |                                                                             | Dim status as Boolean  
|                      |                                                                             | status = application.Attach(sessionID As String)                       |
| CurrencyCode Method  | Returns the three-letter operating currency code.                           | Dim application as SiebelDataControl  
|                      |                                                                             | Dim sCur as String  
|                      |                                                                             | sCur = Application.CurrencyCode                                         |
| Detach Method        | Returns a string containing the Siebel session ID.                          | Dim application as SiebelDataControl  
|                      |                                                                             | Dim sessionID as String  
|                      |                                                                             | sessionID = application.Detach()                                       |
| EnableExceptions Method | Enables/disables native COM error handling.                               | Dim application as SiebelDataControl  
|                      |                                                                             | Dim bEnable as Boolean  
|                      |                                                                             | bEnable = true  
|                      |                                                                             | application.EnableExceptions(bEnable)                                  |
### Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetBusObject Method       | Instantiates and returns a new instance of the business object specified in the argument. | Dim application as SiebelDataControl  
                          Dim busObject as SiebelBusObject  
                          set busObject = application.GetBusObject(busobjName as String) |
| GetLastErrCode Method     | Returns the last error code.                         | Dim application as SiebelDataControl  
                          Dim iErr as Integer  
                          iErr = application.GetLastErrCode |
| GetLastErrText Method     | Returns the last error text message.                  | Dim application as SiebelDataControl  
                          Dim sText as String  
                          sText = application.GetLastErrText |
| GetProfileAttr Method     | Returns the value of an attribute in a user profile.  | Dim application as SiebelDataControl  
                          Dim sText as String  
                          sText = application.GetProfileAttr(profileAttributeName as string) |
| GetService Method         | Instantiates and returns a new instance of the argument-specified service. | Dim application as SiebelDataControl  
                          Dim service as SiebelService  
                          set service = application.GetService(serviceName as String) |
| GetSharedGlobal Method    | Returns the shared user-defined global variables.     | Dim application as SiebelDataControl  
                          Dim sText as string  
                          sText = application.GetSharedGlobal(globalVariableName as string) |
| InvokeMethod Method       | Calls the named specialized method.                   | Dim application as SiebelDataControl  
                          Dim sReturn as String  
                          sReturn = application.InvokeMethod(methodName as String, methodArgs as String or StringArray) |
| Login Method              | Allows external applications to log in to the COM Data Server. | Dim application as SiebelDataControl  
                          Dim sErr as String  
                          sErr = application.Login(connectString as String, userName as String, password as String) |
| LoginId Method            | Returns the login ID of the user who started the Siebel application. | Dim application as SiebelDataControl  
                          Dim sID as String  
                          sID = application.LoginId |
| LoginName Method          | Returns the login name of the user who started the Siebel application. | Dim application as SiebelDataControl  
                          Dim sUser as String  
                          sUser = application>LoginName |
### Table 36. Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| Logoff Method        | Disconnects the client from the server.                                     | Dim SiebApp as SiebelDataControl  
                      Dim boolVal as Boolean  
                      boolVal = SiebApp.LogOff |
| NewPropertySet Method| Constructs and returns a new property set object.                          | Dim application as SiebelDataControl  
                      Dim PropSet as SiebelPropertySet  
                      PropSet = oApplication.NewPropertySet |
| PositionId Method    | Returns the position ID that describes the user’s current position.         | Dim application as SiebelDataControl  
                      Dim sRow as String  
                      sRow = application.PositionId |
| PositionName Method  | Returns the position name of the user’s current position.                   | Dim application as SiebelDataControl  
                      Dim sPosition as String  
                      sPosition = application.PositionName |
| SetPositionId Method | 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 Method| Sets the active position to the position name specified in the argument.   | Dim application as SiebelDataControl  
                      Dim status as Boolean  
                      status = application.SetPositionName(sPosName) |
| SetProfileAttr Method| Used in personalization to assign values to attributes in a user profile.  | Dim application as SiebelDataControl  
                      application.SetProfileAttr(name as String, value as String) |
| SetSharedGlobal Method| Sets a shared user-defined global variable, which may be accessed using    | Dim SiebApp as SiebelDataControl  
                      Dim boolVal as Boolean  
                      boolVal = SiebApp.SetSharedGlobal(varName As String, value As String) |
| Trace Method         | Appends a message to the trace file.                                        | Dim SiebApp as SiebelDataControl  
                      Dim boolVal as Boolean  
                      boolVal = siebApp.TraceOn(msg As String) |
| TraceOff Method      | Turns off the tracing started by the TraceOn method.                        | Dim SiebApp as SiebelDataControl  
                      Dim boolVal as Boolean  
                      boolVal = siebApp.TraceOff |
| TraceOn Method       | Turns on the tracking of allocations and deallocations of Siebel objects,   | Dim SiebApp as SiebelDataControl  
                      Dim boolVal as Boolean  
                      boolVal = siebApp.TraceOn(fileName As String, category As String, src As String) |
## Business Component Methods for COM Data Control

Table 37 lists a summary of the business component methods’ syntax.

Table 37 does not include methods that are not invoked directly from a Business Component object instance. For information on methods that are called with InvokeMethod on the Business Component object, see “InvokeMethod Methods for the Business Component Object” on page 216.

### Table 37. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ActivateField Method</strong></td>
<td>Allows queries to retrieve data for the specified field.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.ActivateField(fieldName as String)</td>
</tr>
<tr>
<td><strong>ActivateMultipleFields Method</strong></td>
<td>Allows queries to retrieve data for the fields specified in the property set.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.ActivateMultipleFields(oPropSet as SiebelPropertySet)</td>
</tr>
<tr>
<td><strong>Associate Method</strong></td>
<td>Creates a new many-to-many relationship for the parent object through an association business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.Associate(whereIndicator as Integer)</td>
</tr>
<tr>
<td><strong>BusObject Method</strong></td>
<td>Returns the business object that contains the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim busObject as SiebelBusObject&lt;br&gt;Set busObject = busComp.BusObject</td>
</tr>
<tr>
<td><strong>ClearToQuery Method</strong></td>
<td>Clears the current query and sort specifications on the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.ClearToQuery</td>
</tr>
<tr>
<td><strong>DeactivateFields Method</strong></td>
<td>Deactivates every currently activated field.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.DeactivateFields</td>
</tr>
<tr>
<td><strong>DeleteRecord Method</strong></td>
<td>Removes the current record from the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.DeleteRecord</td>
</tr>
<tr>
<td><strong>ExecuteQuery Method</strong></td>
<td>Retrieves a set of BusComp records.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.ExecuteQuery(cursorMode As Integer) As Boolean</td>
</tr>
<tr>
<td><strong>ExecuteQuery2 Method</strong></td>
<td>Retrieves a set of BusComp records.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.ExecuteQuery2(cursorMode As Integer, ignoreMaxCursorSize As Integer) As Boolean</td>
</tr>
<tr>
<td><strong>FirstRecord Method</strong></td>
<td>Moves to the first record in the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim bIsRecord as Boolean&lt;br&gt;bIsRecord = busComp.FirstRecord</td>
</tr>
</tbody>
</table>
### Table 37. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetFieldValue Method       | 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 Method      | Returns the most recent error code.                                        | Dim errCode As Integer  
                          Dim SiebApp as SiebelDataControl  
                          errCode = SiebApp.GetLastErrorCode                                                                                                     |
| GetLastErrText Method      | Returns the most recent 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  
                          Dim MultiFieldValues as SiebelPropertySet  
                          busComp.GetMultipleFieldValues(oFieldNames as SiebelPropertySet, oFieldValues as SiebelPropertySet)                             |
| GetMVGBusComp Method       | 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 Method      | Returns the argument-named search specification.                          | Dim busComp as SiebelBusComp  
                          Dim sValue as String  
                          sValue = busComp.GetNamedSearch(SearchName as String)                                                                                   |
| GetPicklistBusComp Method  | 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 Method       | Returns the current search expression.                                     | Dim busComp as SiebelBusComp  
                          Dim sExpr as String  
                          sExpr = busComp.GetSearchExpr                                                                                                           |
| GetSearchSpec Method       | 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)                                                                                     |
## Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetUserProperty Method   | Returns the value of a named user property.                                | Dim buscomp as SiebelBusComp
                                Dim retStr as String
                                retStr = buscomp.GetUserProperty(prop As String) As String             |
| GetViewMode Method       | Returns the visibility mode for the business component.                    | Dim busComp as SiebelBusComp
                                Dim iMode as Integer
                                iMode = busComp.GetViewMode                                             |
| InvokeMethod Method      | 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 Method        | Moves to the last record in the business component.                        | Dim busComp as SiebelBusComp
                                Dim bReturn as Boolean
                                bReturn = busComp.LastRecord                                             |
| Name Method              | Returns the name of the business component.                                | Dim busComp as SiebelBusComp
                                Dim sName as String
                                sName = busComp.Name                                                      |
| NewRecord Method         | Adds a new record to the business component.                               | Dim busComp as SiebelBusComp
                                busComp.NewRecord(whereIndicator as Integer)                            |
| NextRecord Method        | Moves to the next record in the business component.                        | Dim busComp as SiebelBusComp
                                Dim bReturn as Boolean
                                bReturn = busComp.NextRecord                                             |
| ParentBusComp Method     | Returns the parent business component.                                     | Dim busComp as SiebelBusComp
                                Dim parentBusComp as SiebelBusComp
                                Set parentBusComp = busComp.ParentBusComp                              |
| Pick Method              | 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 Method    | Moves to the previous record in the business component.                   | Dim busComp as SiebelBusComp
                                Dim bReturn as Boolean
                                bReturn = busComp.PreviousRecord                                         |
| RefineQuery Method       | Refines a query after a query has been executed.                           | Dim busComp as SiebelBusComp
                                busComp.RefineQuery                                                       |
### Table 37. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| **SetFieldValue Method**    | 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)` |
| **SetFormattedFieldValue Method** | 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 Method** | 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 Method**   | Sets a named search specification on the business component. | `Dim busComp as SiebelBusComp
busComp.SetNameSearch(searchName as String, searchSpec as String)` |
| **SetSearchExpr Method**    | Sets the search specification for the business component. | `Dim busComp as SiebelBusComp
busComp.SetSearchExpr(searchSpec as String)` |
| **SetSearchSpec Method**    | Sets the search specification for the specified field. | `Dim busComp as SiebelBusComp
busComp.SetSearchSpec(FieldName as String, searchSpec as String)` |
| **SetSortSpec Method**      | Sets the sort specification for a query. | `Dim busComp as SiebelBusComp
busComp.SetSortSpec(sortSpec as String)` |
| **SetUserProperty Method**  | Sets the value of a named user property. | `Dim busComp as SiebelBusComp
busComp.SetUserProperty(property-Name as String, newValue as String)` |
| **SetViewMode Method**      | Sets the visibility type for the business component. | `Dim busComp as SiebelBusComp
Dim boolVal as Boolean
boolVal = busComp.SetViewMode(mode As Integer)` |
| **UndoRecord Method**       | Reverses any uncommitted changes made to the record. | `Dim busComp as SiebelBusComp
busComp.UndoRecord` |
| **WriteRecord Method**      | Commits to the database any changes made to the current record. | `Dim busComp as SiebelBusComp
busComp.WriteRecord` |
Business Object Methods for COM Data Control

Table 38 lists a summary of the business object methods’ syntax.

Table 38. Business Object Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetBusComp Method</td>
<td>Returns the specified business component.</td>
<td>Dim busObject as SiebelBusObject&lt;br&gt;Dim busComp as SiebelBusComp&lt;br&gt;Set busComp = BusObject.GetBusComp(BusCompName as String)</td>
</tr>
<tr>
<td>GetLastErrCode Method</td>
<td>Returns the most recent error code.</td>
<td>Dim busObject as SiebelBusObject&lt;br&gt;IErr as Integer&lt;br&gt;IErr = busObject.GetLastErrCode</td>
</tr>
<tr>
<td>GetLastErrText Method</td>
<td>Returns the most recent error text.</td>
<td>Dim busObject as SiebelBusObject&lt;br&gt;Dim sErr as String&lt;br&gt;sErr = busObject.GetLastErrText</td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the control.</td>
<td>Dim busObject as SiebelBusObject&lt;br&gt;Dim sName as String&lt;br&gt;sName = busObject.Name</td>
</tr>
</tbody>
</table>

Business Service Methods for COM Data Control

Table 39 lists a summary of the business service methods’ syntax.

Table 39. Business Service Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetFirstProperty Method</td>
<td>Retrieves the name of the first property of a business service.</td>
<td>Dim oService as SiebelService&lt;br&gt;Dim sName as String&lt;br&gt;sName = oService.GetFirstProperty()</td>
</tr>
<tr>
<td>GetNextProperty Method</td>
<td>After the name of the first property has been retrieved, retrieves the name of the next property of a business service.</td>
<td>Dim oService as SiebelService&lt;br&gt;Dim sName as String&lt;br&gt;sName = oService.GetNextProperty()</td>
</tr>
<tr>
<td>GetProperty Method</td>
<td>Retrieves the value stored in the specified property.</td>
<td>Dim oService as SiebelService&lt;br&gt;Dim sValue as String&lt;br&gt;sValue = oService.GetProperty(propName as String)</td>
</tr>
</tbody>
</table>
**Table 39. Business Service Methods Syntax Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| Name Method          | Returns the name of the business service.        | Dim oService as SiebelService  
                     Dim sName as String  
                     sName = oService.Name |
| InvokeMethod Method  | Calls a specialized method or a user-created method on the business service. | Dim oService as SiebelService  
                     oService.InvokeMethod(methodName as String, InputArguments as SiebelPropertySet, OutputArguments as SiebelPropertySet) |
| PropertyExists Method| Returns a Boolean value indicating whether the property specified in the argument exists. | Dim oService as SiebelService  
                     Dim propExists as Boolean  
                     propExists = oService.PropertyExists(propName as String) |
| RemoveProperty Method| Removes a property from a business service.      | Dim oService as SiebelService  
                     oService.RemoveProperty(propName as String) |
| SetProperty Method    | Assigns a value to a property of a business service. | Dim oService as SiebelService  
                     oService.SetProperty(propName as String, propValue as String) |

**Property Set Methods for COM Data Control**

Table 40 lists a summary of the property set methods' syntax.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| AddChild Method| Adds subsidiary property sets to a property set. | Dim oPropSet as SiebelPropertySet  
                     Dim iIndex as Integer  
                     iIndex = oPropSet.AddChild(childObject as Property Set) |
| Copy Method    | Returns a copy of a property set.                | Dim oPropSet1 as SiebelPropertySet  
                     Dim oPropSet2 as SiebelPropertySet  
                     oPropSet2 = oPropSet1.Copy() |
| GetChild Method| Returns a specified child property set of a property set. | Dim oPropSet as SiebelPropertySet  
                     Dim oPropSet1 as SiebelPropertySet  
                     oPropSet1 = oPropSet.GetChild(index as Integer) |
| GetChildCount Method | Returns the number of child property sets attached to a parent property set. | Dim oPropSet as SiebelPropertySet  
                     Dim iCount as Integer  
                     iCount = oPropSet.GetChildCount() |
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetFirstProperty Method</td>
<td>Returns the name of the first property in a property set.</td>
</tr>
<tr>
<td>GetNextProperty Method</td>
<td>Returns the name of the next property in a property set.</td>
</tr>
<tr>
<td>GetProperty Method</td>
<td>Returns the value of a property when given the property name.</td>
</tr>
<tr>
<td>GetPropertyCount Method</td>
<td>Returns the number of properties attached to a property set.</td>
</tr>
<tr>
<td>GetType Method</td>
<td>Returns the value stored in a type in a property set.</td>
</tr>
<tr>
<td>GetValue Method</td>
<td>Returns a value stored as part of a property set.</td>
</tr>
<tr>
<td>InsertChildAt Method</td>
<td>Inserts a child property set into a parent property set at a specific location.</td>
</tr>
<tr>
<td>PropertyExists Method</td>
<td>Returns a Boolean value indicating whether the property specified in the argument exists.</td>
</tr>
<tr>
<td>RemoveChild Method</td>
<td>Removes a child property set as a specified index from a parent property set.</td>
</tr>
<tr>
<td>RemoveProperty Method</td>
<td>Removes the property specified in its argument from a property set.</td>
</tr>
<tr>
<td>Reset Method</td>
<td>Removes every property and child property set from a property set.</td>
</tr>
<tr>
<td>SetProperty Method</td>
<td>Assigns a value to the property of a property set specified in its argument.</td>
</tr>
</tbody>
</table>

**Syntax**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| GetFirstProperty Method | Dim oPropSet as SiebelPropertySet  
Dim sPropName as String  
sPropName = oPropSet.GetFirstProperty() |
| GetNextProperty Method   | Dim oPropSet as SiebelPropertySet  
Dim sPropName as String  
sPropName = oPropSet.GetNextProperty() |
| GetProperty Method      | Dim oPropSet as SiebelPropertySet  
Dim sPropVal as String  
sPropVal = oPropSet.GetProperty(propName as String) |
| GetPropertyCount Method | Dim oPropSet as SiebelPropertySet  
Dim count as Long  
count = oPropSet.GetPropertyCount() |
| GetType Method         | Dim oPropSet as SiebelPropertySet  
Dim sTypeVal as String  
sTypeVal = oPropSet.GetType() |
| GetValue Method        | Dim oPropSet as SiebelPropertySet  
Dim sValVal as String  
sValVal = oPropSet.GetValue() |
| InsertChildAt Method   | Dim oPropSet as SiebelPropertySet  
oPropSet.InsertChildAt(childObject as SiebelPropertySet, index as Long) |
| PropertyExists Method   | Dim oPropSet as Property Set  
Dim propExists as Boolean  
propExists = oPropSet.PropertyExists(propName as String) |
| RemoveChild Method     | Dim oPropSet as SiebelPropertySet  
oPropSet.RemoveChild(index as Long) |
| RemoveProperty Method  | Dim oPropSet as SiebelPropertySet  
oPropSet.RemoveProperty(propName as String) |
| Reset Method           | Dim oPropSet as SiebelPropertySet  
oPropSet.Reset() |
| SetProperty Method      | Dim oPropSet as SiebelPropertySet  
oPropSet.SetProperty(propName as String, propValue as String) |
### Table 40. Property Set Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| SetType Method | Assigns a data value to a type member of a property set. | Dim oPropSet as SiebelPropertySet  
oPropSet.SetType(value as String) |
| SetValue Method| Assigns a data value to a value member of a property set. | Dim oPropSet as SiebelPropertySet  
oPropSet.SetValue(value as String) |
This chapter provides a quick reference for Siebel COM Data Server methods. It has the following topics:

- "Application Methods for COM Data Server"
- "Business Component Methods for COM Data Server” on page 346
- "Business Object Methods for COM Data Server” on page 350
- "Business Service Methods for COM Data Server” on page 351
- "Property Set Methods for COM Data Server” on page 352

### Application Methods for COM Data Server

Table 41 lists a summary of the application methods’ syntax.

Table 41 does not include methods that are not invoked directly from an Application object instance. For information on methods that are called with InvokeMethod on the Application object, see “InvokeMethod Methods for the Application Object” on page 143.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| CurrencyCode     | Returns the three-letter operating currency code.                           | `Dim application as SiebelApplication
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 busObject as SiebelBusObject
set busObject = application.GetBusObject(busobjName as String, ErrCode as Integer)`                                             |
| GetLastErrText   | Returns the last error text message.                                         | `Dim sText as String
sText = application.GetLastErrText(ErrCode as Integer)`                                                                             |
Table 41. Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetProfileAttr Method | Returns the value of an attribute in a user profile. | Dim application as SiebelApplication  
Dim sText as String  
sText = application.GetProfileAttr(Name as String) |
| GetService Method     | 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 Method| Gets the shared user-defined global variables. | Dim application as SiebelApplication  
Dim sName as String  
sName = application.GetSharedGlobal(varName as String, ErrCode as Integer) |
| InvokeMethod Method   | Calls the named specialized method. | Dim application as SiebelApplication  
InvokeMethod(methodName as String, methodArgs as String or StringArray) |
| LoadObjects Method    | Starts the COM Data Server object and returns a reference to the Application object. | Dim application as SiebelApplication  
LoadObjects(pathName\cfgFile Name as String, ErrCode as Integer) |
| Login Method          | Allows external applications to log in to the COM Data Server. | Dim application as SiebelApplication  
application.Login(userName as String, password as String, ErrCode as Integer) |
| LoginId Method        | 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 Method      | 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 Method | Constructs and returns a new property set object. | Dim oApplication as SiebelApplication  
Dim oPropSet as SiebelPropertySet  
oPropSet = oApplication.NewPropertySet() |
| PositionId Method     | 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) |
### Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| **PositionName Method** | Returns the position name of the user’s current position. | `Dim application as SiebelApplication
DimsPosition as String
sPosition = application.PositionName(ErrCode as Integer)` |
| **SetPositionId Method** | Sets the active position to the position ID specified in the argument. Returns a Boolean value indicating if the method succeeded. | `Dim application as SiebelApplication
Dim posId as String
Dim status as Boolean
status = application.SetPositionId(posId as String, ErrCode as Integer)` |
| **SetPositionName Method** | Sets the active position to the position name specified in the argument. Returns a Boolean value indicating if the method succeeded. | `Dim application as SiebelApplication
Dim posName as String
Dim status as Boolean
status = application.SetPositionName(posName as String, ErrCode as Integer)` |
| **SetProfileAttr Method** | Used in personalization to assign values to attributes in a user profile. | `Dim application as SiebelApplication
SetProfileAttr(name as String, value as String, ErrCode as Integer)` |
| **SetSharedGlobal Method** | Sets a shared user-defined global variable. | `Dim application as SiebelApplication
SetSharedGlobal(varName as String, value as String, ErrCode as Integer)` |
| **Trace Method** | Appends a message to the trace file. | `Dim application as SiebelApplication
application.Trace(message as String, ErrCode as Integer)` |
| **TraceOff Method** | Turns off the tracing started by TraceOn. | `Dim application as SiebelApplication
application.TraceOff(ErrCode as Integer)` |
| **TraceOn Method** | Turns tracing on | `Dim application as SiebelApplication
application.TraceOn(filename as String, type as Integer, Selection as String, ErrCode as Integer)` |
## Business Component Methods for COM Data Server

Table 42 lists a summary of the business component methods’ syntax.

**NOTE:** Table 42 does not include methods that are called with `InvokeMethod`. For information on methods that are called with `InvokeMethod` on the Business Component object, see “InvokeMethod Methods for the Business Component Object” on page 216.

