Siebel Systems, Inc., 2207 Bridgepointe Parkway, San Mateo, CA 94404
Copyright © 2005 Siebel Systems, Inc.
All rights reserved.
Printed in the United States of America

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

Siebel, the Siebel logo, UAN, Universal Application Network, Siebel CRM OnDemand, TrickleSync, Universal Agent, and other Siebel names referenced herein are trademarks of Siebel Systems, Inc., and may be registered in certain jurisdictions.

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

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 Siebel sales representative.

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

**Proprietary Information**

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

Chapter 1: What’s New in This Release

Chapter 2: Siebel Programming Tools
Components of the Siebel Programming Environment 19
Supported Uses of Siebel Programming Languages 20
  Business Rule Definition 20
  Custom Behavior for User Interface Components 21
Adding New Business Rules to a Business Component 21
The Siebel Script Editor 22
  Using the Siebel Script Editor 23
  Scripted Flag 24
  Script Editing Preferences 25
The Siebel Debugger 25
  Using the Siebel Debugger 26
  Debugging and Run-Time Preferences 27
  Checking Syntax 28
  Breakpoints 29
  Variable Window 30
  Siebel Calls Window 30
Script Tracing 30
Siebel Compiler and Run-Time Engine 32
Getting Started with Siebel VB 32
A Few Notes About Siebel eScript 35

Chapter 3: Programming
Installing Siebel Object Interfaces 40
Siebel Object Interfaces 40
  Siebel COM Interfaces 41
  Siebel Java Interfaces 43
  Built-in Scripting 44
  Usage Evaluation Matrix 44
Chapter 4: Interfaces Reference

Object Interface Methods Tables 87
   Applet Methods 88
   Application Methods 88
   Business Component Methods 90
   Business Object Methods 91
   Business Service Methods 92
   Control Methods 92
   Property Set Methods 93
   Miscellaneous Methods 93

Object Interface Events 94
   Applet Events 94
   Application Events 95
   Business Component Events 95
   Business Service Events 96

Siebel Constants 96

Applet Methods 97
   ActiveMode Method 97
   BusComp Method 98
   BusObject Method 98
   FindActiveXControl Method 99
   FindControl Method 100
   InvokeMethod Method 101
   Name Method 102

Applet Events 103
   Applet_ChangeFieldValue Event 104
   Applet_ChangeRecord Event 105
   Applet_InvokeMethod Event 106
   Applet_Load Event 107
   Applet_PreInvokeMethod Event 109
   WebApplet_InvokeMethod Event 110
   WebApplet_Load Event 111
   WebApplet_PreCanInvokeMethod Event 112
   WebApplet_PreInvokeMethod Event 113
   WebApplet_ShowControl Event 115
   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
GetDataSource Method 133
GetLastErrCode Method 134
GetLastErrText Method 135
GetProfileAttr Method 135
GetService Method 136
GetSharedGlobal Method 138
GotoView Method 140
InvokeMethod Method 142
LoadObjects Method 143
LoadUserAttributes Method 144
Login Method 145
LoginId Method 147
LoginName Method 148
Logoff Method 149
LookupMessage Method 150
LookupValue Method 151
Name Method 151
NewPropertySet Method 152
PositionId Method 154
PositionName Method 154
RaiseError Method 156
RaiseErrorText Method 157
SetPositionId Method 159
SetPositionName Method 160
SetProfileAttr Method 160
SetSharedGlobal Method 162
ShowModalDialog Method 163
SWEAlert Method 165
Trace Method 166
TraceOff Method 168
TraceOn Method 169

Application Events 172
Application_Close Event 173
Application_InvokeMethod Event 173
Application_Navigate Event 174
Application_PreInvokeMethod Event 174
Application_PreNavigate Event 176
Application_Start Event 177