**Table 42. Business Component Methods Syntax Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ActivateField Method</code></td>
<td>Allows queries to retrieve data for the specified field.</td>
<td><code>Dim busComp as SiebelBusComp</code>&lt;br&gt;<code>busComp.ActivateField(fieldName as String, ErrCode as Integer)</code></td>
</tr>
<tr>
<td><code>ActivateMultipleFields Method</code></td>
<td>Allows queries to retrieve data for the fields specified in the property set.</td>
<td><code>Dim busComp as SiebelBusComp</code>&lt;br&gt;<code>busComp.ActivateMultipleFields(oPropSet as SiebelPropertySet, ErrCode as Integer)</code></td>
</tr>
<tr>
<td><code>Associate Method</code></td>
<td>Creates a new many-to-many relationship for the parent object through an association business component.</td>
<td><code>Dim busComp as SiebelBusComp</code>&lt;br&gt;<code>busComp.Associate(whereIndicator as Integer, ErrCode as Integer)</code></td>
</tr>
<tr>
<td><code>BusObject Method</code></td>
<td>Returns the business object that contains the business component.</td>
<td><code>Dim busComp as SiebelBusComp</code>&lt;br&gt;<code>Dim busObject as BusObject</code>&lt;br&gt;<code>Set busObject = busComp.BusObject(ErrCode as Integer)</code></td>
</tr>
<tr>
<td><code>ClearToQuery Method</code></td>
<td>Clears the current query and sort specifications on the business component.</td>
<td><code>Dim busComp as SiebelBusComp</code>&lt;br&gt;<code>busComp.ClearToQuery(ErrCode as Integer)</code></td>
</tr>
<tr>
<td><code>DeactivateFields Method</code></td>
<td>Deactivates every currently activated field.</td>
<td><code>Dim busComp as SiebelBusComp</code>&lt;br&gt;<code>busComp.DeactivateFields(ErrCode as Integer)</code></td>
</tr>
<tr>
<td><code>DeleteRecord Method</code></td>
<td>Removes the current record from the business component.</td>
<td><code>Dim busComp as SiebelBusComp</code>&lt;br&gt;<code>busComp.DeleteRecord(ErrCode as Integer)</code></td>
</tr>
<tr>
<td><code>ExecuteQuery Method</code></td>
<td>Retrieves a set of BusComp records.</td>
<td><code>Dim busComp as SiebelBusComp</code>&lt;br&gt;<code>busComp.ExecuteQuery(cursorMode as Boolean, ErrCode as Integer)</code></td>
</tr>
<tr>
<td><code>ExecuteQuery2 Method</code></td>
<td>Retrieves a set of BusComp records.</td>
<td><code>Dim busComp as SiebelBusComp</code>&lt;br&gt;<code>busComp.ExecuteQuery2(cursorMode as Boolean, ignoreMaxCursorSize as Boolean, ErrCode as Integer)</code></td>
</tr>
</tbody>
</table>
### Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| **FirstRecord Method**      | Moves to the first record in the business component.                       | `Dim busComp as SiebelBusComp`  
|                             | `Dim bIsRecord as Boolean`  
|                             | `bIsRecord = busComp.FirstRecord(ErrCode as Integer)`                       |                                                                                                                                          |
| **FirstSelected Method**    | Moves to the first record of the multiple selection in the business component. | `Dim busComp as SiebelBusComp`  
|                             | `Dim iRecord as Integer`  
|                             | `iRecord = busComp.FirstSelected`                                           |                                                                                                                                          |
| **GetAssocBusComp Method**  | Returns the association business component.                                | `Dim busComp as SiebelBusComp`  
|                             | `Dim AssocBusComp as BusComp`  
|                             | `Set AssocBusComp = busComp.GetAssocBusComp(ErrCode as Integer)`            |                                                                                                                                          |
| **GetFieldValue Method**    | 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 Method** | 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 Method** | Returns a value for the fields specified in the property set.              | `Dim busComp as SiebelBusComp`  
|                             | `Dim retValue as Boolean`  
|                             | `retValue = busComp.GetMultipleFieldValues(oPropSetName as SiebelPropertySet, oPropSetValue as SiebelPropertySet, ErrCode as Integer)` |                                                                                                                                          |
| **GetMVGBusComp Method**    | 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 Method**   | Returns the argument-named search specification.                           | `Dim busComp as SiebelBusComp`  
|                             | `Dim sValue as String`  
<p>|                             | <code>sValue = busComp.GetNamedSearch(SearchName as String, ErrCode as Integer)</code> |                                                                                                                                          |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetPicklistBusComp Method     | 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 Method          | Returns the current search expression.                                      | Dim busComp as SiebelBusComp  
Dim sExpr as String  
sExpr = busComp.GetSearchExpr(ErrCode as Integer) |
| GetSearchSpec Method          | 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 Method        | 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) |
| GetViewMode Method            | Returns the visibility mode for the business component.                     | Dim busComp as SiebelBusComp  
Dim iMode as Integer  
iMode = busComp.GetViewMode(ErrCode as Integer) |
| LastRecord Method             | Moves to the last record in the business component.                         | Dim busComp as SiebelBusComp  
Dim bReturn as Boolean  
bReturn = busComp.LastRecord(ErrCode as Integer) |
| Name Method                   | Returns the name of the business component.                                 | Dim busComp as SiebelBusComp  
Dim sName as String  
sName = busComp.Name(ErrCode as Integer) |
| NewRecord Method              | Adds a new record to the business component.                                | Dim busComp as SiebelBusComp  
busComp.NewRecord(whereIndicator as Integer, ErrCode as Integer) |
| NextRecord Method             | Moves to the next record in the business component.                         | Dim busComp as SiebelBusComp  
Dim bReturn as Boolean  
bReturn = busComp.NextRecord(ErrCode as Integer) |
### Table 42. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParentBusComp Method</td>
<td>Returns the parent business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim parentBusComp as SiebelBusComp&lt;br&gt;&lt;br&gt;Set parentBusComp = busComp.ParentBusComp(ErrCode as Integer)</td>
</tr>
<tr>
<td>Pick Method</td>
<td>Places the currently selected record in a picklist business component into the appropriate fields of the parent business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.Pick(ErrCode as Integer)</td>
</tr>
<tr>
<td>PreviousRecord Method</td>
<td>Moves to the previous record in the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim bReturn as Boolean&lt;br&gt;bReturn = busComp.PreviousRecord(ErrCode as Integer)</td>
</tr>
<tr>
<td>RefineQuery Method</td>
<td>Refines a query after a query has been executed.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.RefineQuery(ErrCode as Integer)</td>
</tr>
<tr>
<td>SetFieldValuMethod</td>
<td>Assigns a new value to the named field for the current row of the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;SetFieldValue(fieldName As String, fieldValue As String, ErrCode as Integer)</td>
</tr>
<tr>
<td>SetFormattedFieldValue Method</td>
<td>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.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.SetFormattedFieldValue(fieldName as String, fieldValue as String, ErrCode as Integer)</td>
</tr>
<tr>
<td>SetMultipleFieldValues Method</td>
<td>Assigns a new value to the fields specified in the property set for the current row of the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.SetMultipleFieldValues(propSet as SiebelPropertySet, ErrCode as Integer)</td>
</tr>
<tr>
<td>SetNamedSearch Method</td>
<td>Sets a named search specification on the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.SetNamedSearch(searchName as String, searchSpec as String, ErrCode as Integer)</td>
</tr>
<tr>
<td>SetSearchExpr Method</td>
<td>Sets the search specification for the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.SetSearchExpr(searchSpec as String, ErrCode as Integer)</td>
</tr>
</tbody>
</table>

**Siebel Object Interfaces Reference** Version 8.0, Rev. B
Table 43. Business Object Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| SetSearchSpec Method    | Sets the search specification for the specified field. | Dim busComp as SiebelBusComp  
busComp.SetSearchSpec(FieldName as String, searchSpec as String, ErrCode as Integer) |
| SetSortSpec Method      | Sets the sort specification for a query.          | Dim busComp as SiebelBusComp  
busComp.SetSortSpec(sortSpec as String, ErrCode as Integer) |
| SetUserProperty Method  | Sets the value of the specified User Property.    | Dim busComp as SiebelBusComp  
busComp.SetUserProperty(propertyName as String, newValue as String, ErrCode as Integer) |
| SetViewMode Method      | Sets the visibility type for the business component. | Dim buscomp as SiebelBusComp  
buscomp.SetViewMode(mode As Integer, errCode As Integer) |
| UndoRecord Method       | Reverses any uncommitted changes made to the record. | Dim busComp as SiebelBusComp  
busComp.UndoRecord(ErrCode as Integer) |
| WriteRecord Method      | Commits to the database any changes made to the current record | Dim busComp as SiebelBusComp  
busComp.WriteRecord(ErrCode as Integer) |

**Business Object Methods for COM Data Server**

Table 42 lists a summary of the business component methods’ syntax.

Table 43. Business Object Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetBusComp Method    | Returns the specified business component.        | Dim busObject as SiebelBusObject  
Dim busComp as SiebelBusComp  
set busComp = busObject.GetBusComp(BusCompName as String, ErrCode as Integer) |
| Name Method          | Returns the name of the control.                 | Dim busObject as SiebelBusObject  
Dim sName as String  
sName = busObject.Name(ErrCode as Integer) |
## Business Service Methods for COM Data Server

Table 44 lists a summary of the business service methods’ syntax.

Table 44. Business Service Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetFirstProperty Method | 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 Method | After 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 Method | 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 Method       | Returns the name of the business service.                                    | Dim oService as SiebelService  
Dim sName as String  
sName = oService.Name |
| InvokeMethod Method | Calls a specialized method or a user-created method on the business service. | Dim oService as SiebelService  
InvokeMethod(methodName as String, InputArguments as SiebelPropertySet, OutputArguments as SiebelPropertySet, ErrCode as Integer) |
| PropertyExists Method | Returns a Boolean value indicating whether the property specified in the argument exists. | Dim oService as SiebelService  
Dim propExists as Boolean  
propExists = oService.PropertyExists(propName as String) |
| RemoveProperty Method | Removes a property from a business service.                                 | Dim oService as SiebelService  
RemoveProperty(propName as String, ErrCode as Integer) |
| SetProperty Method | Assigns a value to a property of a business service.                        | Dim oService as SiebelService  
SetProperty(propName as String, propValue as String, ErrCode as Integer) |
# Property Set Methods for COM Data Server

Table 45 lists a summary of the property set methods’ syntax.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| AddChild Method     | Adds subsidiary property sets to a property set. | Dim oPropSet as SiebelPropertySet  
                       | Dim iIndex as Integer  
                       | iIndex = oPropSet.AddChild(childObject as PropertySet, errCode as Integer) |
| Copy Method         | Returns a copy of a property set.                | Dim oPropSet1 as SiebelPropertySet  
                       | Dim oPropSet2 as SiebelPropertySet  
                       | oPropSet2 = oPropSet1.Copy(ErrCode as Integer) |
| GetChild Method     | Returns a specified child property set of a property set. | Dim oPropSet as SiebelPropertySet  
                       | Dim oChildPropSet as SiebelPropertySet  
                       | oChildPropSet = oPropSet.GetChild(index as Integer, ErrCode as Integer) |
| GetChildCount Method| 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) |
| GetFirstProperty Method | 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 Method | 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 Method  | 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 Method | Returns the number of properties contained within the property set. | Dim oPropSet as SiebelPropertySet  
                       | Dim propCount as Integer  
                       | propCount = oPropSet.GetPropertyCount(ErrCode as Integer) |
### Property Set Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| **Get Type Method**  | 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)` |
| **Get Value Method** | Returns a value stored as part of a property set.                            | `Dim oPropSet as SiebelPropertySet`  
`Dim sValVal as String`  
`sValVal = oPropSet.GetValue(ErrCode as Integer)` |
| **Insert Child At Method** | 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)` |
| **Property Exists Method** | Returns a Boolean value indicating whether the property specified in the argument exists. | `Dim oPropSet as SiebelPropertySet`  
`Dim propExists as Boolean`  
`propExists = oPropSet.PropertyExists(propName as String, ErrCode as Integer)` |
| **Remove Child Method** | Removes a child property set as a specified index from a parent property set. | `Dim oPropSet as SiebelPropertySet`  
`oPropSet.RemoveChild(index as Integer, errCode as Integer)` |
| **Remove Property Method** | Removes the property specified in its argument from a property set.          | `Dim oPropSet as SiebelPropertySet`  
`oPropSet.RemoveProperty(propName as String, ErrCode as Integer)` |
| **Reset Method**     | Removes every property and child property set from a property set.           | `Dim oPropSet as SiebelPropertySet`  
`oPropSet.Reset(ErrCode as Integer)` |
| **Set Property Method** | 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)` |
| **Set Type Method**  | Assigns a data value to a type member of a property set.                      | `Dim oPropSet as SiebelPropertySet`  
`oPropSet.SetType(value as String, ErrCode as Integer)` |
| **Set Value Method** | Assigns a data value to a value member of a property set.                    | `Dim oPropSet as SiebelPropertySet`  
`oPropSet.SetValue(value as String, errCode as Integer)` |
This chapter provides a quick reference for Siebel Mobile Web Client Automation Server methods. It has the following topics:

- “Application Methods for Mobile Web Client Automation Server”
- “Business Component Methods for Mobile Web Client Automation Server” on page 358
- “Property Set Methods for Mobile Web Client Automation Server” on page 364

### Application Methods for Mobile Web Client Automation Server

Table 46 lists a summary of the application methods’ syntax.

Table 46 does not include methods that are not invoked directly from an Application object instance. For information on methods that are called with InvokeMethod on the Application object, see “InvokeMethod Methods for the Application Object” on page 143.

#### Table 46. Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| ActiveBusObject Method | Returns the business object for the business component of the active applet. | `Dim application as SiebelWebApplication
Dim busObject as SiebelBusObject
Set busObject = application.ActiveBusObject` |
| ActiveViewName Method  | Returns the name of the active view.                                        | `Dim application as SiebelWebApplication
Dim sView as String
sView = application.ActiveViewName` |
| CurrencyCode Method    | Returns the three-letter operating currency code.                           | `Dim application as SiebelWebApplication
Dim sCur as String
sCur = Application.CurrencyCode` |
| EnableExceptions Method| Enables or disables native COM error handling.                              | `Dim application as SiebelWebApplication
application.EnableExceptions(bEnable as Boolean)` |
### Table 46. Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetBusObject Method</td>
<td>Instantiates and returns a new instance of the business object specified in the argument.</td>
<td><code>Dim application as SiebelWebApplication</code>&lt;br&gt;<code>Dim busObject as SiebelBusObject</code>&lt;br&gt;<code>set busObject = application.GetBusObject(busobjName as String)</code></td>
</tr>
<tr>
<td>GetLastErrCode Method</td>
<td>Gets the last error code.</td>
<td><code>Dim application as SiebelWebApplication</code>&lt;br&gt;<code>Dim iErr as Integer</code>&lt;br&gt;<code>iErr = application.GetLastErrorCode</code></td>
</tr>
<tr>
<td>GetLastErrText Method</td>
<td>Returns the last error text message.</td>
<td><code>Dim application as SiebelWebApplication</code>&lt;br&gt;<code>Dim sText as String</code>&lt;br&gt;<code>sText = application.GetLastErrorText</code></td>
</tr>
<tr>
<td>GetProfileAttr Method</td>
<td>Returns the value of an attribute in a user profile.</td>
<td><code>Dim application as SiebelWebApplication</code>&lt;br&gt;<code>Dim profValue as String</code>&lt;br&gt;<code>profValue = application.GetProfileAttr(profName as String)</code></td>
</tr>
<tr>
<td>GetService Method</td>
<td>Instantiates and returns a new instance of the argument-specified service.</td>
<td><code>Dim application as SiebelWebApplication</code>&lt;br&gt;<code>Dim oService as SiebelService</code>&lt;br&gt;<code>set oService = Application.GetService(serviceName as String)</code></td>
</tr>
<tr>
<td>GetSharedGlobal Method</td>
<td>Returns the shared user-defined global variables.</td>
<td><code>Dim application as SiebelWebApplication</code>&lt;br&gt;<code>Dim name as String</code>&lt;br&gt;<code>name = application.GetSharedGlobal(sName as String)</code></td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>Calls the named specialized method.</td>
<td><code>Dim application as SiebelWebApplication</code>&lt;br&gt;<code>application.InvokeMethod(methodName as String, methodArgs as String or StringArray)</code></td>
</tr>
<tr>
<td>Login Method</td>
<td>Allows external applications to log in to the Mobile Web Client Automation Server.</td>
<td><code>Dim application as SiebelWebApplication</code>&lt;br&gt;<code>Dim sErr as String</code>&lt;br&gt;<code>sErr = application.Login(connectString as String, userName as String, password as String)</code></td>
</tr>
<tr>
<td>LoginId Method</td>
<td>Returns the login ID of the user who started the Siebel application.</td>
<td><code>Dim application as SiebelWebApplication</code>&lt;br&gt;<code>Dim sID as String</code>&lt;br&gt;<code>sID = application.LoginId</code></td>
</tr>
<tr>
<td>LoginName Method</td>
<td>Returns the login name of the user who started the Siebel application.</td>
<td><code>Dim application as SiebelWebApplication</code>&lt;br&gt;<code>Dim sUser as String</code>&lt;br&gt;<code>sUser = application.LoginName</code></td>
</tr>
<tr>
<td>Logoff Method</td>
<td>Terminates the Mobile Web Client session.</td>
<td><code>Dim application as SiebelWebApplication</code>&lt;br&gt;<code>Dim status as Boolean</code>&lt;br&gt;<code>Status = application.Logoff</code></td>
</tr>
</tbody>
</table>

**356 Siebel Object Interfaces Reference Version 8.0, Rev. B**
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewPropertySet</td>
<td>Constructs a new property set object.</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim propset As SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>set propset = application.NewPropertySet</td>
</tr>
<tr>
<td>PositionId Method</td>
<td>Returns the position ID that describes the user’s</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td>current position.</td>
<td>Dim sRow as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sRow = application.PositionId</td>
</tr>
<tr>
<td>PositionName Method</td>
<td>Returns the position name of the user’s current</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td>position.</td>
<td>Dim sPosition as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sPosition = application.PositionName</td>
</tr>
<tr>
<td>SetPositionId Method</td>
<td>Sets the active position to the Position ID</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td>specified in the argument.</td>
<td>Dim posId as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim status as Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>status = application.SetPositionId(posId)</td>
</tr>
<tr>
<td>SetPositionName Method</td>
<td>Sets the active position to the</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td>position name specified in the argument.</td>
<td>Dim posName as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim status as Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>status = application.SetPositionName(posName)</td>
</tr>
<tr>
<td>SetProfileAttr</td>
<td>Used in personalization to assign values to</td>
<td>Dim oApplication as SiebelWebApplication</td>
</tr>
<tr>
<td>Method</td>
<td>attributes in a user profile.</td>
<td>Dim bool as Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bool = oApplication.SetProfileAttr(name as String, value as String)</td>
</tr>
<tr>
<td>SetSharedGlobal</td>
<td>Sets a shared user-defined global variable.</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td>Method</td>
<td></td>
<td>Dim bool as Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bool = application.SetSharedGlobal(varName as String, value as String)</td>
</tr>
<tr>
<td>Trace Method</td>
<td>Appends a message to the trace file.</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>application.Trace(message as String)</td>
</tr>
<tr>
<td>TraceOff Method</td>
<td>Turns off the tracing started by TraceOn.</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim bool as Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bool = application.TraceOff</td>
</tr>
<tr>
<td>TraceOn Method</td>
<td>Turns tracing on.</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim bool as Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bool = application.TraceOn(filename as String, type as String,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Selection as String)</td>
</tr>
</tbody>
</table>
Business Component Methods for Mobile Web Client Automation Server

Table 47 lists a summary of the business component methods’ syntax.

Table 47 does not include methods that are not invoked directly from a Business Component object instance. For information on methods that are called with InvokeMethod on the Business Component object, see “InvokeMethod Methods for the Business Component Object” on page 216.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ActivateField Method</strong></td>
<td>Allows queries to retrieve data for the specified field.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.ActivateField(fieldName as String)</td>
</tr>
<tr>
<td><strong>ActivateMultipleFields Method</strong></td>
<td>Allows queries to retrieve data for the fields specified in the property set.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.ActivateMultipleFields(oPropSet as SiebelPropertySet)</td>
</tr>
<tr>
<td><strong>Associate Method</strong></td>
<td>Creates a new many-to-many relationship for the parent object through an association business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.Associate(whereIndicator as Integer)</td>
</tr>
<tr>
<td><strong>BusObject Method</strong></td>
<td>Returns the business object that contains the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim busObject as SiebelBusObject&lt;br&gt;Set BusObject = busComp.BusObject</td>
</tr>
<tr>
<td><strong>ClearToQuery Method</strong></td>
<td>Clears the current query and sort specifications on the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim bool as Boolean&lt;br&gt;bool = busComp.ClearToQuery</td>
</tr>
<tr>
<td><strong>DeactivateFields Method</strong></td>
<td>Deactivates every currently activated field.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim bool as Boolean&lt;br&gt;bool = busComp.DeactivateFields</td>
</tr>
<tr>
<td><strong>DeleteRecord Method</strong></td>
<td>Removes the current record from the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim bool as Boolean&lt;br&gt;bool = busComp.DeleteRecord</td>
</tr>
<tr>
<td><strong>ExecuteQuery Method</strong></td>
<td>Retrieves a set of BusComp records.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim bool as Boolean&lt;br&gt;bool = busComp.ExecuteQuery(cursorMode as Integer)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td>Syntax</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ExecuteQuery2 Method</td>
<td>Retrieves a set of BusComp records.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim bool as Boolean&lt;br&gt;bool = busComp.ExecuteQuery2(cursorMode as Integer, ignoreMaxCursorSize as Boolean)</td>
</tr>
<tr>
<td>FirstRecord Method</td>
<td>Moves to the first record in the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim blsRecord as Boolean&lt;br&gt;blsRecord = busComp.FirstRecord</td>
</tr>
<tr>
<td>GetAssocBusComp Method</td>
<td>Returns the association business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim AssocBusComp as SiebelBusComp Set AssocBusComp = busComp.GetAssocBusComp</td>
</tr>
<tr>
<td>GetFieldValue Method</td>
<td>Returns a value for the field specified in the argument.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim sValue as String&lt;br&gt;sValue = busComp.GetFieldValue(FieldName as String)</td>
</tr>
<tr>
<td>GetFormattedFieldValue Method</td>
<td>Returns a formatted value for the field specified in the argument.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim sValue as String&lt;br&gt;sValue = busComp.GetFormattedFieldValue(FieldName as String)</td>
</tr>
<tr>
<td>GetLastErrCode Method</td>
<td>Returns the last Siebel error number.</td>
<td>Dim buscomp as SiebelBusComp&lt;br&gt;Dim iErr as Integer&lt;br&gt;iErr = buscomp.GetLastErrCode</td>
</tr>
<tr>
<td>GetLastErrText Method</td>
<td>Returns the last error text message.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim sErr as String&lt;br&gt;sErr = busComp.GetLastErrText</td>
</tr>
<tr>
<td>GetMultipleFieldValues Method</td>
<td>Returns a value for the fields specified in the property set.</td>
<td>Dim buscomp as SiebelBusComp&lt;br&gt;buscomp.GetMultipleFieldValues(oPropSet as SiebelPropertySet, PValues as SiebelPropertySet)</td>
</tr>
<tr>
<td>GetMVGBusComp Method</td>
<td>Returns the MVG business component associated with the field specified in the argument.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim mVGBusComp as SiebelBusComp Set mVGBusComp = busComp.GetMVGBusComp(FieldName as String)</td>
</tr>
<tr>
<td>GetNamedSearch Method</td>
<td>Returns the argument-named search specification.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim sValue as String&lt;br&gt;sValue = busComp.GetNamedSearch(SetName as String)</td>
</tr>
</tbody>
</table>
## Table 47. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetPicklistBusComp Method | 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 Method     | Returns the current search expression.                                        | Dim busComp as SiebelBusComp  
Dim sExpr as String  
sExpr = busComp.GetSearchExpr |
| GetSearchSpec Method    | 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 Method  | 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 Method      | Returns the visibility mode for the business component.                       | Dim busComp as SiebelBusComp  
Dim iMode as Integer  
iMode = busComp.GetViewMode |
| InvokeMethod Method     | 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 Method       | Moves to the last record in the business component.                          | Dim busComp as SiebelBusComp  
Dim bReturn as Boolean  
bReturn = busComp.LastRecord |
| Name Method             | Returns the name of the business component.                                  | Dim busComp as SiebelBusComp  
Dim sName as String  
sName = busComp.Name |
| NewRecord Method        | Adds a new record to the business component.                                 | Dim busComp as SiebelBusComp  
Dim bool as Boolean  
bool = busComp.NewRecord(whereIndicator as Integer) |
| NextRecord Method       | Moves to the next record in the business component.                         | Dim busComp as SiebelBusComp  
Dim bReturn as Boolean  
bReturn = busComp.NextRecord |
| ParentBusComp Method    | Returns the parent business component.                                        | Dim busComp as SiebelBusComp  
Dim parentBusComp as SiebelBusComp  
Set parentBusComp = busComp.ParentBusComp |
### Table 47. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pick Method</td>
<td>Places the currently selected record in a picklist business component into the appropriate fields of the parent business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.Pick</td>
</tr>
<tr>
<td>PreviousRecord Method</td>
<td>Moves to the previous record in the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim bReturn as Boolean&lt;br&gt;bReturn = busComp.PreviousRecord</td>
</tr>
<tr>
<td>RefineQuery Method</td>
<td>Refines a query after a query has been executed.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.RefineQuery</td>
</tr>
<tr>
<td>SetFieldValue Method</td>
<td>Assigns a new value to the named field for the current row of the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.SetFieldValue(FieldName as String, FieldValue as String)</td>
</tr>
<tr>
<td>SetFormattedFieldValue Method</td>
<td>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.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.SetFormattedFieldValue(FieldName as String, FieldValue as String)</td>
</tr>
<tr>
<td>SetMultipleFieldValues Method</td>
<td>Assigns a new value to the fields specified in the property set for the current row of the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.SetMultipleFieldValues(propSet as SiebelPropertySet)</td>
</tr>
<tr>
<td>SetNamedSearch Method</td>
<td>Sets a named search specification on the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.SetNamedSearch(searchName as String, searchSpec as String)</td>
</tr>
<tr>
<td>SetSearchExpr Method</td>
<td>Sets the search expression for the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.SetSearchExpr(searchSpec as String)</td>
</tr>
<tr>
<td>SetSearchSpec Method</td>
<td>Sets the search specification for the specified field.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.SetSearchSpec(FieldName as String, searchSpec as String)</td>
</tr>
<tr>
<td>SetSortSpec Method</td>
<td>Sets the sort specification for a query.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.SetSortSpec(sortSpec as String)</td>
</tr>
<tr>
<td>SetUserProperty Method</td>
<td>Sets the value of the specified User Property.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.SetUserProperty(propertyName as String, newValue as String)</td>
</tr>
</tbody>
</table>
Table 47. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetBusComp Method | Returns the specified business component. | Dim busObject as SiebelBusObject  
Dim busComp as SiebelBusComp  
set busComp = busObject.GetBusComp(BusCompName as String) |
| GetLastErrCode Method | Returns the last Siebel error number. | Dim busObject as SiebelBusObject  
Dim iErr as Integer  
iErr = busObject.GetLastErrorCode |
| GetLastErrText Method | Returns the last error text message. | Dim busObject as SiebelBusObject  
Dim sValue as String  
sValue= busObject.GetLastErrorText |
| Name Method      | Returns the name of the business object. | Dim busObject as SiebelBusObject  
Dim sName as String  
sName = busObject.Name |

Table 48 lists a summary of the business object methods’ syntax.

Table 48. Business Object Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| SetViewMode Method   | Sets the visibility type for the business component. | Dim buscomp as SiebelBusComp  
buscomp.SetViewMode(mode As Integer) |
| UndoRecord Method     | Reverses any uncommitted changes made to the record. | Dim busComp as SiebelBusComp  
buComp.UndoRecord |
| WriteRecord Method    | Commits to the database any changes made to the current record. | Dim busComp as SiebelBusComp  
buComp.WriteRecord |
## Business Service Methods for Mobile Web Client Automation Server

Table 49 lists a summary of the business service methods’ syntax.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetFirstProperty     | Retrieves the name of the first property of a business service.              | Dim oService as SiebelService  
Dim sName as String  
  sName = oService.GetFirstProperty |
| GetNextProperty      | After 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  
  oService.InvokeMethod(methodName as String, InputArguments as SiebelPropertySet, OutputArguments as SiebelPropertySet) |
| Name Method          | 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  
  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 Methods for Mobile Web Client Automation Server

Table 50 lists a summary of the property set methods’ syntax.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddChild Method</td>
<td>Adds subsidiary property sets to a property set.</td>
<td>Dim oPropSet as SiebelPropertyset oPropSet.AddChild(childObject as SiebelPropertySet)</td>
</tr>
<tr>
<td>Copy Method</td>
<td>Returns a copy of a property set.</td>
<td>Dim oPropSet1 as SiebelPropertyset Dim oPropSet2 as SiebelPropertyset set oPropSet2 = oPropSet1.Copy</td>
</tr>
<tr>
<td>GetChild Method</td>
<td>Returns a specified child property set of a property set.</td>
<td>Dim oPropSet as SiebelPropertySet Dim childPropSet as SiebelPropertySet set childPropSet = oPropSet.GetChild(index as Long)</td>
</tr>
<tr>
<td>GetChildCount</td>
<td>Returns the number of child property sets attached to a parent property set.</td>
<td>Dim oPropSet as SiebelPropertySet Dim iCount as Long iCount = oPropSet.GetChildCount</td>
</tr>
<tr>
<td>GetFirstProperty Method</td>
<td>Returns the name of the first property in a property set.</td>
<td>Dim oPropSet as SiebelPropertySet Dim sPropName as String sPropName = oPropSet.GetFirstProperty</td>
</tr>
<tr>
<td>GetNextProperty Method</td>
<td>Returns the last Siebel error number.</td>
<td>Dim oPropSet as SiebelPropertySet Dim iErr as Integer iErr = oPropSet.GetLastErrCode</td>
</tr>
<tr>
<td>GetLastErrText Method</td>
<td>Returns the last error text message.</td>
<td>Dim oPropSet as SiebelPropertySet Dim sValue as String sValue = oPropSet.GetLastErrText</td>
</tr>
<tr>
<td>GetNextProperty Method</td>
<td>Returns the name of the next property in a property set.</td>
<td>Dim oPropSet as SiebelPropertySet Dim sPropName as String sPropName = oPropSet.GetNextProperty</td>
</tr>
<tr>
<td>GetProperty Method</td>
<td>Returns the value of a property when given the property name.</td>
<td>Dim oPropSet as SiebelPropertySet Dim sPropVal as String sPropVal = oPropSet.GetProperty(propName as String)</td>
</tr>
<tr>
<td>GetPropertyCount Method</td>
<td>Returns the number of properties contained within the property set.</td>
<td>Dim oPropSet as SiebelPropertySet Dim lCount as Long lCount = oPropSet.GetPropertyCount</td>
</tr>
</tbody>
</table>
### Table 50. Property Set Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| **Gettype Method** | Retrieves the data value stored in the type attribute of a property set. | `Dim oPropSet as SiebelPropertySet
Dim sTypeVal as String
sTypeVal = oPropSet.GetType` |
| **GetValue Method** | Retrieves the data value stored in the value attribute of a property set. | `Dim oPropSet as SiebelPropertySet
Dim sValVal as String
sValVal = oPropSet.GetValue` |
| **InsertChildAt Method** | 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 Method** | 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 Method** | Removes a child property set as a specified index from a parent property set. | `Dim oPropSet as SiebelPropertySet
oPropSet.RemoveChild(index as Long)` |
| **RemoveProperty Method** | Removes the property specified in its argument from a property set. | `Dim oPropSet as SiebelPropertySet
oPropSet.RemoveProperty(propName as String)` |
| **Reset Method** | Removes every property and child property set from a property set. | `Dim oPropSet as SiebelPropertySet
oPropSet.Reset` |
| **SetProperty Method** | 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 Method** | Assigns a data value to a type member of a property set. | `Dim oPropSet as SiebelPropertySet
oPropSet.SetType(value as String)` |
| **SetValue Method** | Assigns a data value to a value member of a property set. | `Dim oPropSet as SiebelPropertySet
oPropSet.SetValue(value as String)` |
This chapter provides a quick reference for Siebel Java Data Bean methods. It has the following topics:

- "Data Bean Methods for Java Data Bean"
- "Business Component Methods for Java Data Bean" on page 369
- "Business Object Methods for Java Data Bean" on page 373
- "Business Service Methods for Java Data Bean" on page 373
- "Property Set Methods for Java Data Bean" on page 374
- "SiebelException Methods for Java Data Bean" on page 376

### Data Bean Methods for Java Data Bean

Table 51 lists a summary of the SiebelDataBean methods’ syntax.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach Method</td>
<td>Allows an external application to reconnect to an existing Siebel session.</td>
<td>boolean attach(String sessionID) throws SiebelException</td>
</tr>
<tr>
<td>CurrencyCode Method</td>
<td>Returns the three-letter operating currency code.</td>
<td>String currencyCode()</td>
</tr>
<tr>
<td>Detach Method</td>
<td>Returns a string containing the Siebel session ID.</td>
<td>String detach() throws SiebelException</td>
</tr>
<tr>
<td>GetBusObject Method</td>
<td>Instantiates and returns a new instance of the business object specified in the argument.</td>
<td>SiebelBusObject getBusObject(String boName) throws SiebelException</td>
</tr>
<tr>
<td>GetProfileAttr Method</td>
<td>Returns the value of an attribute in a user profile.</td>
<td>String getProfileAttr(String attrName) throws SiebelException</td>
</tr>
<tr>
<td>GetService Method</td>
<td>Returns a specified service. If the service is not already running, it is constructed.</td>
<td>SiebelService getService(String serviceName) throws SiebelException</td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>Calls the named specialized method.</td>
<td>String invokeMethod(String name, String[] args) throws SiebelException</td>
</tr>
</tbody>
</table>
### Data Bean Methods for Java Data Bean

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Login Method</strong></td>
<td>Allows external applications to log in to the Data Bean.</td>
<td>boolean login(String connString, String userName, String passWord) throws SiebelException</td>
</tr>
<tr>
<td><strong>LoginId Method</strong></td>
<td>Returns the login ID of the user who started the Siebel application.</td>
<td>String loginId()</td>
</tr>
<tr>
<td><strong>LoginName Method</strong></td>
<td>Returns the login name of the user who started the Siebel application.</td>
<td>String loginName()</td>
</tr>
<tr>
<td><strong>Logoff Method</strong></td>
<td>Disconnects the client from the server.</td>
<td>boolean logoff() throws SiebelException</td>
</tr>
<tr>
<td><strong>NewPropertySet Method</strong></td>
<td>Constructs and returns a new property set object.</td>
<td>SiebelPropertySet newPropertySet()</td>
</tr>
<tr>
<td><strong>PositionId Method</strong></td>
<td>Returns the position ID that describes the user’s current position.</td>
<td>String positionId()</td>
</tr>
<tr>
<td><strong>PositionName Method</strong></td>
<td>Returns the position name of the user's current position.</td>
<td>String positionName()</td>
</tr>
<tr>
<td><strong>setPositionId Method</strong></td>
<td>Sets the active position to the Position ID specified in the argument.</td>
<td>boolean setPositionId(String posId) throws SiebelException</td>
</tr>
<tr>
<td><strong>setPositionName Method</strong></td>
<td>Sets the active position to the position name specified in the argument. Returns a Boolean value indicating if the method succeeded.</td>
<td>boolean setPositionName(String posName) throws SiebelException</td>
</tr>
<tr>
<td><strong>SetProfileAttr Method</strong></td>
<td>SetProfileAttr is used in personalization to assign values to attributes in a user profile.</td>
<td>boolean setProfileAttr(String attrName, String attrValue) throws SiebelException</td>
</tr>
<tr>
<td><strong>Trace Method</strong></td>
<td>The Trace method appends a message to the trace file. Trace is useful for debugging SQL query execution. This method does not trace Java standard output.</td>
<td>boolean trace(String message) throws SiebelException</td>
</tr>
</tbody>
</table>
Table 52. SiebelBusComp Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>TraceOff Method</td>
<td>TraceOff turns off the tracing started by the TraceOn method. This method does not trace Java standard output.</td>
<td>boolean traceOff() throws SiebelException</td>
</tr>
<tr>
<td>TraceOn Method</td>
<td>TraceOn turns on the tracking of allocations and deallocations of Siebel objects, and SQL statements generated by the Siebel application. This method does not trace Java standard output.</td>
<td>boolean traceOn(String filename, String Category, String selection) throws SiebelException</td>
</tr>
</tbody>
</table>

**Business Component Methods for Java Data Bean**

Table 52 lists a summary of the SiebelBusComp methods’ syntax.

Table 52 does not include methods that are not invoked directly from a Business Component object instance. For information on methods that are called with InvokeMethod on the Business Component object, see "InvokeMethod Methods for the Business Component Object" on page 216.

Table 52. SiebelBusComp Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActivateField Method</td>
<td>Allows queries to retrieve data for the specified field.</td>
<td>boolean activateField(String fieldName) throws SiebelException</td>
</tr>
<tr>
<td>ActivateMultipleFields Method</td>
<td>Allows queries to retrieve data for the fields specified in the property set.</td>
<td>boolean activateMultipleFields(SiebelPropertySet psFields) throws SiebelException</td>
</tr>
<tr>
<td>Associate Method</td>
<td>Creates a new many-to-many relationship for the parent object through an association business component.</td>
<td>boolean associate(boolean isInsertBefore) throws SiebelException</td>
</tr>
<tr>
<td>BusObject Method</td>
<td>Returns the business object that contains the business component.</td>
<td>SiebelBusObject busObject() throws SiebelException</td>
</tr>
<tr>
<td>ClearToQuery Method</td>
<td>Clears the current query and sort specifications on the business component.</td>
<td>boolean clearToQuery() throws SiebelException</td>
</tr>
</tbody>
</table>
### Table 52. SiebelBusComp Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeactivateFields Method</td>
<td>Deactivates every currently activated field.</td>
<td>boolean deactivateFields()</td>
</tr>
<tr>
<td>DeleteRecord Method</td>
<td>Removes the current record from the business component.</td>
<td>boolean deleteRecord() throwsSiebelException</td>
</tr>
<tr>
<td>ExecuteQuery Method</td>
<td>Retrieves a set of BusComp records.</td>
<td>boolean executeQuery(boolean cursorMode) throws SiebelException</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE:</strong> When using the ExecuteQuery method with Java Data Bean, use True for ForwardOnly and False for ForwardBackward.</td>
<td></td>
</tr>
<tr>
<td>ExecuteQuery2 Method</td>
<td>Retrieves a set of BusComp records.</td>
<td>boolean executeQuery2(boolean cursorMode, boolean ignoreMaxCursorSize) throws SiebelException</td>
</tr>
<tr>
<td>FirstRecord Method</td>
<td>Moves to the first record in the business component.</td>
<td>boolean firstRecord() throws SiebelException</td>
</tr>
<tr>
<td>GetFieldValue Method</td>
<td>Returns a value for the field specified in the argument.</td>
<td>String getFieldValue(String fieldName) throws SiebelException</td>
</tr>
<tr>
<td>GetFormattedFieldValue Method</td>
<td>Returns a formatted value for the field specified in the argument.</td>
<td>String getFormattedFieldValue(String fieldName) throws SiebelException</td>
</tr>
<tr>
<td>GetMultipleFieldValues Method</td>
<td>Returns values for the fields specified in the property set.</td>
<td>boolean getMultipleFieldValues(SiebelPropertySet Src, SiebelPropertySet result) throws SiebelException</td>
</tr>
<tr>
<td>GetMVGBusComp Method</td>
<td>Returns the MVG business component associated with the field specified in the argument.</td>
<td>SiebelBusComp getMVGBusComp(String fieldName) throws SiebelException</td>
</tr>
<tr>
<td>GetNamedSearch Method</td>
<td>Returns the argument-named search specification.</td>
<td>String getNamedSearch(String searchName) throws SiebelException</td>
</tr>
<tr>
<td>GetPicklistBusComp Method</td>
<td>Returns the pick business component associated with the field specified in the argument.</td>
<td>SiebelBusComp getPicklistBusComp(String fieldName) throws SiebelException</td>
</tr>
<tr>
<td>GetSearchExpr Method</td>
<td>Returns the current search expression.</td>
<td>String getSearchExpr() throws SiebelException</td>
</tr>
</tbody>
</table>
### Table 52. SiebelBusComp Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetSearchSpec Method</td>
<td>Returns the current search specification for the field specified in the argument.</td>
<td><code>String getSearchSpec(String fieldName) throws SiebelException</code></td>
</tr>
<tr>
<td>GetUserProperty Method</td>
<td>Returns the value for the specified property.</td>
<td><code>String getUserProperty(String property) throws SiebelException</code></td>
</tr>
<tr>
<td>GetViewMode Method</td>
<td>Returns the visibility mode for the business component.</td>
<td><code>int getViewMode()</code></td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>Calls the specialized method named in the argument. The methodArgs parameter must be an array of strings.</td>
<td><code>String invokeMethod(String methodName, String[] methodArgs) throws SiebelException</code></td>
</tr>
<tr>
<td>LastRecord Method</td>
<td>Moves to the last record in the business component.</td>
<td><code>boolean lastRecord() throws SiebelException</code></td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the business component.</td>
<td><code>String name()</code></td>
</tr>
<tr>
<td>NewRecord Method</td>
<td>Adds a new record to the business component.</td>
<td><code>boolean newRecord(boolean isInsertBefore) throws SiebelException</code></td>
</tr>
<tr>
<td>NextRecord Method</td>
<td>Moves to the next record in the business component.</td>
<td><code>boolean nextRecord() throws SiebelException</code></td>
</tr>
<tr>
<td>ParentBusComp Method</td>
<td>Returns the parent business component.</td>
<td><code>SiebelBusComp parentBusComp() throws SiebelException</code></td>
</tr>
<tr>
<td>Pick Method</td>
<td>Places the currently selected record in a picklist business component into the appropriate fields of the parent business component.</td>
<td><code>boolean pick() throws SiebelException</code></td>
</tr>
<tr>
<td>PreviousRecord Method</td>
<td>Moves to the previous record in the business component.</td>
<td><code>boolean previousRecord() throws SiebelException</code></td>
</tr>
<tr>
<td>RefineQuery Method</td>
<td>Refines a query after a query has been executed.</td>
<td><code>boolean refineQuery() throws SiebelException</code></td>
</tr>
<tr>
<td>Release Method</td>
<td>Enables the release of the business component and its resources on the Siebel Server.</td>
<td><code>void release()</code></td>
</tr>
<tr>
<td>SetFieldVal Method</td>
<td>Assigns a new value to the named field for the current row of the business component.</td>
<td><code>boolean setFieldValue(String fieldName, String fieldValue) throws SiebelException</code></td>
</tr>
</tbody>
</table>
### Table 52. SiebelBusComp Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SetFormattedFieldValue Method</strong></td>
<td>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.</td>
<td>boolean setFormattedFieldValue(String fieldName, String fieldValue) throws SiebelException</td>
</tr>
<tr>
<td><strong>SetMultipleFieldValues Method</strong></td>
<td>Assigns new values to the multiple fields specified in the property set for the current row of the business component.</td>
<td>boolean setMultipleFieldValues(SiebelPropertySet psFields) throws SiebelException</td>
</tr>
<tr>
<td><strong>SetNamedSearch Method</strong></td>
<td>Sets a named search specification on the business component.</td>
<td>boolean setNamedSearch(String searchName, String searchText) throws SiebelException</td>
</tr>
<tr>
<td><strong>SetSearchExpr Method</strong></td>
<td>Sets an entire search expression on the business component.</td>
<td>boolean setSearchExpr(String searchExpr) throws SiebelException</td>
</tr>
<tr>
<td><strong>SetSearchSpec Method</strong></td>
<td>Sets the search specification for the specified field.</td>
<td>boolean setSearchSpec(String fieldName, String searchSpec) throws SiebelException</td>
</tr>
<tr>
<td><strong>SetSortSpec Method</strong></td>
<td>Sets the sort specification for a query.</td>
<td>boolean setSortSpec(String sortSpec) throws SiebelException</td>
</tr>
<tr>
<td><strong>SetUserProperty Method</strong></td>
<td>Sets the value of the specified User Property.</td>
<td>boolean setUserProperty(String propName, String propVal)</td>
</tr>
<tr>
<td><strong>SetViewMode Method</strong></td>
<td>Sets the visibility type for the business component.</td>
<td>boolean setViewMode(int mode) throws SiebelException</td>
</tr>
<tr>
<td><strong>UndoRecord Method</strong></td>
<td>Reverses any uncommitted changes made to the record.</td>
<td>boolean undoRecord() throws SiebelException</td>
</tr>
<tr>
<td><strong>WriteRecord Method</strong></td>
<td>Commits to the database any changes made to the current record.</td>
<td>boolean writeRecord() throws SiebelException</td>
</tr>
</tbody>
</table>
Business Object Methods for Java Data Bean

Table 53 lists a summary of the SiebelBusObject methods’ syntax.

Table 53. SiebelBusObject Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetBusComp Method</td>
<td>Returns the specified business component.</td>
<td><code>SiebelBusComp getBusComp(String busCompName) throws SiebelException</code></td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the business object.</td>
<td><code>String name()</code></td>
</tr>
<tr>
<td>Release Method</td>
<td>Enables the release of the business object and its resources on the Siebel Server.</td>
<td><code>void release()</code></td>
</tr>
</tbody>
</table>

Business Service Methods for Java Data Bean

Table 54 lists a summary of the SiebelService methods’ syntax.