Business Component Methods 179
ActivateField Method 180
ActivateMultipleFields Method 182
Associate Method 183
BusObject Method 185
ClearToQuery Method 186
CountRecords Method 187
DeactivateFields Method 188
DeleteRecord Method 190
ExecuteQuery Method 191
ExecuteQuery2 Method 193
FirstRecord Method 193
FirstSelected Method 196
GetAssocBusComp Method 197
GetFieldValue Method 199
GetFormattedFieldValue Method 201
GetLastErrCode Method 202
GetLastErrText Method 203
GetMultipleFieldValues Method 204
GetMVGBusComp Method 204
GetNamedSearch Method 206
GetPicklistBusComp Method 206
GetSearchExpr Method 208
GetSearchSpec Method 209
GetUserProperty Method 210
getViewMode Method 211
InvokeMethod Method 212
LastRecord Method 218
Name Method 218
NewRecord Method 219
NextRecord Method 220
NextSelected Method 221
ParentBusComp Method 222
Pick Method 223
PreviousRecord Method 224
RefineQuery Method 225
Release Method 226
SetFieldValue Method 228
SetFormattedFieldValue Method 230
Contents

SetMultipleFieldValues Method 232
SetNamedSearch Method 233
SetSearchExpr Method 235
SetSearchSpec Method 237
SetSortSpec Method 241
SetUserProperty Method 243
SetViewMode Method 245
UndoRecord Method 248
WriteRecord Method 248

Business Component Events 250
  BusComp_Associate Event 250
  BusComp_ChangeRecord Event 251
  BusComp_CopyRecord Event 252
  BusComp_DeleteRecord Event 253
  BusComp_InvokeMethod Event 254
  BusComp_NewRecord Event 254
  BusComp_PreAssociate Event 255
  BusComp_PreCopyRecord Event 255
  BusComp_PreDeleteRecord Event 256
  BusComp_PreGetFieldValue Event 257
  BusComp_PreInvokeMethod Event 258
  BusComp_PreNewRecord Event 259
  BusComp_PreQuery Event 259
  BusComp_PreSetFieldValue Event 260
  BusComp_PreWriteRecord Event 262
  BusComp_Query Event 263
  BusComp_SetFieldValue Event 265
  BusComp_WriteRecord Event 265

Business Object Methods 266
  GetBusComp Method 266
  GetLastErrCode Method 267
  GetLastErrText Method 268
  Name Method 269
  Release Method 269

Business Service Methods 270
  GetFirstProperty Method 271
  GetLastErrCode Method 272
  GetLastErrText Method 273
  GetNextProperty Method 273
 GetProperty Method 275
  InvokeMethod Method 276
<table>
<thead>
<tr>
<th>Name Method</th>
<th>PropertyExists Method</th>
<th>Release Method</th>
<th>RemoveProperty Method</th>
<th>SetProperty Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business Service Events</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service_InvokeMethod Event</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service_PreCanInvokeMethod Event</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service_PreInvokeMethod Event</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control Methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Applet Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetProperty Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetValue Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetLabelProperty Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetProperty Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetValue Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Property Set Methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AddChild Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Copy Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetChild Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetChildCount Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetFirstProperty Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetNextProperty Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetProperty Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetType Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetValue Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InsertChildAt Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PropertyExists Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RemoveChild Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RemoveProperty Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetProperty Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetType Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetValue Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miscellaneous Methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetErrorCode Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetErrorMessage Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TheApplication Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 5: Accessing Siebel COM Data Server with C++
Building the Siebel COM Client in C++ 315
Testing Your Program 321

Chapter 6: COM Data Control Quick Reference
Application Methods for COM Data Control 323
Business Component Methods for COM Data Control 326
Business Object Methods for COM Data Control 330
Business Service Methods for COM Data Control 330
Property Set Methods for COM Data Control 331

Chapter 7: COM Data Server Quick Reference
Application Methods for COM Data Server 335
Business Component Methods for COM Data Server 338
Business Object Methods for COM Data Server 342
Business Service Methods for COM Data Server 343
Property Set Methods for COM Data Server 344

Chapter 8: Mobile/Dedicated Web Client Automation Server Quick Reference
Application Methods for Mobile/Dedicated Web Client Automation Server 347
Property Set Methods for Mobile/Dedicated Web Client Automation Server 356