Table 54. SiebelService Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetFirstProperty Method</td>
<td>Retrieves the name of the first property of a business service.</td>
<td><code>String getFirstProperty()</code></td>
</tr>
<tr>
<td>getNextProperty Method</td>
<td>After the name of the first property has been retrieved, retrieves the name of the next property of a business service.</td>
<td><code>String getNextProperty()</code></td>
</tr>
<tr>
<td>GetProperty Method</td>
<td>Retrieves the value stored in the specified property.</td>
<td><code>String getProperty(String propName) throws SiebelException</code></td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>Calls a specialized method or a user-created method on the business service.</td>
<td><code>boolean invokeMethod(String methodName, SiebelPropertySet inputPropertySet, SiebelPropertySet outputPropertySet) throws SiebelException</code></td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the business service.</td>
<td><code>String Name()</code></td>
</tr>
</tbody>
</table>
Table 54. SiebelService Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyExists Method</td>
<td>Returns a Boolean value indicating whether the property specified in the argument exists.</td>
<td>boolean propertyExists(String propName) throws SiebelException</td>
</tr>
<tr>
<td>Release Method</td>
<td>Enables the release of the Business Service and its resources on the Siebel Server.</td>
<td>void release()</td>
</tr>
<tr>
<td>RemoveProperty Method</td>
<td>Removes a property from a business service.</td>
<td>void removeProperty(String propName) throws SiebelException</td>
</tr>
<tr>
<td>SetProperty Method</td>
<td>Assigns a value to a property of a business service.</td>
<td>void setProperty(String propName, String propValue) throws SiebelException</td>
</tr>
</tbody>
</table>

Table 55. SiebelPropertySet Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddChild Method</td>
<td>Adds subsidiary property sets to a property set.</td>
<td>int addChild(SiebelPropertySet propertySet)</td>
</tr>
<tr>
<td>Copy Method</td>
<td>Returns a copy of a property set.</td>
<td>SiebelPropertySet copy(SiebelPropertySet propertySet)</td>
</tr>
<tr>
<td>GetByteValue Method</td>
<td>Returns a byte array if a byte value has been set.</td>
<td>public byte[] getByteValue()</td>
</tr>
<tr>
<td>GetChild Method</td>
<td>Returns a specified child property set of a property set.</td>
<td>SiebelPropertySet getChild(int index)</td>
</tr>
<tr>
<td>GetChildCount Method</td>
<td>Returns the number of child property sets attached to a parent property set.</td>
<td>int getCount()</td>
</tr>
<tr>
<td>GetFirstProperty Method</td>
<td>Returns the name of the first property in a property set.</td>
<td>String getFirstProperty()</td>
</tr>
<tr>
<td>GetNextProperty Method</td>
<td>Returns the name of the next property in a property set.</td>
<td>String getNextProperty()</td>
</tr>
<tr>
<td>GetProperty Method</td>
<td>Returns the value of a property when given the property name.</td>
<td>String getProperty(String propertyName)</td>
</tr>
<tr>
<td>GetPropertyCount Method</td>
<td>Returns the number of properties attached to a property set.</td>
<td>int getRowCount()</td>
</tr>
</tbody>
</table>
### SiebelPropertySet Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetType Method</td>
<td>Returns the value stored in the Type attribute of a PropertySet.</td>
<td><code>String getType()</code></td>
</tr>
<tr>
<td>GetValue Method</td>
<td>Returns the value stored in the Value attribute of a PropertySet.</td>
<td><code>String getValue()</code></td>
</tr>
<tr>
<td>InsertChildAt Method</td>
<td>Inserts a child property set into a parent property set at a specific location.</td>
<td><code>boolean insertChildAt(SiebelPropertySet propertySet, int index)</code></td>
</tr>
<tr>
<td>PropertyExists Method</td>
<td>Returns a Boolean value indicating whether the property specified in the argument exists.</td>
<td><code>boolean propertyExists(String propertyName)</code></td>
</tr>
<tr>
<td>RemoveChild Method</td>
<td>Removes a child property set as a specified index from a parent property set.</td>
<td><code>boolean removeChild(int index)</code></td>
</tr>
<tr>
<td>RemoveProperty Method</td>
<td>Removes the property specified in its argument from a property set.</td>
<td><code>boolean removeProperty(String propertyName)</code></td>
</tr>
<tr>
<td>Reset Method</td>
<td>Removes every property and child property set from a property set.</td>
<td><code>boolean reset()</code></td>
</tr>
<tr>
<td>SetByteValue Method</td>
<td>Sets the value portion of a property set.</td>
<td><code>public void setByteValue(byte[] value)</code></td>
</tr>
<tr>
<td>SetProperty Method</td>
<td>Assigns a value to the property of a property set specified in its argument.</td>
<td><code>boolean setProperty(String propertyName, String propertyValue)</code></td>
</tr>
<tr>
<td>SetType Method</td>
<td>Assigns a data value to a type member of a property set.</td>
<td><code>boolean setType(String type)</code></td>
</tr>
<tr>
<td>SetValue Method</td>
<td>Assigns a data value to a value member of a property set.</td>
<td><code>boolean setValue(String value)</code></td>
</tr>
</tbody>
</table>
SiebelException Methods for Java Data Bean

Table 56 lists a summary of the SiebelException methods’ syntax.

Table 56. SiebelException Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetErrorCode Method</td>
<td>Gets a numeric error code.</td>
<td>int getErrorCode()</td>
</tr>
<tr>
<td>GetErrorMessage Method</td>
<td>Gets an error message.</td>
<td>String getErrorMessage()</td>
</tr>
</tbody>
</table>

For more information on the Java Data Bean Interface, read the Javadoc files, which are contained in a file named Siebel_JavaDoc.jar. This file is normally located in: \siebsrvr\CLASSES.
Siebel Web Client Automation Server Quick Reference

This chapter provides a quick reference for Siebel Web Client Automation Server methods. It has the following topics:

- “SiebelHTMLApplication Methods for Siebel Web Client Automation Server”
- “SiebelService Methods for Siebel Web Client Automation Server” on page 378
- “Property Set Methods for Siebel Web Client Automation Server” on page 379

SiebelHTMLApplication Methods for Siebel Web Client Automation Server

Table 57 lists a summary of the SiebelHTMLApplication methods’ syntax.

Table 57 does not include methods that are not invoked directly from an Application object instance. For information on methods that are called with InvokeMethod on the Application object, see “InvokeMethod Methods for the Application Object” on page 143.

Table 57. SiebelHTMLApplication Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetLastErrCode Method | Returns the last error code.                     | Dim siebelApp As SiebelHTMLApplication  
Dim iErr as Long  
iErr = siebelApp.GetLastErrCode |
| GetLastErrText Method | Returns the last error text message.             | Dim siebelApp As SiebelHTMLApplication  
Dim sText as String  
sText = siebelApp.GetLastErrText |
| GetService Method    | 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) |
Table 57. SiebelHTMLApplication Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| Name Method     | Returns the name of the current application as defined in the repository. | Dim siebelApp As SiebelHTMLApplication  
 name = siebelApp.Name                                                 |
| NewPropertySet Method | Constructs and returns a new property set object. | Dim siebelApp As SiebelHTMLApplication  
 Set propSet = siebelApp.NewPropertySet                                   |

Table 58. SiebelService Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetNextProperty Method | Returns the last error code. | Dim svc As SiebelService  
 iErr = svc.GetLastErrCode                                                  |
| InvokeMethod Method | Calls a specialized method or a user-created method on the business service. | Dim svc As SiebelService  
 svc.InvokeMethod(MethodName as String, inputPropSet as SiebelPropertySet, outputPropSet as SiebelPropertySet) |
| Name Method     | Returns the name of the business service. | Dim svc As SiebelService  
 name = svc.Name                                                              |
# Property Set Methods for Siebel Web Client Automation Server

Table 59 lists a summary of the SiebelPropertySet methods’ syntax.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| AddChild Method         | Adds subsidiary property sets to a property set.        | Dim oPropSet as SiebelPropertySet  
                         | oPropSet.AddChild(childObject as SiebelPropertySet)                     |
| Copy Method             | Returns a copy of a property set.                       | Dim oPropSet1 as SiebelPropertySet  
                         | Dim oPropSet2 as SiebelPropertySet  
                         | Set oPropSet2 = oPropSet1.Copy                                          |
| GetChild Method         | 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)                   |
| GetChildCount Method    | 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 Method | Returns the name of the first property in a property set. | Dim oPropSet as SiebelPropertySet  
                         | Dim sPropName as String  
                         | sPropName = oPropSet.GetFirstProperty                                  |
| GetLastErrCode Method   | Returns the last error code.                           | Dim oPropSet as SiebelPropertySet  
                         | Dim iErr as Long  
                         | iErr = oPropSet.GetLastError                                           |
| GetLastErrText Method   | Returns the last error text message.                    | Dim oPropSet as SiebelPropertySet  
                         | Dim sText as String  
                         | sText = oPropSet.GetLastErrText                                       |
| GetNextProperty Method  | Returns the name of the next property in a property set. | Dim oPropSet as SiebelPropertySet  
                         | Dim sPropName as String  
                         | sPropName = oPropSet.GetNextProperty                                  |
| GetProperty Method      | 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 Method | Returns the number of properties attached to a property set. | Dim oPropSet as SiebelPropertySet  
                         | Dim iCount as Long  
                         | iCount = oPropSet.GetPropertyCount()                                 |
### SiebelPropertySet Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| **GetType Method**   | Returns the value stored in a type in a property set. | Dim oPropSet as SiebelPropertySet  
|                      |                                                       | Dim type as String  
|                      |                                                       | type = oPropSet.GetType                                                |
| **GetValue Method**  | Returns a value stored as part of a property set.    | Dim oPropSet as SiebelPropertySet  
|                      |                                                       | Dim sValue as String  
|                      |                                                       | sValue = oPropSet.GetValue                                             |
| **InsertChildAt Method** | 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 Method** | 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 Method** | Removes a child property set as a specified index from a parent property set. | Dim oPropSet as SiebelPropertySet  
|                      |                                                       | oPropSet.RemoveChild(index as Long)                                    |
| **RemoveProperty Method** | Removes the property specified in its argument from a property set. | Dim oPropSet as SiebelPropertySet  
|                      |                                                       | oPropSet.RemoveProperty(propName as String)                            |
| **Reset Method**     | Removes every property and child property set from a property set. | Dim oPropSet as SiebelPropertySet  
|                      |                                                       | oPropSet.Reset                                                         |
| **SetProperty Method** | 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 Method**   | Assigns a data value to a type member of a property set. | Dim oPropSet as SiebelPropertySet  
|                      |                                                       | oPropSet.SetType(value as String)                                       |
| **SetValue Method**  | Assigns a data value to a value member of a property set. | Dim oPropSet as SiebelPropertySet  
|                      |                                                       | oPropSet.SetValue(value as String)                                      |
11 Siebel VB Quick Reference

This chapter provides a quick reference for Siebel VB methods and events. It has the following topics:

- "Applet Methods for Siebel VB"
- "WebApplet Events for Siebel VB" on page 382
- "Application Methods for Siebel VB" on page 383
- "Application Events for Siebel VB" on page 386
- "Business Component Methods for Siebel VB" on page 386
- "Business Component Events for Siebel VB" on page 391
- "Business Object Methods for Siebel VB" on page 393
- "Business Service Methods for Siebel VB" on page 393
- "Business Service Events for Siebel VB" on page 394
- "Property Set Methods for Siebel VB" on page 395
- "Miscellaneous Methods for Siebel VB" on page 397

Applet Methods for Siebel VB

Table 60 lists a summary of the applet methods’ syntax.

Table 60. Applet Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| BusComp Method  | 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 Method| Function that returns the business object for the business component of the applet. | `Dim oApplet as Applet
Dim oBusObject as BusObject
Set oBusObject = oApplet.BusObject` |
| InvokeMethod    | Invokes the specialized or custom method specified by its argument. | `Dim oApplet as Applet
oApplet.InvokeMethod methodName as String, methodArgs as String or StringArray` |
| Name Method     | Function that returns the name of the applet.         | `Dim oApplet as Applet
Dim sApplet as String
sApplet = oApplet.Name` |
**WebApplet Events for Siebel VB**

Table 61 lists a summary of the WebApplet events.

Table 61.  WebApplet Events Summary

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebApplet_InvokeMethod Event</td>
<td>Called after a specialized method or a user-defined method on the Web applet has been executed.</td>
<td>WebApplet_InvokeMethod(MethodName as String)</td>
</tr>
<tr>
<td>WebApplet_PreCanInvokeMethod Event</td>
<td>Called before the PreInvokeMethod, allowing the developer to determine whether or not the user has the authority to invoke the applet method.</td>
<td>WebApplet_PreCanInvokeMethod(MethodName as String, CanInvoke as String)</td>
</tr>
<tr>
<td>WebApplet_PreInvokeMethod Event</td>
<td>Called before a specialized method for the Web applet is invoked or a user-defined method is invoked through oWebApplet.Invoke Method.</td>
<td>WebApplet_PreInvokeMethod(MethodName as String) As Integer</td>
</tr>
<tr>
<td>WebApplet_Load Event</td>
<td>Called just after an applet is loaded.</td>
<td>WebApplet_Load</td>
</tr>
</tbody>
</table>
Table 61. WebApplet Events Summary

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebApplet_ShowControl Event</td>
<td>Allows scripts to modify the HTML generated by the Siebel Web Engine to render a control on a Web page in a standard interactivity application.</td>
<td>WebApplet_ShowControl (controlName as String, property as String, mode as String, HTML as String)</td>
</tr>
<tr>
<td>WebApplet_ShowListColumn Event</td>
<td>Allows scripts to modify the HTML generated by the Siebel Web Engine to render a list column on a Web page in a standard interactivity application.</td>
<td>WebApplet_ShowListColumn (columnName as String, property as String, mode as String, HTML as String)</td>
</tr>
</tbody>
</table>

Application Methods for Siebel VB

Table 62 lists a summary of the application methods’ syntax.

Table 62 does not include methods that are not invoked directly from an Application object instance. For information on methods that are called with InvokeMethod on the Application object, see “InvokeMethod Methods for the Application Object” on page 143.

Table 62. Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| ActiveBusObject Method | Returns the business object of the active view. | Dim oApplication as Application  
Dim oBusObject as BusObject  
Set oBusObject = oApplication.ActiveBusObject |
| ActiveViewName Method  | Function that returns the name of the active view. | Dim oApplication as Application  
Dim sView as String  
sView = oApplication.ActiveViewName |
| CurrencyCode Method    | Returns the three-letter operating currency code. | Dim oApplication as Application  
Dim sCur as String  
sCur = oApplication.CurrencyCode |
### Table 62. Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| **GetBusObject Method** | 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)` |
| **GetProfileAttr Method** | 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 Method** | 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 Method** | Gets the shared user-defined global variables. | `Dim oApplication as Application  
Dim sName as String  
sName = Application.GetSharedGlobal(varName as String)` |
| **GotoView Method** | Activates the named view and its business object. | `Dim oApplication as Application  
oApplication.GotoView viewName as String[, BusinessObjectName as BusObject]` |
| **InvokeMethod Method** | Calls the named specialized method. | `Dim oApplication as Application  
oApplication.InvokeMethod(methodName as String, methodArgs as String or StringArray)` |
| **LoginId Method** | 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 Method** | 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 Method** | Constructs and returns a new property set object. | `Dim oApplication as Application  
Dim oPropSet as PropertySet  
oPropSet = oApplication.NewPropertySet` |
| **PositionId Method** | 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 Method** | Function that returns the position name of the user's current position. | `Dim oApplication as Application  
Dim sPosition as String  
sPosition = oApplication.PositionName` |
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>RaiseError Method</td>
<td>Raises a scripting error message to the browser. The error code is a canonical number.</td>
<td>Dim oApplication as Application&lt;br/&gt;oApplication.RaiseError keyValue as String, param1 as String, ...</td>
</tr>
<tr>
<td>RaiseErrorText Method</td>
<td>Raises a scripting error message to the browser. The error text is the specified literal string.</td>
<td>Dim oApplication as Application&lt;br/&gt;oApplication.RaiseErrorText message as String</td>
</tr>
<tr>
<td>SetPositionId Method</td>
<td>Sets the active position to the position ID specified in the argument.</td>
<td>Dim oApplication as Application&lt;br/&gt;oApplication.SetPositionId posId as string</td>
</tr>
<tr>
<td>SetPositionName Method</td>
<td>Sets the active position to the position name specified in the argument. Returns a Boolean value indicating whether or not method succeeded.</td>
<td>Dim oApplication as Application&lt;br/&gt;oApplication.SetPositionName posName as string</td>
</tr>
<tr>
<td>SetProfileAttr Method</td>
<td>Used in personalization to assign values to attributes in a user profile.</td>
<td>Dim oApplication as Application&lt;br/&gt;oApplication.SetProfileAttr name as String, value as String</td>
</tr>
<tr>
<td>SetSharedGlobal Method</td>
<td>Sets a shared user-defined global variable.</td>
<td>Dim oApplication as Application&lt;br/&gt;oApplication.SetSharedGlobal varName as String, value as String</td>
</tr>
<tr>
<td>Trace Method</td>
<td>Appends a message to the trace file.</td>
<td>Dim oApplication as Application&lt;br/&gt;oApplication.Trace message as String</td>
</tr>
<tr>
<td>TraceOff Method</td>
<td>Turns off the tracing started by TraceOn.</td>
<td>Dim oApplication as Application&lt;br/&gt;oApplication.TraceOff</td>
</tr>
<tr>
<td>TraceOn Method</td>
<td>Turns tracing on.</td>
<td>Dim oApplication as Application&lt;br/&gt;oApplication.TraceOn filename as String, type as String, selection as String</td>
</tr>
</tbody>
</table>
### Application Events for Siebel VB

Table 63 lists a summary of the application events.

#### Table 63. Application Events Summary

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application_Close Event</td>
<td>Called before the application exits.</td>
<td>Application_Close</td>
</tr>
<tr>
<td>Application_Navigate Event</td>
<td>Called after the client has navigated to a view.</td>
<td>Application_Navigate</td>
</tr>
<tr>
<td>Application_InvokeMethod Event</td>
<td>Called after a specialized method is invoked.</td>
<td>Application_InvokeMethod (MethodName as String)</td>
</tr>
<tr>
<td>Application_PreInvokeMethod Event</td>
<td>Called before a specialized method is invoked.</td>
<td>Application_PreInvokeMethod (MethodName as String) as Integer</td>
</tr>
<tr>
<td>Application_PreNavigate Event</td>
<td>Called before the client has navigated from one view to the next.</td>
<td>Application_PreNavigate (DestViewName as String, DestBusObjName as String)</td>
</tr>
<tr>
<td>Application_Start Event</td>
<td>Called when the client starts.</td>
<td>Application_Start (commandLine as String)</td>
</tr>
</tbody>
</table>

### Business Component Methods for Siebel VB

Table 64 lists a summary of the business component methods’ syntax.

Table 64 does not include methods that are not invoked directly from a Business Component object instance. For information on methods that are called with InvokeMethod on the Business Component object, see “InvokeMethod Methods for the Business Component Object” on page 216.

#### Table 64. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActivateField Method</td>
<td>Allows queries to retrieve data for the specified field.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;oBusComp.ActivateField fieldName as String</td>
</tr>
<tr>
<td>ActivateMultipleFields Method</td>
<td>Allows queries to retrieve data for the fields specified in the property set.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;oBusComp.ActivateMultipleFields oPropSet as PropertySet</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td>Syntax</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Associate Method</td>
<td>Creates a new many-to-many relationship for the parent object through an association business component.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;oBusComp.Associate whereIndicator as Integer</td>
</tr>
<tr>
<td>BusObject Method</td>
<td>Function that returns the business object that contains the business component.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;Dim oBusObject as BusObject&lt;br&gt;Set oBusObject = oBusComp.BusObject</td>
</tr>
<tr>
<td>ClearToQuery Method</td>
<td>Clears the current query and sort specifications on the business component.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;oBusComp.ClearToQuery</td>
</tr>
<tr>
<td>DeactivateFields Method</td>
<td>Deactivates every currently activated field.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;oBusComp.DeactivateFields</td>
</tr>
<tr>
<td>DeleteRecord Method</td>
<td>Removes the current record from the business component.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;oBusComp.DeleteRecord</td>
</tr>
<tr>
<td>ExecuteQuery Method</td>
<td>Retrieves a set of BusComp records.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;oBusComp.ExecuteQuery cursorMode as Integer</td>
</tr>
<tr>
<td>ExecuteQuery2 Method</td>
<td>Retrieves a set of BusComp records.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;oBusComp.ExecuteQuery2 cursorMode as Integer, ignoreMaxCursorSize as Integer</td>
</tr>
<tr>
<td>FirstRecord Method</td>
<td>Moves to the first record in the business component.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;Dim iIsRecord as Integer&lt;br&gt;iIsRecord = oBusComp.FirstRecord</td>
</tr>
<tr>
<td>FirstSelected Method</td>
<td>Moves the focus to the first record of the multiple selection in the business component.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;Dim iIsMultipleSelection as Integer&lt;br&gt;iIsMultipleSelection = oBusComp.FirstSelected</td>
</tr>
<tr>
<td>GetAssocBusComp Method</td>
<td>Function that returns the association business component.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;Dim AssocBusComp as BusComp&lt;br&gt;Set AssocBusComp = oBusComp.GetAssocBusComp</td>
</tr>
<tr>
<td>GetFieldValue Method</td>
<td>Function that returns a value for the argument-specified field.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;Dim sValue as String&lt;br&gt;sValue = oBusComp.GetFieldValue(FieldName as String)</td>
</tr>
</tbody>
</table>
Table 64. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetFormattedFieldValue Method  | 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 Method  | Returns a value for the fields specified in the property set.                | `Dim oBusComp as BusComp
oBusComp.GetMultipleFieldValues(Fields as PropertySet, oValues as PropertySet)` |
| GetMVGBusComp Method           | 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 Method          | Function that returns the argument-named search specification.               | `Dim oBusComp as BusComp
Dim sValue as String
sValue = oBusComp.GetNamedSearch(SearchName as String)` |
| GetPicklistBusComp Method      | 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 Method           | Function that returns the current search expression.                         | `Dim oBusComp as BusComp
Dim sExpr as String
sExpr = oBusComp.GetSearchExpr` |
| GetSearchSpec Method           | 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)` |
| GetSortSpec Method             | Function that returns the active sort specification of the object that has context. | `Dim sSortSpec as String
sSortSpec = GetSortSpec` |
| GetUserProperty Method         | 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 Method             | Function that returns the visibility mode for the business component.        | `Dim oBusComp as BusComp
Dim iMode as Integer
iMode = oBusComp.GetViewMode` |
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvokeMethod Method</td>
<td>Calls the specialized method or user-created method specified in the argument.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;oBusComp.InvokeMethod(methodName as String, methodArgs as String or StringArray)</td>
</tr>
<tr>
<td>LastRecord Method</td>
<td>Moves to the last record in the business component.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;Dim iReturn as Integer&lt;br&gt;iReturn = oBusComp.LastRecord</td>
</tr>
<tr>
<td>Name Method</td>
<td>Function that returns the name of the business component.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;Dim sName as String&lt;br&gt;sName = oBusComp.Name</td>
</tr>
<tr>
<td>NewRecord Method</td>
<td>Adds a new record to the business component.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;oBusComp.NewRecord whereIndicator as Integer</td>
</tr>
<tr>
<td>NextRecord Method</td>
<td>Moves to the next record in the business component.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;Dim iReturn as Integer&lt;br&gt;iReturn = oBusComp.NextRecord</td>
</tr>
<tr>
<td>NextSelected Method</td>
<td>Moves to the next record of the current multiple selection.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;Dim iReturn as Integer&lt;br&gt;iReturn = oBusComp.NextSelected</td>
</tr>
<tr>
<td>ParentBusComp Method</td>
<td>Function that returns the parent business component.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;Dim parentBusComp as BusComp&lt;br&gt;Set parentBusComp = oBusComp.ParentBusComp</td>
</tr>
<tr>
<td>Pick Method</td>
<td>Places the currently selected record in a picklist business component into the appropriate fields of the parent business component.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;oBusComp.Pick</td>
</tr>
<tr>
<td>PreviousRecord Method</td>
<td>Moves to the previous record in the business component.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;Dim iReturn as Integer&lt;br&gt;iReturn = oBusComp.PreviousRecord</td>
</tr>
<tr>
<td>RefineQuery Method</td>
<td>Refines a query after a query has been executed.</td>
<td>Dim oBusComp as BusComp&lt;br&gt;oBusComp.RefreshQuery</td>
</tr>
</tbody>
</table>
### Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| SetFieldValue Method          | 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 Method  | 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 Method | 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 Method         | Sets a named search specification on the business component.                 | `Dim oBusComp as BusComp
oBusComp.SetNamedSearch searchName as String, searchSpec as String` |
| SetSearchExpr Method          | Sets the entire search expression for the business component.                | `Dim oBusComp as BusComp
oBusComp.SetSearchExpr searchSpec as String` |
| SetSearchSpec Method          | Sets the search specification for the specified field.                       | `Dim oBusComp as BusComp
oBusComp.SetSearchSpec fieldName as String, searchSpec as String` |
| SetSortSpec Method            | Sets the sort specification for a query.                                     | `Dim oBusComp as BusComp
oBusComp.SetSortSpec sortSpec as String` |
| SetUserProperty Method        | Sets the value of the specified User Property.                              | `Dim oBusComp as BusComp
oBusComp.SetUserProperty propertyName as String, newValue as String` |
| SetViewMode Method            | Sets the visibility type for the business component.                        | `Dim oBusComp as BusComp
oBusComp.SetViewMode viewMode as Integer` |
| UndoRecord Method             | Reverses any uncommitted changes made to the record.                        | `Dim oBusComp as BusComp
oBusComp.UndoRecord` |
| WriteRecord Method            | Commits to the database any changes made to the current record.             | `Dim oBusComp as BusComp
oBusComp.WriteRecord` |
### Business Component Events for Siebel VB

Table 65 lists a summary of the business component events.

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusComp_Associate Event</td>
<td>Called after a record is added to a business component to create an association.</td>
<td>BusComp_Associate</td>
</tr>
<tr>
<td>BusComp_ChangeRecord Event</td>
<td>Called after the current row changes in the business component.</td>
<td>BusComp_ChangeRecord</td>
</tr>
<tr>
<td>BusComp_CopyRecord Event</td>
<td>Called after a new row is copied in the business component.</td>
<td>BusComp_CopyRecord</td>
</tr>
<tr>
<td>BusComp_DeleteRecord Event</td>
<td>Called after a row is deleted in the business component.</td>
<td>BusComp_DeleteRecord</td>
</tr>
<tr>
<td>BusComp_InvokeMethod Event</td>
<td>Called after a custom or specialized method is called on a business component.</td>
<td>BusComp_InvokeMethod (methodName as String)</td>
</tr>
<tr>
<td>BusComp_NewRecord Event</td>
<td>Called after a new row has been created and made active in the business component.</td>
<td>BusComp_NewRecord</td>
</tr>
<tr>
<td>BusComp_PreAssociate Event</td>
<td>Called before a record is added to a business component to create an association.</td>
<td>BusComp_PreAssociate</td>
</tr>
<tr>
<td>BusComp_PreCopyRecord Event</td>
<td>Called before a new row is copied in the business component.</td>
<td>BusComp_PreCopyRecord</td>
</tr>
<tr>
<td>BusComp_PreDeleteRecord Event</td>
<td>Called before a row is deleted in the business component.</td>
<td>BusComp_PreDeleteRecord</td>
</tr>
<tr>
<td>BusComp_PreGetFieldValue Event</td>
<td>Called when the value of a business component field is accessed.</td>
<td>BusComp_PreGetFieldValue (FieldName as String, FieldValue as String)</td>
</tr>
</tbody>
</table>
Table 65. Business Component Events Summary

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusComp_PreInvokeMethod Event</td>
<td>Called before a specialized or custom method is invoked on a business component.</td>
<td>BusComp_PreInvokeMethod (methodName as String)</td>
</tr>
<tr>
<td>BusComp_PreNewRecord Event</td>
<td>Called before a new row is created in the business component.</td>
<td>BusComp_PreNewRecord</td>
</tr>
<tr>
<td>BusComp_PreQuery Event</td>
<td>Called before query execution.</td>
<td>BusComp_PreQuery</td>
</tr>
<tr>
<td>BusComp_PreSetFieldValue Event</td>
<td>Called when a value is pushed down into the business component from the user interface or through a call to SetFieldValue.</td>
<td>BusComp_PreSetFieldValue (FieldName as String, FieldValue as String)</td>
</tr>
<tr>
<td>BusComp_PreWriteRecord Event</td>
<td>Called before a row is written out to the database.</td>
<td>BusComp_PreWriteRecord</td>
</tr>
<tr>
<td>BusComp_Query Event</td>
<td>Called after the query is complete and every row has been retrieved, but before they have been displayed.</td>
<td>BusComp_Query</td>
</tr>
<tr>
<td>BusComp_SetFieldValue Event</td>
<td>Called after a value has been pushed down into the business component from the user interface or through a call to SetFieldValue.</td>
<td>BusComp_SetFieldValue (fieldName as String)</td>
</tr>
<tr>
<td>BusComp_WriteRecord Event</td>
<td>Called after a row is written to the database.</td>
<td>BusComp_WriteRecord</td>
</tr>
</tbody>
</table>
Business Object Methods for Siebel VB

Table 66 lists a summary of the business object methods’ syntax.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetBusComp Method | Function that returns the specified business component.     | Dim oBusObject as BusObject  
               Dim oBusComp as BusComp  
               set oBusComp = BusObject.GetBusComp(BusCompName as String) |
| Name Method    | Function that returns the name of the business object.     | Dim oBusObject as BusObject  
               Dim sName as String  
               sName = oBusObject.Name |
Table 67. Business Service Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| PropertyExists Method   | Returns a Boolean value indicating whether the property specified in the argument exists. | Dim oService as Service  
                        |                                                                           | iReturn as Boolean  
                        |                                                                           | iReturn = oService.PropertyExists(propName as String) |
| RemoveProperty Method   | Removes a property from a business service.                                 | Dim oService as Service  
                        |                                                                           | oService.RemoveProperty propName as String |
| SetProperty Method       | Assigns a value to a property of a business service.                        | Dim oService as Service  
                        |                                                                           | oService.SetProperty propName as String, propValue as String |

**Business Service Events for Siebel VB**

Table 68 lists a summary of the business service events.

Table 68. Business Service Events Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service_InvokeMethod Event</td>
<td>Called after the InvokeMethod method is called on a business service.</td>
<td>Service_InvokeMethod(methodName as String)</td>
</tr>
<tr>
<td>Service_PreCanInvokeMethod Event</td>
<td>Called before the PreInvokeMethod, allowing the developer to determine whether or not the user has the authority to invoke the business service method.</td>
<td>Service_PreCanInvokeMethod(methodName as String, CanInvoke As String)</td>
</tr>
<tr>
<td>Service_PreInvokeMethod Event</td>
<td>Called before a specialized or user-defined method is invoked on a business service.</td>
<td>Service_PreInvokeMethod(methodName as String, Inputs as PropertySet, Outputs as PropertySet)</td>
</tr>
</tbody>
</table>
Property Set Methods for Siebel VB

Table 69 lists a summary of the property set methods’ syntax.

Table 69. Property Set Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddChild Method</td>
<td>Adds subsidiary property sets to a property set.</td>
<td>Dim oPropSet as PropertySet&lt;br&gt;<code>oPropSet.AddChild childObject as PropertySet</code></td>
</tr>
<tr>
<td>Copy Method</td>
<td>Returns a copy of a property set.</td>
<td>Dim oPropSet1 as PropertySet&lt;br&gt;<code>Dim oPropSet2 as PropertySet&lt;br&gt;</code>set oPropSet2 = oPropSet1.Copy</td>
</tr>
<tr>
<td>GetChild Method</td>
<td>Returns a specified child property set of a property set.</td>
<td>Dim oPropSet as PropertySet&lt;br&gt;<code>Dim childPropSet as SiebelPropertySet&lt;br&gt;</code>set childPropSet = oPropSet.GetChild(index as Long)</td>
</tr>
<tr>
<td>GetChildCount Method</td>
<td>Returns the number of child property sets attached to a parent property set.</td>
<td>Dim oPropSet as PropertySet&lt;br&gt;<code>Dim iCount as Integer&lt;br&gt;</code>iCount = oPropSet.GetChildCount</td>
</tr>
<tr>
<td>GetFirstProperty Method</td>
<td>Returns the name of the first property in a property set.</td>
<td>Dim oPropSet as PropertySet&lt;br&gt;<code>Dim sPropName as String&lt;br&gt;</code>sPropName = oPropSet.GetFirstProperty</td>
</tr>
<tr>
<td>GetNextProperty Method</td>
<td>Returns the name of the next property in a property set.</td>
<td>Dim oPropSet as PropertySet&lt;br&gt;<code>Dim sPropName as String&lt;br&gt;</code>sPropName = oPropSet.GetNextProperty</td>
</tr>
<tr>
<td>GetProperty Method</td>
<td>Returns the value of a property when given the property name.</td>
<td>Dim oPropSet as PropertySet&lt;br&gt;<code>Dim sPropVal as String&lt;br&gt;</code>sPropVal = oPropSet.GetProperty(propName as String)</td>
</tr>
<tr>
<td>GetPropertyCount Method</td>
<td>Returns the number of properties attached to a property set.</td>
<td>Dim oPropSet as PropertySet&lt;br&gt;<code>Dim count as Long&lt;br&gt;</code>count = oPropSet.GetPropertyCount</td>
</tr>
<tr>
<td>GetType Method</td>
<td>Returns the value stored in a type in a property set.</td>
<td>Dim oPropSet as PropertySet&lt;br&gt;<code>Dim sTypeVal as String&lt;br&gt;</code>sTypeVal = oPropSet.GetType</td>
</tr>
<tr>
<td>GetValue Method</td>
<td>Returns a value stored as part of a property set.</td>
<td>Dim oPropSet as PropertySet&lt;br&gt;<code>Dim sValVal as String&lt;br&gt;</code>sValVal = oPropSet.GetValue</td>
</tr>
<tr>
<td>InsertChildAt Method</td>
<td>Inserts a child property set into a parent property set at a specific location.</td>
<td>Dim oPropSet as PropertySet&lt;br&gt;<code>oPropSet.InsertChildAt childObject as SiebelPropertySet, index as Integer</code></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td>Syntax</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| PropertyExists Method | Returns a Boolean value indicating whether the property specified in the argument exists. | Dim oPropSet as PropertySet  
oPropSet.PropertyExists(propName as String) |
| GetPropertyCount Method | Returns the number of properties attached to a property set.  | Dim oPropSet as PropertySet  
dim count as Long  
count = oPropSet.GetPropertyCount |
| RemoveChild Method | Removes a child property set as a specified index from a parent property set. | Dim oPropSet as PropertySet  
oPropSet.RemoveChild(index as Integer) |
| RemoveProperty Method | Removes the property specified in its argument from a property set.  | Dim oPropSet as PropertySet  
oPropSet.RemoveProperty(propName as String) |
| Reset Method | Removes every property and child property set from a property set.  | Dim oPropSet as PropertySet  
oPropSet.Reset |
| SetProperty Method | 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 Method | Assigns a data value to a type member of a property set.  | Dim oPropSet as PropertySet  
oPropSet.SetType(value as String) |
| SetValue Method | Assigns a data value to a value member of a property set.  | Dim oPropSet as PropertySet  
oPropSet.SetValue(value as String) |
Miscellaneous Methods for Siebel VB

Table 70 lists a summary of the miscellaneous method’s syntax.

Table 70. Miscellaneous Method’s Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>TheApplication Method</td>
<td>Global method that returns the unique object of type Application.</td>
<td>TheApplication</td>
</tr>
</tbody>
</table>
This chapter provides information about Browser Scripting and its available events and methods:

- "About Browser Script" on page 399
- "Applet Methods for Browser Script" on page 400
- "Applet Events for Browser Script" on page 401
- "Application Methods for Browser Script" on page 401
- "Application Events for Browser Script" on page 403
- "Business Component Methods for Browser Script" on page 403
- "Business Component Events for Browser Script" on page 405
- "Business Object Methods for Browser Script" on page 405
- "Business Service Methods for Browser Script" on page 406
- "Business Service Events for Browser Script" on page 407
- "Property Set Methods for Browser Script" on page 407
- "Control Methods for Browser Script" on page 409
- "Supported DOM Events for High Interactivity Mode" on page 410
- "Supported DOM Events for Standard Interactivity Mode" on page 411

About Browser Script

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 82 and Table 83 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.
A high interactivity application can contain standard interactivity views (Home Page view and Dashboard view for example). Applet-level browser scripts cannot be used on applets in those views (the same as in standard interactivity applications). Instead the server script WebApplet_ShowControl that is not supported in high interactivity is triggered on the applets for those standard interactivity views.

For information on generating browser scripts, see *Configuring Siebel Business Applications*

### Applet Methods for Browser Script

Table 71 lists a summary of the applet methods’ syntax.

**Table 71. Applet Methods Syntax Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| ActiveMode Method       | Returns a string containing the name of the current Web Template mode. | var oApplet;  
var mode = oApplet.ActiveMode(); |
| BusComp Method          | Returns the business component that is associated with the applet. | var oApplet;  
var busComp = oApplet.BusComp(); |
| BusObject Method        | Returns the business object for the business component for the applet. | var oApplet;  
var oBusObject = oApplet.BusObject(); |
| FindActiveXControl Method | Returns the ActiveX control whose name is specified in the argument. | var oApplet;  
var oControl;  
oControl = oApplet.FindActiveXControl(controlName as String); |
| FindControl Method      | Returns the control whose name is specified in the argument. | var oApplet;  
var oControl;  
oControl = oApplet.FindControl(controlName as String); |
| InvokeMethod Method     | Calls an argument-specified specialized method.            | var oApplet;  
var outPs;  
outPs = oApplet.InvokeMethod(MethodName as String, inputPropSet as PropertySet); |
| Name Method             | Returns the name of the applet.                           | var oApplet;  
var name = oApplet.Name(); |
Applet Events for Browser Script

Table 72 lists a summary of the applet events.

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applet_ChangeFieldValue Event</td>
<td>Called when the user updates a field value in the browser.</td>
<td>Applet_ChangeFieldValue(field, value)</td>
</tr>
<tr>
<td>Applet_ChangeRecord Event</td>
<td>Called when the user moves to a different row or view.</td>
<td>Applet_ChangeRecord()</td>
</tr>
<tr>
<td>Applet_InvokeMethod Event</td>
<td>Called after a specialized method or a user-defined method is invoked.</td>
<td>Applet_InvokeMethod(name, inputPropSet)</td>
</tr>
<tr>
<td>Applet_Load Event</td>
<td>Triggered after an applet has loaded and after data is displayed.</td>
<td>Applet_Load()</td>
</tr>
<tr>
<td>Applet_PreInvokeMethod Event</td>
<td>Called before a specialized method for the Web applet is invoked or a user-defined method is invoked through oWebApplet.InvokeMethod.</td>
<td>Applet_PreInvokeMethod(name, inputPropSet)</td>
</tr>
</tbody>
</table>

Application Methods for Browser Script

Table 73 lists a summary of the application methods’ syntax.

Table 73 does not include methods that are not invoked directly from an Application object instance. For information on methods that are called with InvokeMethod on the Application object, see “InvokeMethod Methods for the Application Object” on page 143.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveApplet Method</td>
<td>Returns the name of the applet that has input focus.</td>
<td>var applet; applet = theApplication().ActiveApplet();</td>
</tr>
<tr>
<td>ActiveBusComp Method</td>
<td>Returns the business component associated with the active applet.</td>
<td>var busComp; busComp = theApplication().ActiveBusComp();</td>
</tr>
<tr>
<td>ActiveBusObject Method</td>
<td>Returns the business object for the business component of the active applet.</td>
<td>var busObject; busObject = theApplication().ActiveBusObject();</td>
</tr>
</tbody>
</table>
### Table 73. Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveViewName Method</td>
<td>Returns the name of the active view.</td>
<td><code>var viewName; viewName = theApplication().ActiveViewName();</code></td>
</tr>
<tr>
<td>FindApplet Method</td>
<td>Returns the applet object identified in the argument.</td>
<td><code>var applet; applet = theApplication().FindApplet (appletName);</code></td>
</tr>
<tr>
<td>GetProfileAttr Method</td>
<td>Returns the value of an attribute in a user profile.</td>
<td><code>var sAttr; sAttr = theApplication().GetProfileAttr(name);</code></td>
</tr>
<tr>
<td>GetService Method</td>
<td>Instantiates and returns a new instance of the service specified in the argument.</td>
<td><code>var svc; svc = theApplication().GetService(serviceName);</code></td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>Calls the named specialized method.</td>
<td><code>var outPs; outPs = theApplication().InvokeMethod(methodName, methodArgs);</code></td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns name of the application.</td>
<td><code>var appName; appName = theApplication().Name();</code></td>
</tr>
<tr>
<td>NewPropertySet Method</td>
<td>Constructs and returns a new property set object.</td>
<td><code>var PropSet; PropSet = theApplication().NewPropertySet();</code></td>
</tr>
<tr>
<td>SetProfileAttr Method</td>
<td>Used in personalization to assign values to attributes in a user profile.</td>
<td><code>theApplication().SetProfileAttr(name, value);</code></td>
</tr>
<tr>
<td>ShowModalDialog Method</td>
<td>Allows you to show a modal dialog box with the cursor maintained in its default state.</td>
<td><code>theApplication().ShowModalDialog(url[, argIn[, options]])</code></td>
</tr>
<tr>
<td>SWEAlert Method</td>
<td>Displays a modal dialog box containing a message to the user.</td>
<td><code>theApplication().SWEAlert(message);</code></td>
</tr>
</tbody>
</table>
Application Events for Browser Script

Table 74 lists a summary of the application events.

Table 74. Application Events Syntax Summary

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application_InvokeMethod Event</td>
<td>Called after a specialized</td>
<td>Application_InvokeMethod(name, inputPropSet)</td>
</tr>
<tr>
<td></td>
<td>method is invoked.</td>
<td></td>
</tr>
<tr>
<td>Application_PreInvokeMethod</td>
<td>Called before a specialized</td>
<td>Application_PreInvokeMethod(name, inputPropSet)</td>
</tr>
<tr>
<td>Event</td>
<td>method is invoked.</td>
<td></td>
</tr>
</tbody>
</table>

Business Component Methods for Browser Script

Table 75 lists a summary of the business component methods’ syntax.

Table 75 does not include methods that are not invoked directly from a Business Component object instance. For information on methods that are called with InvokeMethod on the Business Component object, see “InvokeMethod Methods for the Business Component Object” on page 216.

Table 75. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusObject Method</td>
<td>Returns the business object</td>
<td>var busComp;</td>
</tr>
<tr>
<td></td>
<td>that contains the business</td>
<td>var busObject;</td>
</tr>
<tr>
<td></td>
<td>component.</td>
<td>busObject = busComp.BusObject();</td>
</tr>
<tr>
<td>GetFieldValue Method</td>
<td>Returns a value for the field</td>
<td>var busComp;</td>
</tr>
<tr>
<td></td>
<td>specified in the argument.</td>
<td>var value;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value = busComp.GetFieldValue(field);</td>
</tr>
<tr>
<td>GetFormattedFieldValue</td>
<td>Returns a formatted value for</td>
<td>var busComp;</td>
</tr>
<tr>
<td>Method</td>
<td>the field specified in the</td>
<td>var sValue;</td>
</tr>
<tr>
<td></td>
<td>argument.</td>
<td>sValue = busComp.GetFormattedFieldValue(field);</td>
</tr>
<tr>
<td>GetSearchExpr Method</td>
<td>Returns the current search</td>
<td>var busComp;</td>
</tr>
<tr>
<td></td>
<td>expression.</td>
<td>var sExpr;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sExpr = busComp.GetSearchExpr();</td>
</tr>
<tr>
<td>GetSearchSpec Method</td>
<td>Returns the current search</td>
<td>var busComp;</td>
</tr>
<tr>
<td></td>
<td>specification for the field</td>
<td>var sSpec;</td>
</tr>
<tr>
<td></td>
<td>specified in the argument.</td>
<td>sSpec = busComp.GetSearchSpec(field);</td>
</tr>
</tbody>
</table>
Table 75. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvokeMethod Method</td>
<td>Calls the specialized method named in the argument.</td>
<td><code>var BusComp;</code> <code>var sReturn;</code> <code>sReturn = BusComp.InvokeMethod(methodName, methodArg1, methodArg2,..., methodArgn);</code></td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the business component.</td>
<td><code>var busComp;</code> <code>var sName;</code> <code>sName = busComp.Name();</code></td>
</tr>
<tr>
<td>SetFieldValue Method</td>
<td>Assigns a new value to the named field for the current row of the business</td>
<td><code>var busComp;</code> <code>busComp.SetFieldValue(fieldName, fieldValue);</code></td>
</tr>
<tr>
<td></td>
<td>component.</td>
<td></td>
</tr>
<tr>
<td>SetFormattedFieldValue Method</td>
<td>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.</td>
<td><code>var busComp;</code> <code>busComp.SetFormattedFieldValue(fieldName, fieldValue);</code></td>
</tr>
<tr>
<td>WriteRecord Method</td>
<td>Commits to the database any changes made to the current record.</td>
<td><code>var busComp;</code> <code>busComp.WriteRecord();</code></td>
</tr>
</tbody>
</table>
Business Component Events for Browser Script

Table 76 lists a summary of the business component events.

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusComp_PreSetFieldValue Event</td>
<td>Called when a value is changed by the user in the user interface. This Browser Script event is invoked after the server round trip if the Immediate Post Changes property of the Business Component field is set to TRUE.</td>
<td>BusComp_PreSetFieldValue (fieldName, value)</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE:</strong> This event is not invoked on picklists and multivalue fields.</td>
<td></td>
</tr>
</tbody>
</table>

Business Object Methods for Browser Script

Table 77 lists a summary of the business object methods’ syntax.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetBusComp Method</td>
<td>Returns the specified business component.</td>
<td>var busObject; var busComp; busComp = busObject.GetBusComp(busCompName);</td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the business object.</td>
<td>Var sName; var busObject; sName = busObject.Name();</td>
</tr>
</tbody>
</table>
## Business Service Methods for Browser Script

Table 78 lists a summary of the business service methods’ syntax.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetFirstProperty</td>
<td>Retrieves the name of the first property of a business service.</td>
<td><code>var svc;</code> <code>var sName = svc.GetFirstProperty();</code></td>
</tr>
<tr>
<td>GetNextProperty</td>
<td>After the name of the first property has been retrieved, retrieves the name of the next property of a business service.</td>
<td><code>var svc;</code> <code>var sName = svc.GetNextProperty();</code></td>
</tr>
<tr>
<td>GetProperty</td>
<td>Retrieves the value stored in the specified property.</td>
<td><code>var svc;</code> <code>var value;</code> <code>value = svc.GetProperty(name);</code></td>
</tr>
<tr>
<td>InvokeMethod</td>
<td>Calls a specialized method or a user-created method on the business service.</td>
<td><code>var svc = TheApplication().GetService(&quot;Business Service&quot;);</code> <code>var inputPropSet = TheApplication().NewPropertySet();</code> <code>svc.InvokeMethod(methodName, inputPropSet);</code></td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the business service.</td>
<td><code>var svc;</code> <code>var name;</code> <code>name = svc.Name();</code></td>
</tr>
<tr>
<td>PropertyExists</td>
<td>Returns a Boolean value indicating whether the property specified in the argument exists.</td>
<td><code>var svc;</code> <code>var bool;</code> <code>bool = svc.PropertyExists(name);</code></td>
</tr>
<tr>
<td>RemoveProperty</td>
<td>Removes a property from a business service.</td>
<td><code>var svc;</code> <code>svc.RemoveProperty(name);</code></td>
</tr>
<tr>
<td>SetProperty</td>
<td>Assigns a value to a property of a business service.</td>
<td><code>var svc;</code> <code>svc.SetProperty(name, value);</code></td>
</tr>
</tbody>
</table>
Table 79 lists a summary of the business service events.

Table 79. Business Service Events Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service_InvokeMethod Event</td>
<td>Called when a business service is accessed.</td>
<td>Service_InvokeMethod(methodName, input)</td>
</tr>
<tr>
<td>Service_PreCanInvokeMethod Event</td>
<td>Called before the PreInvokeMethod, allowing the developer to determine whether or not the user has the authority to invoke the business service method.</td>
<td>Service_PreCanInvokeMethod(methodName)</td>
</tr>
<tr>
<td>Service_PreInvokeMethod Event</td>
<td>Called before a specialized method is invoked on a business service.</td>
<td>Service_PreInvokeMethod(methodName, inputPropSet)</td>
</tr>
</tbody>
</table>

Table 80 lists a summary of the property set methods’ syntax.