Chapter 9: Siebel Web Client Automation Server Quick Reference
SiebelHTMLApplication Methods for Siebel Web Client Automation Server 359
SiebelService Methods for Siebel Web Client Automation Server 360
PropertySet Methods for Siebel Web Client Automation Server 360
Chapter 10: Java Data Bean Quick Reference
Data Bean Methods for Java Data Bean 363
Business Component Methods for Java Data Bean 365
Business Object Methods for Java Data Bean 368
Business Service Methods for Java Data Bean 369
Property Set Methods for Java Data Bean 370
SiebelException Methods for Java Data Bean 371

Chapter 11: Siebel VB Quick Reference
Applet Methods for Siebel VB 373
Application Methods for Siebel VB 375
Business Component Methods for Siebel VB 378
Business Object Methods for Siebel VB 384
Business Service Methods for Siebel VB 384
Property Set Methods for Siebel VB 386
Miscellaneous Methods for Siebel VB 388

Chapter 12: Browser Scripting
Browser Script Events and Methods 389
Applet Methods for Browser Script 390
Application Methods for Browser Script 391
Business Component Methods for Browser Script 393
Business Object Methods for Browser Script 394
Business Service Methods for Browser Script 395
Property Set Methods for Browser Script 396
Control Methods for Browser Script 398
Supported DOM Events for High Interactivity Mode 399
Supported DOM Events for Standard Interactivity Mode 400

Chapter 13: eScript Quick Reference
Applet Methods for eScript 403
Application Methods for eScript 405
Business Component Methods for eScript 407
What’s New in Siebel Object Interfaces Reference, Version 7.7, Rev. C

Table 1 lists changes in this version of the documentation to support release 7.7 of the software.

Table 1. What’s New in Siebel Object Interfaces Reference Version 7.7, Rev. C

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Connect String&quot; on page 82</td>
<td>The roles of the <em>host</em> and <em>port</em> parameters are clarified. Implementation of Siebel native load balancing through external interfaces is documented in a new section, “Leveraging Load Balancing with the Connect String” on page 84.</td>
</tr>
<tr>
<td>&quot;Application Methods&quot; on page 119</td>
<td>Clarification is provided for:</td>
</tr>
<tr>
<td></td>
<td>■ Standard representations of Application object instances in the various scripting languages</td>
</tr>
<tr>
<td></td>
<td>■ Conventions for representing the Application object instance in the Syntax sections of Application object methods</td>
</tr>
<tr>
<td>&quot;ShowModalDialog Method&quot; on page 163</td>
<td></td>
</tr>
<tr>
<td>&quot;GetFieldVal Method&quot; on page 199</td>
<td>The system Id field is added as a valid argument for this method.</td>
</tr>
<tr>
<td>&quot;Pick Method&quot; on page 223</td>
<td>In recent releases of Siebel Business Applications, this method cannot be used to change the record in a read-only picklist field.</td>
</tr>
<tr>
<td>&quot;SetSearchSpec Method&quot; on page 237</td>
<td>Recommendations are added for calling this method multiple times to set search specifications on a business component.</td>
</tr>
<tr>
<td>&quot;SetViewMode Method&quot; on page 245</td>
<td>Clarification is provided on:</td>
</tr>
<tr>
<td></td>
<td>■ Source of Siebel ViewModes</td>
</tr>
<tr>
<td></td>
<td>■ Definitions of the Siebel ViewMode constants AllView and OrganizationView</td>
</tr>
<tr>
<td>&quot;ActiveBusObject Method&quot; on page 122</td>
<td>Clarified the return value.</td>
</tr>
</tbody>
</table>
Table 2 lists changes in Version 7.7 Rev B of the documentation to support release 7.7 of the software.

Table 2. What’s New in Siebel Object Interfaces Reference Version 7.7 Rev B

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughout document</td>
<td>Deleted documentation for CORBA support, including:</td>
</tr>
<tr>
<td></td>
<td>- CORBA Quick Reference chapter</td>
</tr>
<tr>
<td></td>
<td>- Methods supported by CORBA Object Manager only</td>
</tr>
<tr>
<td></td>
<td>- References to Siebel Application Factory</td>
</tr>
</tbody>
</table>

Table 3 lists changes in Version 7.7 Rev A of the documentation to support release 7.7 of the software.