Table 80. Property Set Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| AddChild Method   | Adds subsidiary property sets to a property set.        | var oPropSet;  
|                   |                                                         | var iIndex;  
|                   |                                                         | iIndex = oPropSet.AddChild(childObject);                               |
| Copy Method       | Returns a copy of a property set.                       | var oPropSet1;  
|                   |                                                         | var oPropSet2;  
|                   |                                                         | oPropSet2 = oPropSet1.Copy();                                         |
| GetChild Method   | Returns a specified child property set of a property set.| var oPropSet;  
|                   |                                                         | var oChildPropSet;  
|                   |                                                         | oChildPropSet = oPropSet.GetChild(index);                             |
| GetChildCount Method | Returns the number of child property sets attached to a parent property set. | var oPropSet;  
|                   |                                                         | var iCount;  
|                   |                                                         | iCount = oPropSet.GetChildCount();                                    |
### Property Set Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetFirstProperty Method</td>
<td>Returns the name of the first property in a property set.</td>
<td><code>var oPropSet;</code>&lt;br&gt;<code>var sPropName;</code>&lt;br&gt;<code>sPropName = oPropSet.GetFirstProperty();</code></td>
</tr>
<tr>
<td>GetNextProperty Method</td>
<td>Returns the name of the next property in a property set.</td>
<td><code>var oPropSet;</code>&lt;br&gt;<code>var sPropName;</code>&lt;br&gt;<code>sPropName = oPropSet.GetNextProperty();</code></td>
</tr>
<tr>
<td>GetProperty Method</td>
<td>Returns the value of a property when given the property name.</td>
<td><code>var oPropSet;</code>&lt;br&gt;<code>var sValue;</code>&lt;br&gt;<code>sValue = oPropSet.GetProperty(propName);</code></td>
</tr>
<tr>
<td>GetPropertyCount Method</td>
<td>Returns the number of properties attached to a property set.</td>
<td><code>var oPropSet;</code>&lt;br&gt;<code>var iCount;</code>&lt;br&gt;<code>iCount = oPropSet.GetPropertyCount();</code></td>
</tr>
<tr>
<td>GetType Method</td>
<td>Returns the value stored in a type in a property set.</td>
<td><code>var oPropSet;</code>&lt;br&gt;<code>var type;</code>&lt;br&gt;<code>type = oPropSet.GetType();</code></td>
</tr>
<tr>
<td>GetValue Method</td>
<td>Returns a value stored as part of a property set.</td>
<td><code>var oPropSet;</code>&lt;br&gt;<code>var sValue;</code>&lt;br&gt;<code>sValue = oPropSet.GetValue();</code></td>
</tr>
<tr>
<td>InsertChildAt Method</td>
<td>Inserts a child property set into a parent property set at a specific location.</td>
<td><code>var oPropSet;</code>&lt;br&gt;<code>oPropSet.InsertChildAt(childObject, index);</code></td>
</tr>
<tr>
<td>PropertyExists Method</td>
<td>Returns a Boolean value indicating whether the property specified in the argument exists.</td>
<td><code>var oPropSet;</code>&lt;br&gt;<code>var bool;</code>&lt;br&gt;<code>bool = oPropSet.PropertyExists(propName);</code></td>
</tr>
<tr>
<td>RemoveChild Method</td>
<td>Removes a child property set as a specified index from a parent property set.</td>
<td><code>var oPropSet;</code>&lt;br&gt;<code>oPropSet.RemoveChild(index);</code></td>
</tr>
<tr>
<td>RemoveProperty Method</td>
<td>Removes the property specified in its argument from a property set.</td>
<td><code>var oPropSet;</code>&lt;br&gt;<code>oPropSet.RemoveProperty(propName);</code></td>
</tr>
<tr>
<td>Reset Method</td>
<td>Removes every property and child property set from a property set.</td>
<td><code>var oPropSet;</code>&lt;br&gt;<code>oPropSet.Reset();</code></td>
</tr>
<tr>
<td>SetProperty Method</td>
<td>Assigns a value to the property of a property set specified in its argument.</td>
<td><code>var oPropSet;</code>&lt;br&gt;<code>oPropSet.SetProperty(propName, propValue);</code></td>
</tr>
</tbody>
</table>
### Control Methods for Browser Script

Table 81 lists a summary of the control methods’ syntax.

#### Table 81. Control Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetType Method</td>
<td>Assigns a data value to a type member of a property set.</td>
<td><code>var oPropSet; oPropSet.SetType(value);</code></td>
</tr>
<tr>
<td>SetValue Method</td>
<td>Assigns a data value to a value member of a property set.</td>
<td><code>var oPropSet; oPropSet.SetValue(value);</code></td>
</tr>
<tr>
<td>Applet Method</td>
<td>Returns the parent applet for the control.</td>
<td><code>var oControl; var oApplet; oApplet = oControl.Applet();</code></td>
</tr>
<tr>
<td>BusComp Method</td>
<td>Returns the corresponding business component for the control.</td>
<td><code>var oControl; var busComp; busComp = oControl.BusComp();</code></td>
</tr>
<tr>
<td>GetProperty Method</td>
<td>Returns the value of the property of a control.</td>
<td><code>var oControl; var propVal; propVal = oControl.GetProperty(propName);</code></td>
</tr>
<tr>
<td>GetValue Method</td>
<td>Returns the value of a control.</td>
<td><code>var oControl; var sValue; sValue = oControl.GetValue();</code></td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the control.</td>
<td><code>var oControl; var sName; sName = oControl.Name();</code></td>
</tr>
<tr>
<td>SetProperty Method</td>
<td>Sets the visual properties of a control.</td>
<td><code>var oControl; oControl.SetProperty(propertyName, propValue);</code></td>
</tr>
<tr>
<td>SetValue Method</td>
<td>Sets the contents of the control to the indicated value.</td>
<td><code>var oControl; oControl.SetValue(value);</code></td>
</tr>
</tbody>
</table>
### Supported DOM Events for High Interactivity Mode

*Table 82* lists the supported Document Object Model (DOM) events for high interactivity mode.

**Table 82. Supported DOM Events for High Interactivity Mode**

<table>
<thead>
<tr>
<th>Control</th>
<th>Siebel Control Type</th>
<th>Supported Events</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button</td>
<td>Native</td>
<td>OnFocus, OnBlur</td>
<td>None</td>
</tr>
<tr>
<td>CheckBox</td>
<td>Native</td>
<td>OnFocus, OnBlur</td>
<td>Rendered as Input Type=CHECKBOX.</td>
</tr>
<tr>
<td>Link</td>
<td>Native</td>
<td>OnFocus, OnBlur</td>
<td>Rendered through paired anchor tags or as INPUT TYPE = TEXT in edit mode.</td>
</tr>
<tr>
<td>List Column</td>
<td>Native</td>
<td></td>
<td>This control does not expose any scriptable events.</td>
</tr>
<tr>
<td>Mailto</td>
<td>Native</td>
<td>OnFocus, OnBlur</td>
<td>Rendered as anchor tags with HREF=mailto or as INPUT TYPE=TEXT in Edit mode.</td>
</tr>
<tr>
<td>MiniButton</td>
<td>Native</td>
<td>OnFocus, OnBlur</td>
<td>None</td>
</tr>
<tr>
<td>Password</td>
<td>Native</td>
<td>OnFocus, OnBlur</td>
<td>Rendered as Input Type = password.</td>
</tr>
<tr>
<td>Text</td>
<td>Native</td>
<td>OnFocus, OnBlur</td>
<td>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.</td>
</tr>
<tr>
<td>TextArea</td>
<td>Native</td>
<td>OnFocus, OnBlur</td>
<td>Rendered as TEXTAREA.</td>
</tr>
<tr>
<td>Tree</td>
<td>Native</td>
<td></td>
<td>Tree applets and controls do not expose any scriptable events.</td>
</tr>
<tr>
<td>URL</td>
<td>Native</td>
<td>OnFocus, OnBlur</td>
<td>Rendered through paired anchor tags with an HREF = underlying field value or as INPUT TYPE = TEXT in edit mode.</td>
</tr>
</tbody>
</table>

---

**Browser Scripting** - Supported DOM Events for High Interactivity Mode

Table 82 lists the supported Document Object Model (DOM) events for high interactivity mode.
NOTE: Siebel objects (business components, applets, and so on) cannot be accessed from DOM events.

Usually in scripting you can call routines in the General section from anywhere in the object. However you cannot call routines written in the General section from the 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 MiniButton Controls."

## Supported DOM Events for Standard Interactivity Mode

Table 83 lists the supported Document Object Model (DOM) events and template modes for standard interactivity mode.

### Table 83. Supported DOM Events and Template Modes for Standard Interactivity Mode

<table>
<thead>
<tr>
<th>Control</th>
<th>Siebel Control Type</th>
<th>Supported Events</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button</td>
<td>Native</td>
<td>OnFocus (Base/Edit) OnBlur (Base/Edit) OnMouseOut (Base/Edit) OnMouseOver (Base/Edit)</td>
<td>None</td>
</tr>
<tr>
<td>CheckBox</td>
<td>Native</td>
<td>OnBlur (Base/Edit) OnFocus (Edit) OnChange (Edit) OnMouseOut (Edit) OnMouseOver(Edit)</td>
<td>In Base mode, a CheckBox appears as a Y or N text value. In Edit mode, a CheckBox is rendered as Input Type=CHECKBOX.</td>
</tr>
<tr>
<td>Link</td>
<td>Native</td>
<td>OnFocus (Base/Edit) OnBlur (Base/Edit) OnMouseOut (Base/Edit) OnMouseOver (Base/Edit) OnClick (Base/Edit)</td>
<td>Rendered through paired anchor tags or as INPUT TYPE = TEXT in Edit mode.</td>
</tr>
<tr>
<td>List Column</td>
<td>Native</td>
<td>List Columns currently do not expose any scriptable events.</td>
<td>None</td>
</tr>
</tbody>
</table>
Table 83. Supported DOM Events and Template Modes for Standard Interactivity Mode

<table>
<thead>
<tr>
<th>Control</th>
<th>Siebel Control Type</th>
<th>Supported Events</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mailto</td>
<td>Native</td>
<td>OnChange (Edit)</td>
<td>Rendered as anchor tags with HREF=mailto or as INPUT TYPE=TEXT in Edit mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnFocus (Base/Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnBlur (Base/Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnMouseOut (Base/Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnMouseOver (Base/Edit)</td>
<td></td>
</tr>
<tr>
<td>MiniButton</td>
<td>Native</td>
<td>OnFocus (Base/Edit)</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnBlur (Base/Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnMouseOut (Base/Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnMouseOver (Base/Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnClick (Base/Edit)</td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td>Native</td>
<td>OnChange (Edit)</td>
<td>In Edit mode, a Password control is rendered as Input type = password.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnFocus (Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnBlur (Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnMouseOut (Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnMouseOver (Edit)</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td>Native</td>
<td>OnChange (Edit)</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnFocus (Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnBlur (Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnMouseOut (Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnMouseOver (Edit)</td>
<td></td>
</tr>
<tr>
<td>TextArea</td>
<td>Native</td>
<td>OnChange (Edit)</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnFocus (Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnBlur (Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnMouseOut (Base/Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnMouseOver (Edit)</td>
<td></td>
</tr>
</tbody>
</table>
At this time, tree applets and controls do not expose any scriptable events.

<table>
<thead>
<tr>
<th>Control</th>
<th>Siebel Control Type</th>
<th>Supported Events</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree</td>
<td>Native</td>
<td>At this time, tree applets and controls do not expose any scriptable events.</td>
<td>None</td>
</tr>
<tr>
<td>URL</td>
<td>Native</td>
<td>OnChange (Edit)</td>
<td>Rendered through paired anchor tags with an HREF = underlying field value or as INPUT TYPE = TEXT in Edit mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnFocus (Base/Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnBlur (Base/Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnMouseOut (Base/Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnMouseOver (Base/Edit)</td>
<td></td>
</tr>
</tbody>
</table>
This chapter provides a quick reference for Siebel eScript methods and events. It has the following topics:

- "Applet Methods for eScript" on page 415
- "WebApplet Events for eScript" on page 416
- "Application Methods for eScript" on page 417
- "Application Events for eScript" on page 419
- "Business Component Methods for eScript" on page 420
- "Business Component Events for eScript" on page 424
- "Business Object Methods for eScript" on page 426
- "Business Service Methods for eScript" on page 426
- "Business Service Events for eScript" on page 427
- "Property Set Methods for eScript" on page 428
- "Miscellaneous Methods for eScript" on page 429

**NOTE:** The ST eScript engine is the default eScript scripting engine in version 8.0. For information on syntax differences between it and the T engine, see *Siebel eScript Language Reference*.

### Applet Methods for eScript

Table 84 lists a summary of the applet methods’ syntax.

Table 84. Applet Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| BusComp Method   | Returns the business component that is associated with the applet. | var applet;
|                  |                                                       | var myBusComp;
|                  |                                                       | myBusComp = applet.BusComp();                                          |
| BusObject Method | Returns the business object for the business component for the applet. | var applet;
|                  |                                                       | var busObject;
|                  |                                                       | busObject = applet.BusObject();                                       |
Table 85. WebApplet Events Summary

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebApplet_InvokeMethod Event</td>
<td>Called after a specialized method or a user-defined method on the Web applet has been executed.</td>
<td>WebApplet_InvokeMethod (MethodName)</td>
</tr>
<tr>
<td>WebApplet_Load Event</td>
<td>Called just after the Web applet is loaded.</td>
<td>WebApplet_Load</td>
</tr>
<tr>
<td>WebApplet_PreCanInvokeMethod Event</td>
<td>Called before the PreInvokeMethod, allowing the developer to determine whether the user has the authority to invoke the applet method.</td>
<td>WebApplet_PreCanInvokeMethod (MethodName, &amp;CanInvoke)</td>
</tr>
<tr>
<td>WebApplet_PreInvokeMethod Event</td>
<td>Called before a specialized method for the Web applet is invoked or a user-defined method is invoked through oWebApplet.InvokeMethod.</td>
<td>WebApplet_PreInvokeMethod (MethodName)</td>
</tr>
</tbody>
</table>
Table 85. WebApplet Events Summary

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebApplet_ShowControl Event</td>
<td>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.</td>
<td>WebApplet_ShowControl(controlName, property, mode, &amp;HTML)</td>
</tr>
<tr>
<td>WebApplet_ShowListColumn Event</td>
<td>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.</td>
<td>WebApplet_ShowListColumn(columnName, property, mode, &amp;HTML)</td>
</tr>
</tbody>
</table>

Table 86 lists a summary of the application methods’ syntax.

Table 86 does not include methods that are not invoked directly from an Application object instance. For information on methods that are called with InvokeMethod on the Application object, see “InvokeMethod Methods for the Application Object” on page 143.

Table 86. Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveBusObject Method</td>
<td>Returns the business object for the business component for the active applet.</td>
<td>var busObject; busObject = TheApplication().ActiveBusObject();</td>
</tr>
<tr>
<td>ActiveViewName Method</td>
<td>Returns the name of the active view.</td>
<td>var sView; sView = TheApplication().ActiveViewName();</td>
</tr>
<tr>
<td>CurrencyCode Method</td>
<td>Returns the three-letter operating currency code.</td>
<td>var sCur; sCur = TheApplication().CurrencyCode();</td>
</tr>
<tr>
<td>GetBusObject Method</td>
<td>Instantiates and returns a new instance of the business object specified in the argument.</td>
<td>var myBusObject; myBusObject = TheApplication().GetBusObject(BusObjectName);</td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the application.</td>
<td>var name; name = TheApplication().Name();</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td>Syntax</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| `GetService Method` | Instantiates and returns a new instance of the service specified in the argument. | `var Service;  
Service = TheApplication().GetService(serviceName);` |
| `GetSharedGlobal  
Method`         | Gets the shared user-defined global variables.                             | `var sName;  
sName = TheApplication().GetSharedGlobal(varName);` |
| `GotoView Method`  | Activates the named view and its business object.                           | `TheApplication().GotoView(viewName[, BusinessObject]);`               |
| `InvokeMethod 
Method` | Calls the named specialized method.                                         | `TheApplication().InvokeMethod(methodName, methodArg1, methodArg2,..., methodArgn);` |
| `LoginId Method`   | Returns the login ID of the user who started the Siebel application.       | `var sID;  
sID = TheApplication().LoginId();`                        |
| `LoginName Method` | Returns the login name of the user who started Oracle’s Siebel application. | `var sUser;  
sUser = TheApplication().LoginName();`                  |
| `NewPropertySet 
Method` | Constructs and returns a new property set object.                           | `var oPropSet;  
oPropSet = TheApplication().NewPropertySet();`     |
| `PositionId Method` | Returns the position ID that describes the user’s current position.         | `var sRow;  
sRow = TheApplication().PositionId();`                    |
| `PositionName 
Method` | Returns the position name of the user’s current position.                   | `var sPosition;  
sPosition = TheApplication().PositionName();`     |
| `RaiseError Method` | Raises a scripting error message to the browser. The error code is a canonical number. | `var keyVal;  
var arg1 ...;  
TheApplication().RaiseError(keyVal, arg1, ...);` |
| `RaiseErrorText 
Method` | Raises a scripting error message to the browser. The error text is the specified literal string. | `var message;  
TheApplication().RaiseErrorText(message);` |
| `SetPositionId 
Method` | Sets the active position to the position ID specified in the argument.       | `var success;  
success = TheApplication().SetPositionId(posId);`   |
Table 86. Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetPositionName Method</td>
<td>Sets the active position to the position name specified in the argument. Returns a Boolean value indicating whether the method succeeded.</td>
<td><code>var success; success = TheApplication().SetPositionName(posName);</code></td>
</tr>
<tr>
<td>SetProfileAttr Method</td>
<td>Used in personalization to assign values to attributes in a user profile.</td>
<td><code>TheApplication().SetProfileAttr(name, value);</code></td>
</tr>
<tr>
<td>SetSharedGlobal Method</td>
<td>Sets a shared user-defined global variable.</td>
<td><code>TheApplication().SetSharedGlobal(varName, value);</code></td>
</tr>
<tr>
<td>Trace Method</td>
<td>Appends a message to the trace file.</td>
<td><code>TheApplication().Trace(message);</code></td>
</tr>
<tr>
<td>TraceOff Method</td>
<td>Turns off the tracing started by TraceOn.</td>
<td><code>TheApplication().TraceOff();</code></td>
</tr>
<tr>
<td>TraceOn Method</td>
<td>Turns tracing on.</td>
<td><code>TheApplication().TraceOn(filename, type, selection);</code></td>
</tr>
</tbody>
</table>

Table 87 lists a summary of the application events.

Table 87. Application Events Syntax Summary

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application_Close Event</td>
<td>Called before the application exits.</td>
<td><code>Application_Close()</code></td>
</tr>
<tr>
<td>Application_InvokeMethod</td>
<td>Called after a specialized method is invoked.</td>
<td><code>Application_InvokeMethod(methodName)</code></td>
</tr>
<tr>
<td>Application_Navigate Event</td>
<td>Called after the client has navigated to a view.</td>
<td><code>Application_Navigate()</code></td>
</tr>
<tr>
<td>Application_PreInvokeMethod</td>
<td>Called before a specialized method is invoked.</td>
<td><code>Application_PreInvokeMethod(methodName)</code></td>
</tr>
<tr>
<td>Application_PreNavigate</td>
<td>Called before the client has navigated from one view to the next.</td>
<td><code>Application_PreNavigate(DestViewName, DestBusObjName)</code></td>
</tr>
<tr>
<td>Application_Start Event</td>
<td>Called when the client starts.</td>
<td><code>Application_Start(commandLine)</code></td>
</tr>
</tbody>
</table>
Business Component Methods for eScript

Table 88 lists a summary of the business component methods’ syntax.

Table 88 does not include methods that are not invoked directly from a Business Component object instance. For information on methods that are called with InvokeMethod on the Business Component object, see “InvokeMethod Methods for the Business Component Object” on page 216.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActivateField Method</td>
<td>Allows queries to retrieve data for the specified field.</td>
<td><code>var myBusComp; myBusComp.ActivateField(fieldName);</code></td>
</tr>
<tr>
<td>ActivateMultipleFields Method</td>
<td>Allows queries to retrieve data for the fields specified in the property set.</td>
<td><code>var myBusComp; myBusComp.ActivateMultipleFields(oPropSet);</code></td>
</tr>
<tr>
<td>Associate Method</td>
<td>Creates a new many-to-many relationship for the parent object through an association business component.</td>
<td><code>var myBusComp; myBusComp.Associate(whereIndicator);</code></td>
</tr>
<tr>
<td>BusObject Method</td>
<td>Returns the business object that contains the business component.</td>
<td><code>var myBusComp; var busObject; busObject = myBusComp.BusObject();</code></td>
</tr>
<tr>
<td>ClearToQuery Method</td>
<td>Clears the current query and sort specifications on the business component.</td>
<td><code>var myBusComp; myBusComp.ClearToQuery();</code></td>
</tr>
<tr>
<td>DeactivateFields Method</td>
<td>Deactivates every currently activated field.</td>
<td><code>var myBusComp; myBusComp.DeactivateFields();</code></td>
</tr>
<tr>
<td>DeleteRecord Method</td>
<td>Removes the current record from the business component.</td>
<td><code>var myBusComp; myBusComp.DeleteRecord();</code></td>
</tr>
<tr>
<td>ExecuteQuery Method</td>
<td>Retrieves a set of BusComp records.</td>
<td><code>var myBusComp; myBusComp.ExecuteQuery(cursorMode);</code></td>
</tr>
<tr>
<td>ExecuteQuery2 Method</td>
<td>Retrieves a set of BusComp records.</td>
<td><code>var myBusComp; myBusComp.ExecuteQuery2(cursorMode, ignoreMaxCursorSize);</code></td>
</tr>
<tr>
<td>FirstRecord Method</td>
<td>Moves to the first record in the business component.</td>
<td><code>var myBusComp; var b1sRecord; b1sRecord = myBusComp.FirstRecord();</code></td>
</tr>
</tbody>
</table>
### Table 88. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FirstSelected Method</strong></td>
<td>Moves to the first record of the multiple selection in the business component.</td>
<td><code>var myBusComp;</code>&lt;br&gt;<code>var bIsMultipleSelection;</code>&lt;br&gt;<code>bIsMultipleSelection = myBusComp.FirstSelected();</code></td>
</tr>
<tr>
<td><strong>GetAssocBusComp Method</strong></td>
<td>Returns the association business component.</td>
<td><code>var myBusComp;</code>&lt;br&gt;<code>var AssocBusComp;</code>&lt;br&gt;<code>AssocBusComp = myBusComp.GetAssocBusComp();</code></td>
</tr>
<tr>
<td><strong>GetFieldValue Method</strong></td>
<td>Returns a value for the field specified in the argument.</td>
<td><code>var myBusComp;</code>&lt;br&gt;<code>var sValue;</code>&lt;br&gt;<code>sValue = myBusComp.GetFieldValue(FieldName);</code></td>
</tr>
<tr>
<td><strong>GetFormattedFieldValue Method</strong></td>
<td>Returns a formatted value for the field specified in the argument.</td>
<td><code>var myBusComp;</code>&lt;br&gt;<code>var sValue;</code>&lt;br&gt;<code>sValue = myBusComp.GetFormattedFieldValue(FieldName);</code></td>
</tr>
<tr>
<td><strong>GetMultipleFieldValues Method</strong></td>
<td>Returns a value for the fields specified in the property set.</td>
<td><code>var myBusComp;</code>&lt;br&gt;<code>myBusComp.GetMultipleFieldValues(oFields, oValues );</code></td>
</tr>
<tr>
<td><strong>GetMVGBusComp Method</strong></td>
<td>Returns the MVG business component associated with the field specified in the argument.</td>
<td><code>var myBusComp;</code>&lt;br&gt;<code>var MvgBusComp;</code>&lt;br&gt;<code>MvgBusComp = myBusComp.GetMVGBusComp(FieldName);</code></td>
</tr>
<tr>
<td><strong>GetNamedSearch Method</strong></td>
<td>Returns the named search specification specified in the argument.</td>
<td><code>var myBusComp;</code>&lt;br&gt;<code>var sValue;</code>&lt;br&gt;<code>sValue = myBusComp.GetNamedSearch(SearchName);</code></td>
</tr>
<tr>
<td><strong>GetPicklistBusComp Method</strong></td>
<td>Returns the pick business component associated with the field specified in the argument.</td>
<td><code>var myBusComp;</code>&lt;br&gt;<code>var pickBusComp;</code>&lt;br&gt;<code>pickBusComp = myBusComp.GetPicklistBusComp(FieldName);</code></td>
</tr>
<tr>
<td><strong>GetSearchExpr Method</strong></td>
<td>Returns the current search expression.</td>
<td><code>var myBusComp;</code>&lt;br&gt;<code>var sExpr;</code>&lt;br&gt;<code>sExpr = myBusComp.GetSearchExpr();</code></td>
</tr>
<tr>
<td><strong>GetSearchSpec Method</strong></td>
<td>Returns the current search specification for the field specified in the argument.</td>
<td><code>var myBusComp;</code>&lt;br&gt;<code>var sSpec;</code>&lt;br&gt;<code>sSpec = myBusComp.GetSearchSpec(FieldName);</code></td>
</tr>
</tbody>
</table>
### Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetSortSpec Method</td>
<td>Returns the active sort specification of the object that has context.</td>
<td><code>var sSortSpec = this.GetSortSpec();</code></td>
</tr>
<tr>
<td>GetUserProperty Method</td>
<td>Returns the value for a property name specified in the argument.</td>
<td><code>var myBusComp; var sValue; sValue = myBusComp.GetUserProperty(propertyName);</code></td>
</tr>
<tr>
<td>GetViewMode Method</td>
<td>Returns the visibility mode for the business component.</td>
<td><code>var myBusComp; var iMode; iMode = myBusComp.GetViewMode();</code></td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>Calls the specialized method named in the argument.</td>
<td><code>var myBusComp; var sReturn; sReturn = myBusComp.InvokeMethod(methodName, methodArg1, methodArg2,..., methodArgn);</code></td>
</tr>
<tr>
<td>LastRecord Method</td>
<td>Moves to the last record in the business component.</td>
<td><code>var myBusComp; var iReturn; iReturn = myBusComp.LastRecord();</code></td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the business component.</td>
<td><code>var myBusComp; var sName; sName = myBusComp.Name();</code></td>
</tr>
<tr>
<td>NewRecord Method</td>
<td>Adds a new record to the business component.</td>
<td><code>var myBusComp; myBusComp.NewRecord(whereIndicator);</code></td>
</tr>
<tr>
<td>NextRecord Method</td>
<td>Moves to the next record in the business component.</td>
<td><code>var myBusComp; var bFound; bFound = myBusComp.NextRecord();</code></td>
</tr>
<tr>
<td>NextSelected Method</td>
<td>Moves to the next record of the current multiple selection.</td>
<td><code>var myBusComp; var iReturn; iReturn = myBusComp.NextSelected();</code></td>
</tr>
<tr>
<td>ParentBusComp Method</td>
<td>Returns the parent business component.</td>
<td><code>var myBusComp; var parentBusComp; parentBusComp = myBusComp.ParentBusComp();</code></td>
</tr>
<tr>
<td>Pick Method</td>
<td>Places the currently selected record in a picklist business component into the appropriate fields of the parent business component.</td>
<td><code>var myBusComp; myBusComp.Pick();</code></td>
</tr>
</tbody>
</table>
### Table 88. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PreviousRecord Method</strong></td>
<td>Moves to the previous record in the business component.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var iReturn;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>iReturn = myBusComp.PreviousRecord();</td>
</tr>
<tr>
<td><strong>RefineQuery Method</strong></td>
<td>Refines a query after a query has been executed.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.RefineQuery();</td>
</tr>
<tr>
<td><strong>SetFieldValue Method</strong></td>
<td>Assigns a new value to the named field for the current row of the business component.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.SetFieldValue(FieldName, FieldValue);</td>
</tr>
<tr>
<td><strong>SetFormattedFieldValue Method</strong></td>
<td>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.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.SetFormattedFieldValue(FieldName, FieldValue);</td>
</tr>
<tr>
<td><strong>SetMultipleFieldValues Method</strong></td>
<td>Assigns a new value to the fields specified in the property set for the current row of the business component.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.SetMultipleFieldValues(oPropSet);</td>
</tr>
<tr>
<td><strong>SetNamedSearch Method</strong></td>
<td>Sets a named search specification on the business component.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.SetNamedSearch(searchName, searchSpec);</td>
</tr>
<tr>
<td><strong>SetSearchExpr Method</strong></td>
<td>Sets the search specification for the business component.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.SetSearchExpr(searchSpec);</td>
</tr>
<tr>
<td><strong>SetSearchSpec Method</strong></td>
<td>Sets the search specification for the specified field.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.SetSearchSpec(FieldName, searchSpec);</td>
</tr>
<tr>
<td><strong>SetSortSpec Method</strong></td>
<td>Sets the sort specification for a query.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.SetSortSpec(sortSpec);</td>
</tr>
<tr>
<td><strong>SetUserProperty Method</strong></td>
<td>Sets the value of the specified User Property.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.SetUserProperty(propertyName, newValue);</td>
</tr>
<tr>
<td><strong>SetViewMode Method</strong></td>
<td>Sets the visibility type for the business component.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.SetViewMode(viewMode);</td>
</tr>
</tbody>
</table>
Business Component Events for eScript

Table 89 lists a summary of the business component events.

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusComp_Associate Event</td>
<td>Called after a record is added to a business component to create an association.</td>
<td>BusComp_Associate()</td>
</tr>
<tr>
<td>BusComp_ChangeRecord Event</td>
<td>Called after the current row changes in the business component.</td>
<td>BusComp_ChangeRecord()</td>
</tr>
<tr>
<td>BusComp_CopyRecord Event</td>
<td>Called after a new row is copied in the business component.</td>
<td>BusComp_CopyRecord()</td>
</tr>
<tr>
<td>BusComp_DeleteRecord Event</td>
<td>Called after a row is deleted in the business component.</td>
<td>BusComp_DeleteRecord()</td>
</tr>
<tr>
<td>BusComp_InvokeMethod Event</td>
<td>Called after a specialized method is invoked in the business component.</td>
<td>BusComp_InvokeMethod(methodName)</td>
</tr>
<tr>
<td>BusComp_NewRecord Event</td>
<td>Called after a new row has been created and made active in the business component.</td>
<td>BusComp_NewRecord()</td>
</tr>
<tr>
<td>BusComp_PreAssociate Event</td>
<td>Called before a record is added to a business component to create an association.</td>
<td>BusComp_PreAssociate()</td>
</tr>
</tbody>
</table>
Table 89. Business Component Events Syntax Summary

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusComp_PreCopyRecord Event</td>
<td>Called before a new row is copied in the business component.</td>
<td>BusComp_PreCopyRecord()</td>
</tr>
<tr>
<td>BusComp_PreDeleteRecord Event</td>
<td>Called before a row is deleted in the business component.</td>
<td>BusComp_PreDeleteRecord()</td>
</tr>
<tr>
<td>BusComp_PreGetFieldValue Event</td>
<td>Called when the value of the business component field is accessed.</td>
<td>BusComp_PreGetFieldValue(FieldName, &amp;FieldValue)</td>
</tr>
<tr>
<td>BusComp_PreInvokeMethod Event</td>
<td>Called before a specialized method is invoked on a business component.</td>
<td>BusComp_PreInvokeMethod(methodName)</td>
</tr>
<tr>
<td>BusComp_PreNewRecord Event</td>
<td>Called before a new row is created in the business component.</td>
<td>BusComp_PreNewRecord()</td>
</tr>
<tr>
<td>BusComp_PreQuery Event</td>
<td>Called before query execution.</td>
<td>BusComp_PreQuery()</td>
</tr>
<tr>
<td>BusComp_PreSetFieldValue Event</td>
<td>Called before a value is pushed down into the business component from the user interface.</td>
<td>BusComp_PreSetFieldValue(FieldName, FieldValue)</td>
</tr>
<tr>
<td>BusComp_PreWriteRecord Event</td>
<td>Called before a row is written out to the database.</td>
<td>BusComp_PreWriteRecord()</td>
</tr>
<tr>
<td>BusComp_Query Event</td>
<td>Called after the query is complete and every row has been retrieved, but before they have been displayed.</td>
<td>BusComp_Query()</td>
</tr>
<tr>
<td>BusComp_SetFieldValue Event</td>
<td>Called after a value has been pushed down into the business component from the user interface.</td>
<td>BusComp_SetFieldValue(FieldName)</td>
</tr>
<tr>
<td>BusComp_WriteRecord Event</td>
<td>Called after a row is written to the database.</td>
<td>BusComp_WriteRecord()</td>
</tr>
</tbody>
</table>
## Business Object Methods for eScript

Table 90 lists a summary of the business object methods’ syntax.

### Table 90. Business Object Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetBusComp Method</td>
<td>Returns the specified business component.</td>
<td><code>var myBusObject; var myBusComp; myBusComp = myBusObject.GetBusComp(BusCompName);</code></td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the business object.</td>
<td><code>var myBusObject as BusObject; var sName; sName = myBusObject.Name();</code></td>
</tr>
</tbody>
</table>

## Business Service Methods for eScript

Table 91 lists a summary of the business service methods’ syntax.

### Table 91. Business Service Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetFirstProperty Method</td>
<td>Retrieves the name of the first property of a business service.</td>
<td><code>var oService; var sName; sName = oService.GetFirstProperty();</code></td>
</tr>
<tr>
<td>GetNextProperty Method</td>
<td>After the name of the first property has been retrieved, retrieves the name of the next property of a business service.</td>
<td><code>var oService; var sName; sName = oService.GetNextProperty();</code></td>
</tr>
<tr>
<td>GetProperty Method</td>
<td>Retrieves the value stored in the specified property.</td>
<td><code>var oService; var sValue; sValue = oService.GetProperty(propName);</code></td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the business service.</td>
<td><code>var oService; var sName; sName = oService.Name();</code></td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>Calls a specialized method or a user-created method on the business service.</td>
<td><code>var oService; oService.InvokeMethod(methodName, InputArguments, OutputArguments);</code></td>
</tr>
<tr>
<td>PropertyExists Method</td>
<td>Returns a Boolean value indicating whether the property specified in the argument exists.</td>
<td><code>var oService; var propExists; propExists = oService.PropertyExists(propName);</code></td>
</tr>
</tbody>
</table>
Business Service Events for eScript

Table 92 lists a summary of the business service events.

Table 92. Business Service Events Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoveProperty Method</td>
<td>Removes a property from a business service.</td>
<td>var oService; oService.RemoveProperty(propName);</td>
</tr>
<tr>
<td>SetProperty Method</td>
<td>Assigns a value to a property of a business service</td>
<td>var oService; oService.SetProperty(propName, propValue);</td>
</tr>
<tr>
<td>Service_InvokeMethod Event</td>
<td>Called after a method is invoked in a business service.</td>
<td>Service_InvokeMethod (methodName)</td>
</tr>
<tr>
<td>Service_PreCanInvokeMethod Event</td>
<td>Called before the PreInvokeMethod, allowing the developer to determine whether or not the user has the authority to invoke the business service method.</td>
<td>Service_PreCanInvokeMethod (MethodName, &amp;CanInvoke)</td>
</tr>
<tr>
<td>Service_PreInvokeMethod Event</td>
<td>Called before a specialized method is invoked on a business service.</td>
<td>Service_PreInvokeMethod (methodName, Inputs, Outputs)</td>
</tr>
</tbody>
</table>

Siebel Object Interfaces Reference Version 8.0, Rev. B
## Property Set Methods for eScript

Table 93 lists a summary of the property set methods’ syntax.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddChild Method</td>
<td>Adds subsidiary property sets to a property set.</td>
<td>var oPropSet;\nvar iIndex;\niIndex = oPropSet.AddChild(childObject);</td>
</tr>
<tr>
<td>Copy Method</td>
<td>Returns a copy of a property set.</td>
<td>var oPropSet1;\nvar oPropSet2;\noPropSet2 = oPropSet1.Copy();</td>
</tr>
<tr>
<td>GetChild Method</td>
<td>Returns a specified child property set of a property set.</td>
<td>var oPropSet;\nvar sPropVal;\nsPropVal = oPropSet.GetChild(index);</td>
</tr>
<tr>
<td>GetChildCount Method</td>
<td>Returns the number of child property sets attached to a parent property set.</td>
<td>var oPropSet;\nvar iCount;\niCount = oPropSet.GetChildCount();</td>
</tr>
<tr>
<td>GetFirstProperty Method</td>
<td>Returns the name of the first property in a property set.</td>
<td>var oPropSet;\nvar sPropName;\nsPropName = oPropSet.GetFirstProperty();</td>
</tr>
<tr>
<td>GetNextProperty Method</td>
<td>Returns the name of the next property in a property set.</td>
<td>var oPropSet;\nvar sPropName\nsPropName = oPropSet.GetNextProperty();</td>
</tr>
<tr>
<td>GetProperty Method</td>
<td>Returns the value of a property when given the property name.</td>
<td>var oPropSet;\nvar sPropVal\nsPropVal = oPropSet.GetProperty(propName);</td>
</tr>
<tr>
<td>GetPropertyCount Method</td>
<td>Returns the number of properties attached to a property set.</td>
<td>var count;\ncount = oPropSet.GetPropertyCount();</td>
</tr>
<tr>
<td>GetType Method</td>
<td>Returns the value stored in a type in a property set.</td>
<td>var oPropSet;\nvar sTypeVal\nsTypeVal = oPropSet.GetType(value);</td>
</tr>
<tr>
<td>GetValue Method</td>
<td>Returns a value stored as part of a property set.</td>
<td>var oPropSet;\nvar sValVal\nsValVal = oPropSet.GetValue(value);</td>
</tr>
<tr>
<td>InsertChildAt Method</td>
<td>Inserts a child property set into a parent property set at a specific location.</td>
<td>var oPropSet;\noPropSet.InsertChildAt(childObject, index);</td>
</tr>
</tbody>
</table>
Table 93. Property Set Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| PropertyExists Method | Returns a Boolean value indicating whether the   | `Dim oService as SiebelService
prop specified in the argument exists.                        `prop Exists = oService.PropertyExists(propName)` |
| RemoveChild Method   | Removes a child property set as a specified      | `var oPropSet; oPropSet.RemoveChild(index);`                          |
|                      | index from a parent property set.                |                                                                        |
| RemoveProperty Method | Removes the property specified in its argument   | `var oPropSet; oPropSet.RemoveProperty(propName);`                    |
|                      | from a property set.                             |                                                                        |
| Reset Method         | Removes every property and child property set   | `var oPropSet; oPropSet.Reset();`                                      |
|                      | from a property set.                             |                                                                        |
| SetProperty Method    | Assigns a value to the property of a property    | `var oPropSet; oPropSet.SetProperty(propName, propValue);`            |
|                      | set specified in its argument.                   |                                                                        |
| SetType Method       | Assigns a data value to a type member of a       | `var oPropSet; oPropSet.SetType(value);`                              |
|                      | property set.                                    |                                                                        |
| SetValue Method      | Assigns a data value to a value member of a      | `var oPropSet; oPropSet.SetValue(value);`                             |
|                      | property set.                                    |                                                                        |

**Miscellaneous Methods for eScript**

Table 94 lists a summary of the miscellaneous method’s syntax.

Table 94. Miscellaneous Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>TheApplication Method</td>
<td>Global method that returns the unique object of type Application.</td>
<td>TheApplication().Application_method;</td>
</tr>
</tbody>
</table>

**Table 94. Miscellaneous Methods Syntax Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This chapter provides a procedure to invoke a custom method with a MiniButton control.

**Invoking Custom Methods with MiniButton Controls**

Be sure to set the appropriate Target Browser Group in Siebel Tools.

To invoke a custom method with a MiniButton control

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. The Controls/Columns window opens with the available controls, including the one you just created.

3. Change the template mode in the Controls/Columns window to 4: Edit List.

4. Drag and drop the ButtonTest control onto an available location. When you release the mouse button, the button appears.

5. Click Save, and then close the Web Layout Editor.

6. Enable the button using one of the following methods:
   - To enable the button declaratively, select the applet in the Object List Editor, expand the Applet object in the Object Explorer, select the Applet User Prop object, and then create a new user property for the applet in the Object List Editor:
     
     ```
     | Name         | Value |
     |--------------|-------|
     | CanInvokeMethod: MyTest | TRUE |
     ```

     For more information on the CanInvokeMethod user property, see *Siebel Developer’s Reference*.

     **NOTE:** The declarative method is strongly recommended for performance reasons.

   - To add a server script to the applet, 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);
}

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":
        theApplication().SWEAlert("Browser script!");
        return("CancelOperation");
        break;
    }
    return("ContinueOperation");
}

Compile the applet object by right-clicking on it and then choosing Compile Selected Objects.

Run any application that has access to accounts, and navigate to My Accounts.

The new button appears.

Click Test.

The Browser Script displays an alert box indicating "Browser Script!"
Index

A

Applet_ChangeFieldValue event, about 103
Applet_ChangeRecord event, about 105
Applet_InvokeMethod event, about 108
Applet_PreInvokeMethod event, about 108
Applet_Load event, about 107
Applet_PreInvokeMethod event, about 108
applet methods
Applet_ChangeFieldValue, about 103
Applet_ChangeRecord, about 105
Applet_InvokeMethod, about 106
Applet_Load, about 107
Applet_InvokeMethod, about 108
object interface events, table of 93
WebApplet_InvokeMethod, about 109
WebApplet_Load, about 107
WebApplet_PreCanInvokeMethod, about 112
WebApplet_PreInvokeMethod, about 113
WebApplet_ShowControl, about 114
WebApplet_ShowListColumn, about 116
application events
about and list of 74
Application_Close event, about 176
Application_InvokeMethod, about 176
Application_Navigate, about 177
Application_PreNavigate, about 179
Application_Start, about 180
PreInvokeMethod, about 177
syntax summary, table of (eScript) 419
table of object interface events 94
application methods
ActiveApplet, about 120
ActiveBusComp, about returning business component associated with 121
ActiveBusObject, about 122
ActiveViewName, about returning name of active view 124
Attach, about 125
CurrencyCode, about 127
Detach, about 128
EnableExceptions, about 129
FindApplet, about 131
GetBusObject, about 131
GetDataSource, InvokeMethod method 143
GetLastErrorCode, about 133
GetLastErrorText, about 134
GetProfileAttr, about 135
GetService, about 135
GetSharedGlobal, about 138
GotoView, about 139
InvokeMethod, about 142
IsViewReadOnly, InvokeMethod method 144
Language, InvokeMethod method 145
LoadObjects, about 146
LoadUserAttributes, about using to load user profile 148
Login. about 148
LoginID, about 150
LoginName, about 151
Logoff, about 152
LookupMessage, about 153
LookupValue, InvokeMethod method 145
Name, about 154
NewPropertySet, about 154
PositionID, about 156
PositionName, about 157
RaiseError, about 158
RaiseErrorText, about 159
SetPositionID, about 161
SetPositionName, about 162
SetProfileAttr, about 162
SetSharedGlobal, about 164
syntax summary (COM data control), table 331
syntax summary (COM data server), table 343
syntax summary, table of (eScript) 417
Trace, about 169
TraceOff, about 171
TraceOn, about 172
application object type
described 38
unique object type, about using to return 322
Application_Close event, about 176
Application_InvokeMethod application event, about 176
Application_Navigate application event, about 177
Application_PreNavigate application event, about 179
Application_Start application event, about 180
applications
application events syntax summary (eScript), table of 419
application methods summary (Siebel VB), table of 383
application methods syntax summary (COM data control), table 331
application methods syntax summary (COM data server), table 343
application methods syntax summary (eScript), table of 417
application methods syntax summary (Mobile Web client), table 355
events summary (Siebel VB), table of 386
methods syntax summary (Browser Script), table of 401
registering business services in Siebel Tools 136
association business component
Associate, about creating many-to-many relationship 186
BusComp_Associate, about calling after record added to create association 257
GetAssocBusComp, returning association business component 200
Attach application method, about 125
B
Browser Script
about 20
applet methods syntax summary, table 400
application methods syntax summary, table 401
business component methods syntax summary, table 403
business object methods syntax summary, table 405
business service events syntax summary, table 407
business service methods syntax summary, table 406
Control methods syntax summary, table 409
property set methods syntax summary, table 407
WebApplet events syntax summary, table 401
Browser, adding to applet 40
BusComp
applet method, about 97
control method, about 294
ExecuteQuery, about return record using method 193
ExecuteQuery2, about returning records using method 195
object interface events, table of 94
BusComp_Associate business component event, about 257
### Index

<table>
<thead>
<tr>
<th>Business Component Event</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusComp_ChangeRecord</td>
<td>258</td>
</tr>
<tr>
<td>BusComp_CopyRecord</td>
<td>259</td>
</tr>
<tr>
<td>BusComp_DeleteRecord</td>
<td>260</td>
</tr>
<tr>
<td>BusComp_InvokeMethod</td>
<td>261</td>
</tr>
<tr>
<td>BusComp_NewRecord</td>
<td>261</td>
</tr>
<tr>
<td>BusComp_PreAssociate</td>
<td>262</td>
</tr>
<tr>
<td>BusComp_PreCopyRecord</td>
<td>262</td>
</tr>
<tr>
<td>BusComp_PreDeleteRecord</td>
<td>263</td>
</tr>
<tr>
<td>BusComp_PreGetFieldVal</td>
<td>264</td>
</tr>
<tr>
<td>BusComp_PreInvokeMethod</td>
<td>265</td>
</tr>
<tr>
<td>BusComp_PreNewRecord</td>
<td>266</td>
</tr>
<tr>
<td>BusComp_PreQuery</td>
<td>266</td>
</tr>
<tr>
<td>BusComp_PreSetFieldVal</td>
<td>267</td>
</tr>
<tr>
<td>BusComp_PreWriteRecord</td>
<td>270</td>
</tr>
<tr>
<td>BusComp_Query</td>
<td>269</td>
</tr>
<tr>
<td>BusComp_SetFieldVal</td>
<td>272</td>
</tr>
<tr>
<td>BusComp_WriteRecord</td>
<td>272</td>
</tr>
</tbody>
</table>

### Business Component Methods

- ActivateField, about 183
- ActivateMultipleFields, about 184
- Associate, about 186
- BusObject, about 188
- ClearLOVCache, InvokeMethod method 216
- ClearToQuery, about 189
- CreateFile, InvokeMethod method 218
- DeactivateFields, about 191
- DeleteRecord, about 193
- ExecuteQuery, about 193
- ExecuteQuery2, about 195
- FirstRecord, about 196
- FirstSelected, about 198
- GenerateProposal, InvokeMethod method 219
- GetAssocBusComp, about 200
- GetFieldValue, about 201
- GetFile, InvokeMethod method 220
- GetFormattedFieldVal, about 203
- GetLastErrCode, about 205
- GetLastErrText, about 206
- GetMultipleFieldValues, about 206
- GetMVGBusComp, about 207
- GetNamedSearch, about 208
- GetPickListBusComp, about 209
- GetSearchExpr, about 211
- GetSearchSpec, about 212
- GetSortSpec, about 212
- GetUserProperty, about 213
- InvokeMethod, about 215
- LastRecord, about 224
- Name, about 225
- NewRecord, about 225
- NextRecord, about 227
- NextSelected, about 228
- ParentBusComp, about 228
- Pick, about 229
- PreviousRecord, about 231
- PutFile, InvokeMethod method 221
- RefineQuery, about 232
- RefreshBusComp, InvokeMethod method 222
- RefreshRecord, InvokeMethod method 223
- Release, about 233
- SetAdminMode, InvokeMethod method 223
- SetFieldValue, about 235
- SetFormattedFieldVal, about 237
- SetMultipleFieldValues, about 238
- SetNamedSearch, about 240
- SetSearchExpr, about 242
- SetSearchSpec, about 244

---

**Syntax Summary, Table of (eScript)** 424
SetSortSpec, about 248
SetUserProperty, about 250
SetViewMode, about 251
syntax summary (COM data control),
table 334
syntax summary (COM data server),
table 346
UndoRecord, about 254
WriteRecord, about 255

business components
about 59
applet, associated with 97
BusComp method, about returning for the
control 294
BusComp object, logical flow of
instantiating 60
business component events summary (Siebel
VB), table of 391
business component events syntax summary
(eScript), table of 424
business component methods syntax
summary (COM data control),
table 334
business component methods syntax
summary (COM data server),
table 346
business component methods syntax
summary (eScript), table of 420
business component methods syntax
summary (Siebel VB), table of 386
business rules, adding to 22
database, committing records to 59
GetBusComp, about returning for a business
component 273
methods for accessing, list of 62
methods syntax summary (Browser Script),
table of 403
methods syntax summary (mobile Web
client), table 358
methods syntax summary, table of
(eScript) 420
name property, returning 225
object type, described 39
records, adding and inserting 59
scenarios 60
SiebelBusComp methods syntax summary
(Java), table of 369

business object methods
GetBusComp, about 273
GetLastError, about 275
GetLastErrorText, about 275
Name, about 276
Release, about 276
syntax summary (COM data control),
table 338
syntax summary (COM data server),
table 350
table of 89

business objects
active applet, about returning for business
component 122
business object methods syntax summary
(COM data control), table 338
business object methods syntax summary
(COM data server), table 350
business object methods syntax summary
(eScript), table of 426
business object methods syntax summary
(Siebel VB), table of 393
DBusObject, about returning business object
for applet 98
DBusObject, about returning business object
that contains business
compontent 188
methods syntax summary (Browser Script),
table of 405
methods syntax summary (Mobile Web
client), table 362
Name, about using to return name of business
object 276
object type, described 38

business rules
business component, adding to 22
described 21

business service events
Service_InvokeMethod, about 288
Service_PreCanInvokeMethod, about 289
Service_PreInvokeMethod, about 290
syntax summary, table of (eScript) 427

business service methods
GetFirstProperty, about 278
GetLastError, about 309
GetLastErrorText, about 310
GetNextProperty, about 280
GetProperty, about 281
InvokeProperty, about 282
Name, about 283
PropertyExists, about 284
Release, about 284
RemoveProperty, about 286
SetProperty, about 287
syntax summary (COM data control),
table 338, 339
syntax summary (COM data server),
table 351
syntax summary, table of (eScript) 426

business service object type, described 39

business services
business service events syntax summary (eScript), table of 427
business service events syntax summary (Siebel VB), table of 394
business service methods syntax summary (COM data control), table 338, 339
business service methods syntax summary (COM data server), table 351
business service methods syntax summary (eScript), table of 426
business service methods syntax summary (Siebel VB), table of 393
events syntax summary (Browser Script), table of 407
methods syntax summary (Browser Script), table of 406
methods syntax summary (mobile Web client), table 363
object interface events, table of 95
object interface methods, table of 89
registering in Siebel Tools 136
retrieving property names 280
SetProperty, about assigning values to members of 287
SiebelService methods syntax summary (Java), table of 373

BusObject
applet method, about 98
business component method, about 188

C
C++
Siebel COM Server, building in 325
Siebel COM Server, testing program 329
CanInvokeMethod applet user property, using instead of PreCanInvokeMethod applet event 112, 431
ChangeFieldValue, about 103
ChangeEvent event, about 105
ClearLOVCache business component method, about 216
ClearToQuery business component method, about 189
coding, caution, about and using Siebel Tools 19
COM data control
application methods syntax summary (table) 331
business component methods syntax summary (table) 334
business object methods syntax summary (table) 338

COM data server
application methods syntax summary (table) 343
business component methods syntax summary (table) 346
business object methods syntax summary (table) 350
business service methods syntax summary (table) 351
installation, about 37
interface method, about COM error handling 77
LoadObjects method, about using to start object and return reference 146
property set methods syntax summary (table) 352

COM error handling, about and methods 77
COM interfaces
Siebel COM client in C++, building 325
Siebel COM client in C++, testing program 329

comparison operators, using in search expressions 245

connect string
about, syntax, and example 74
leveraging load balancing with 76
Siebel Server, substitutions when logging into (table) 75

constants, table of 95
control methods
Applet method, about returning parent applet object 294
BusComp, about 294
GetProperty, about 295
GetValue, about returning control value 295
Name, about returning object name 296
SetProperty, about 297, 299
SetValue, about using to set contents of the control 300
syntax summary, table of (Browser Script), table of 409

controls
FindControl, about argument specified in 99
GetProperty, returning values of control properties 295
GetValue, returning value of control 295
object interface methods, table of 90
setLabelProperty, assigning values to control
properties 297
SetProperty, assigning values to control properties 299
SetValue, using to set the contents of the control 300
Copy property set method, about 304
copying records, using NewRecord method 226
CreateFile business component method, about 218
CurrencyCode application method, about 127
custom methods, invoking with MiniButton controls 431

D
data bean. See Java Data Bean, SiebelDataBean, individual Siebel Java entries 367
data value
SetProperty, about using to assign value to 318
SetType, about using to assign data value of type to property set 319
database, about using WriteRecord to commit to database 255
DeactivateFields business component method, about 191
deallocations, using TraceOn application method to track 172
debug tracings methods, table of 65
DeleteRecord business component method, about 193
Detach application method, about 128

E
EnableExceptions application method, about 129
error code
application methods, about using
GetLastError to return last error code 133
business component methods, about using
GetLastError to return most recent 205
business object methods, about using
GetLastError to return last error code 275
business service methods, about using
GetLastError to return most recent 309
GetError, about using with Java Data Bean to display numeric code 320

error handling
See also individual Siebel object interface entries
COM error handling, about and examples 77
error message tracking 78
native COM error handling, enabling and disabling 129
error messages
business component methods, about using
GetLastError 206
business object methods, about using
GetLastError 275
business service methods, about using
GetLastError 310
function_name Is An Unknown Function, about and correcting 24
GetErrorMessage, about using with Java Data Bean to display message 322
GetLastError, about returning last text error message 134

event method syntax 67
events, object interface events, table of 93
ExecuteQuery business component method, about 193
ExecuteQuery2 business component method, about 195
exposed object types, table of 41
external applications
logging in 148

F
field value, method of returning in the current local format 203
FindActiveXControl applet method, about 98
FindApplet application method, about 131
FindControl applet method, about 99
FirstRecord business component method, about 196
FirstSelected business component method, about 198

G
GenerateProposal business component method, about 219
GetAssocBusComp business component method, about 200
GetBusComp business object method, about 273
GetBusObject application method, about 131
GetByteValue property set method 305
GetChild property set method, about 306
GetChildCount property set method, about 307
GetDataSource application method, about 143
GetErrorCode method, about 320
GetErrorMessage method, about using to display error messages 322
GetFieldValue business component method, about 201
GetFile business component method, about 220
GetFirstProperty business service method, about 278 property set method, about 308
GetFormattedFieldValue business component method, about 203
GetLastErrCode application method, about 133 business component method, about 205 business object method, about 275 business service method, about 309
GetLastErrText application method, about 134 business component method, about 206 business object method, about 275 business service method, about 310 note, about availability to interfaces 26
GetMultipleFieldValues business component method, about 206
GetMVGBusComp business component method, about 207
GetNamedSearch business component method, about 208
GetNextProperty business service method, about 280 property set method, about 310
GetPicklistBusComp business component method, about 209
GetProfileAttr application method, about 135
GetProperty business service method, about 281 control method, about 295 controls, about returning values of properties 295 property set method, about 311
GetPropertyCount property set method, about 312
GetSearchExpr business component method, about 211
GetSearchSpec business component method, about 212
GetService application method, about 135
GetSharedGlobal application method, about 138
GetSortSpec business component method, about 212
GetType property set method 312
GetUserProperty business component method, about 213
GetValue control method, about 295 property set method, about 313
global state, properties and functions 64
global variables about and VB example 67 GetSharedGlobal application method, about 138
GotoView application method, about 139

H
high interactivity mode, about running Browser scripts 399

I
InsertChildAt property set method, about 314
installation procedures, object interfaces 37
interface methods, table grouped by object interface type 81
IsViewReadOnly application method, about 144

J
java Bean. See individual Siebel Java entries
Java Cryptography Extension (JCE), enabling 57
Java Data Bean
  GetErrorcode, about using to display numeric error codes 320
  GetErrorMessage, about using to display error messages 322
  table of SiebelDataBean method syntax 367
JavaScript. See Siebel eScript
JCE (Java Cryptography Extension), enabling 57

L
Language application method, about 145
LastRecord business component method, about 224
load balancing 76
Load event
  Applet_Load, about triggering after applet is loaded 107
  WebApplet_Load event, about triggering just after applet is loaded 110
LoadObjects application method, about 146
LoadUserAttributes application method, about 148
local variables, described and VB example 65
locating objects method, about and list of methods 58, 59
logical operators in search expressions 245
Login method application method, about 148
LoginId application method, about 150
LoginName application method, about 151
Logoff application method, about 152
LookupMessage application method, about 153
LookupValue application method, about 145

M
methods
  custom methods, invoking with MiniButton controls 431
  table grouped by interface type 81
Microsoft Foundation Class (MFC) library. See Siebel COM Data Server
Microsoft Visual Basic
  Siebel COM Data Control Interface, setting up to access 50
  Siebel COM Data Server, setting up to access 48
  Siebel Mobile Web Client Automation Server, setting up to access 46
  Siebel Web Client Automation Server, setting up to access 44
MiniButton controls, using to invoke custom methods 431
Mobile Web client
  application methods syntax summary, table of 355
  business component methods syntax summary, table of 358
  business object methods syntax summary, table of 362
  business service methods syntax summary, table of 363
  property set methods syntax summary, table of 364
module variables, about and VB example 66
MVG business component, returning 207

N
Name
  applet method, about 102
  application method, about 154
  business component method, about 225
  business object method, about 276
  business service method, about 283
  control method, about 296
named field value, about using SetFieldValue to assign new value to 235
navigation methods, object interfaces 64
NewPropertySet application method, about 154
NewRecord business component method, about 225
NextRecord business component method, about 227
NextSelected business component method, about 228

O
object interface events
  See also Siebel object interfaces, events
  applet, table of 93
  application, table of 94
  BusComp, table of 94
  business service, table of 95
object interface methods
  See also Siebel object interfaces, methods
  applet, table of 81
  application, table of 82
  business component, table of 85
  business object, table of 89
  business service, table of 89
  control, table of 90
miscellaneous methods and events, table of 92
property set, table of 91

**object interfaces. See Siebel object interfaces**

**object types**
- applet object type, described 40
- application, described 38
- business component, described 39
- business object, described 38
- business service, described 39
- property set, described 40
- Siebel object interfaces, object types, table of 41

**object, about using Name method to return object name** 296

**operating currency code, returning** 127

**P**

**ParentBusComp business component method, about** 228

**Pick business component method**
- GetPicklistBusComp, returns component 209
- Pick method, about 229

**Pick Method business component method** 229

**PositionId application method, about** 156

**PositionName application method, about** 157

**PreCanInvokeMethod**
- WebApplet_PreCanInvokeMethod, about 112

**PreInvokeMethod**
- Applet_PreInvokeMethod, about 108
- Application_PreInvokeMethod, about 177
- WebApplet_PreInvokeMethod, about 113

**PreviousRecord business component method, about** 231

**programming**
- custom extension routines, about extending data validation 21
- environment, component of 20
- languages, about 19
- user interface components, about customizing behavior 21

**programming with Siebel Object interfaces, about** 31

**properties of controls**
- GetProperty, about returning values 295
- SetLabelProperty, about assigning visual properties 297
- SetProperty, about assigning visual properties 299

**property set methods**
- AddChild, about adding subsidiary property set to 303
- Copy, about returning copy of set 304
- GetByteValue 305
- GetChild, about returning child property of property set 306
- GetChildCount, about returning child property sets attached to 307
- GetFirstProperty, about returning name of first property 308
- GetNextProperty, about returning next property 311
-GetProperty, about returning property value when given name 311
-GetPropertyCount, about returning number of properties attached to 312
-GetValue, about retrieving data value 313
-InsertChildAt, about inserting child property set into parent property 314
- object interface methods, table of 91
- RemoveChild, about removing child property set from parent property set 315
- RemoveProperty, about removing a property from property set 316
- SetByteValue 317
- SetProperty, about assigning a data value to property 318
- SetType, about assigning data value of type 319
- syntax summary (COM data control), table 339
- syntax summary (COM data server), table 352
- syntax summary table (eScript) 428

**property set object type, described** 40

**property sets**
- business service methods syntax summary (COM data control), table 339
- business service methods syntax summary (COM data server), table 352
- Copy, about returning copy of 304
- GetChild, about retrieving child property set 306
- GetFirstProperty, about retrieving property names 308
- GetNextProperty, about retrieving property names 310
- GetProperty, about retrieving property values 311
- GetPropertyCount, about retrieving values of type members 312
- GetType, about retrieving values of type members 312
GetValue, about retrieving values values 313
InsertChildAt, about adding subsidiary 314
methods syntax summary (Browser Script), table of 407
methods syntax summary (eScript), table of 428
methods syntax summary (Mobile Web client), table of 364
methods syntax summary (Siebel VB), table of 395
methods syntax summary (Siebel Web client), table of 379
RemoveChild, about removing child property set 315
RemoveProperty, about removing properties of 316
Reset, about removing properties and child properties 316
SetProperty, about assigning values to members of 318
SetType, about assigning values to type members 319
SetValue, about assigning values to value member 319
SiebelPropertySet methods syntax summary (Java), table of 374
tree-structured data structures, for 303
PropertyExists
business service method, about 284
property set method, about returning Boolean value 314
PutFile business component method, about 221

Q
queries
ClearToQuery, about using to clear query 189
GetSortSpec, using to get sort specification 212
RefineQuery, about using to define after execution 232
SetSortSpec, using to set sort specification 248
quotation marks, about using in search expressions 245

R
RaiseError application method, about 158
RaiseErrorText application method, about 159
records
LastRecord, about using to move to 224
NewRecord, about adding a new record (row) 225
NextSelected, about using to move focus to next record 228
Pick, about placing record in a picklist 229
PreviousRecord, about moving to previous record 231
UndoRecord, about using to reverse uncommitted changes 254
WriteRecord, about committing database changes 255
RefineQuery business component method, about 232
RefreshBusComp business component method, about 222
RefreshRecord business component method, about 223
Release
business component method, about 233
business object method, about 276
business service method, about 284
RemoveChild property set method, about 315
RemoveProperty
business service method, about 286
property set method, about 316
Reset property set method, about removing properties and child property sets 316
run-time engine, invoking 24

S
script tracing 22
search expression
GetSearchExpr, about using to return current search expression 211
SetSearchExpr, about setting on entire search expression 242
search specification
Field name argument, about returning for field specified in 212
searchName, returns named search specification 208
SetNamedSearch, about setting a named search specification on the business component 240
SetSearchSpec, about setting for a particular field 244
SetSearchSpec, about setting for particular field 244
server components, logging events 22
Server Script, components 20
server, about Logoff method 152
| Service_InvokeMethod business service event, about | 288 |
| Service_PreCanInvokeMethod business service event, about | 289 |
| Service_PreInvokeMethod business service event, about | 290 |
| SetAdminMode business component method, about | 223 |
| SetByteValue property set method | 317 |
| SetFieldValue business component method, about | 235 |
| SetFormattedFieldValue business component method, about | 237 |
| SetLabelProperty controls, about setting visual properties | 297 |
| SetMultipleFieldValues business component method, about | 238 |
| SetNamedSearch business component method, about | 240 |
| SetPositionID application method, about | 161 |
| SetPositionName application method, about | 162 |
| SetProfileAttr application method, about | 162 |
| SetProperty business service method, about assigning | 287 |
| controls, about setting visual properties | 299 |
| property set method, about assigning data value to | 318 |
| SetSearchExpr business component method, about | 242 |
| SetSearchSpec business component method, about | 244 |
| SetSharedGlobal application method, about | 164 |
| SetSortSpec business component method, about | 248 |
| SetType property set method, about | 319 |
| SetUserProperty business component method, about | 250 |
| SetValue control, about using to set contents of | 300 |
| property set, about assigning data value to | 319 |
| SetViewMode business component method, about | 251 |
| ShowModalDialog application method, about | 166 |
| Siebel business components, about events and list of | 71 |
| Siebel COM Data Control about and diagram | 32 |

instantiating 50

**Siebel COM Data Server**
- about and diagram 34
- building in C++ 325
- C++, testing program 329
- instantiating 48

**Siebel COM interfaces**
- accessing 44
- COM Data Control interfaces, about and diagram 32
- COM Data Server, about and diagram 34
- COM error handling 77
- Siebel Mobile Web Client Automation Server, about and diagram 35
- Siebel Web Client Automation Server, about and diagram 34

**Siebel Compiler**
- compiler/interpreter described 20
- order considerations and error message 24
- invoking 24

**Siebel constants table** 95

**Siebel eScript**
- about 19
- applet methods, syntax summary (table) 415
- application events syntax summary, table of 419
- application methods syntax summary, table of 417
- business component events syntax summary, table of 424
- business component methods syntax summary, table of 420
- business object methods syntax summary, table of 426
- business service events syntax summary, table of 427
- business service methods syntax summary, table of 426
- naming conventions, about using standardized 28
- property set methods syntax summary, table of 428
- Siebel VB, differences between 27
- ST eScript engine 24
- Switch construct, making effective use of 29
- syntax conventions 43
- theApplication, method syntax summary, table of 429
- this object reference, about using and example 28
- variables, declaring 28
- WebApplet event summary, table of 416
Siebel eScript language, about 20

Siebel extension events
  applet events, about and list of 73
  application events, about and list of 74
  events occur, determining when 71
  method syntax 67
  program flow, process affected by script 68

Siebel business component events, about and list of 71

Siebel object interfaces, getting started
  See also individual Siebel object interface entries
  connect string, about, syntax, and example 74
  connect string, substitutions when logging into a Siebel Server (table) 75
  Siebel COM Data Control, accessing and screen example 50
  Siebel COM interfaces, accessing 44
  Siebel mobile Web client automation server, accessing 46
  Siebel Web Client Automation Server, accessing 44

Siebel object interfaces, methods
  See also individual Siebel object interface entries
  business components, accessing 59
  examples 42
  global state properties and functions 64
  list of 58
  locating objects, about and list of methods 58, 59
  navigation methods 64
  syntax 41
  user interaction, about and methods 64

Siebel programming
  constants, table of 95
  custom extension routines, about extending data validation 21
  environment, components of 20
  user interface components, about customizing behavior 21

Siebel script
  debug tracing methods, table of 65
  global variables, about and VB example 67
  local variables, about and VB example 65
  module variables, about and VB example 66

Siebel Script Editor
  about 20
  Script Assist 20

Siebel Server
  applet, adding to 40
  JDB and Siebel Server, encrypting between 56

Siebel session ID, about returning string containing Id 128

Siebel VB
  about 19
  applet methods syntax summary, table of 381
  application events summary, table of 386
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>application methods syntax summary</td>
<td>383</td>
</tr>
<tr>
<td>business component methods syntax summary</td>
<td>386</td>
</tr>
<tr>
<td>business components events summary</td>
<td>391</td>
</tr>
<tr>
<td>business object methods syntax summary</td>
<td>393</td>
</tr>
<tr>
<td>business service events syntax summary</td>
<td>394</td>
</tr>
<tr>
<td>business service methods syntax summary</td>
<td>393</td>
</tr>
<tr>
<td>components of</td>
<td>20</td>
</tr>
<tr>
<td>date variables, about working with</td>
<td>27</td>
</tr>
<tr>
<td>getting started</td>
<td>25</td>
</tr>
<tr>
<td>Me object reference, about using and</td>
<td>25</td>
</tr>
<tr>
<td>example</td>
<td></td>
</tr>
<tr>
<td>naming conventions, using standardized</td>
<td>25</td>
</tr>
<tr>
<td>objects, destroying and example</td>
<td>27</td>
</tr>
<tr>
<td>picklist, picking a value from</td>
<td>209</td>
</tr>
<tr>
<td>property set methods syntax summary</td>
<td>395</td>
</tr>
<tr>
<td>run-time errors, about trapping</td>
<td>26</td>
</tr>
<tr>
<td>Select Case, making effective use of</td>
<td>26</td>
</tr>
<tr>
<td>Siebel eScript, differences between</td>
<td>27</td>
</tr>
<tr>
<td>syntax conventions</td>
<td>42</td>
</tr>
<tr>
<td>theApplication method, syntax summary</td>
<td>397</td>
</tr>
<tr>
<td>variables, declaring</td>
<td>25</td>
</tr>
<tr>
<td>WebApplet events, summary (table)</td>
<td>382</td>
</tr>
<tr>
<td>With shortcut, using and example</td>
<td>26</td>
</tr>
<tr>
<td><strong>Siebel VB language, about</strong></td>
<td>20</td>
</tr>
<tr>
<td><strong>Siebel Web client</strong></td>
<td></td>
</tr>
<tr>
<td>property set methods syntax summary</td>
<td>379</td>
</tr>
<tr>
<td>Siebel Service methods syntax summary</td>
<td>378</td>
</tr>
<tr>
<td>SiebelHTMLApplication methods syntax</td>
<td>377</td>
</tr>
<tr>
<td><strong>Siebel Web Client Automation Server</strong></td>
<td></td>
</tr>
<tr>
<td>about and diagram</td>
<td>34</td>
</tr>
<tr>
<td>accessing</td>
<td>44</td>
</tr>
<tr>
<td>installation, about</td>
<td>38</td>
</tr>
<tr>
<td><strong>SiebelBusComp methods syntax summary</strong></td>
<td></td>
</tr>
<tr>
<td>(Java), table of</td>
<td>369</td>
</tr>
<tr>
<td><strong>SiebelDataBean methods syntax summary</strong></td>
<td></td>
</tr>
<tr>
<td>(Java), table of</td>
<td>367</td>
</tr>
<tr>
<td><strong>SiebelException methods</strong></td>
<td></td>
</tr>
<tr>
<td>syntax summary (Java), table of</td>
<td>376</td>
</tr>
<tr>
<td><strong>SiebelHTMLApplication methods syntax</strong></td>
<td></td>
</tr>
<tr>
<td>summary, table of</td>
<td>377</td>
</tr>
<tr>
<td><strong>SiebelPropertySet methods syntax summary</strong>(Java), table of</td>
<td>374</td>
</tr>
<tr>
<td><strong>SiebelService methods</strong></td>
<td></td>
</tr>
</tbody>
</table>
### WebApplet events
- summary, table of (eScript) 416
- syntax summary, table of (Browser Script) 401

- **WebApplet_InvokeMethod event**, about 107
- **WebApplet_Load event**, about 110
- **WebApplet_PreCanInvokeMethod event**, about 112
- **WebApplet_PreInvokeMethod event**, about 113
- **WebApplet_ShowControl event**, about 114
- **WebApplet_ShowListColumn event**, about 116
- **WriteRecord business component method**, about 255