Table 3. What’s New in Siebel Object Interfaces Reference Version 7.7 Rev A

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ActivateField Method” on page 180</td>
<td>Revised usage information.</td>
</tr>
<tr>
<td>&quot;ActivateMultipleFields Method” on page 182</td>
<td>Added new example code.</td>
</tr>
<tr>
<td>&quot;ActiveBusObject Method” on page 122</td>
<td>Added new example code.</td>
</tr>
<tr>
<td>&quot;AddChild Method” on page 296</td>
<td>Revised usage information.</td>
</tr>
<tr>
<td>&quot;Applet_InvokeMethod Event” on page 106</td>
<td>Replaced examples with more detailed examples.</td>
</tr>
<tr>
<td>&quot;Applet_Load Event” on page 107</td>
<td>Added new example code.</td>
</tr>
<tr>
<td>&quot;Associate Method” on page 183</td>
<td>Added new example code.</td>
</tr>
<tr>
<td>&quot;Attach Method” on page 125</td>
<td>Added new example code.</td>
</tr>
<tr>
<td>&quot;BusComp_PreWriteRecord Event” on page 262</td>
<td>Revised usage information.</td>
</tr>
<tr>
<td>&quot;BusComp_WriteRecord Event” on page 265</td>
<td>Revised usage information.</td>
</tr>
<tr>
<td>&quot;CountRecords Method” on page 187</td>
<td>Added new method.</td>
</tr>
<tr>
<td>&quot;Connect String” on page 82</td>
<td>Revised information on the connect string format.</td>
</tr>
<tr>
<td>&quot;Data Bean Methods for Java Data Bean” on page 363</td>
<td>Revised description information for the Trace methods.</td>
</tr>
</tbody>
</table>
What's New in This Release

- **Siebel Object Interfaces Reference**
  - Version 7.7, Rev. C

- Added new example code.
- Replaced examples with more detailed examples.
- Revised information on returned values.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;DeactivateFields Method&quot; on page 188</td>
<td>Added new example code.</td>
</tr>
<tr>
<td>&quot;EnableExceptions Method&quot; on page 129</td>
<td>Replaced examples with more detailed examples.</td>
</tr>
<tr>
<td>&quot;FindActiveXControl Method&quot; on page 99</td>
<td>Added new example code.</td>
</tr>
<tr>
<td>&quot;FindControl Method&quot; on page 100</td>
<td>Added new example code.</td>
</tr>
<tr>
<td>&quot;FirstRecord Method&quot; on page 193</td>
<td>Replaced examples with more detailed examples.</td>
</tr>
<tr>
<td>&quot;FirstSelected Method&quot; on page 196</td>
<td>Added new example code.</td>
</tr>
<tr>
<td>&quot;GenerateProposal&quot; on page 215</td>
<td>Added new InvokeMethod Method.</td>
</tr>
<tr>
<td>&quot;GetBusComp Method&quot; on page 266</td>
<td>Replaced examples with more detailed examples.</td>
</tr>
<tr>
<td>&quot;GetBusObject Method&quot; on page 131</td>
<td>Replaced examples with more detailed examples.</td>
</tr>
<tr>
<td>&quot;GetErrorCode Method&quot; on page 310</td>
<td>Replaced examples with more detailed examples.</td>
</tr>
<tr>
<td>&quot;GetFieldValue Method&quot; on page 199</td>
<td>Revised information on returned values.</td>
</tr>
<tr>
<td>&quot;GetFirstProperty Method&quot; on page 271</td>
<td>Replaced examples with more detailed examples.</td>
</tr>
<tr>
<td>&quot;GetNextProperty Method&quot; on page 273</td>
<td>Replaced examples with more detailed examples.</td>
</tr>
<tr>
<td>&quot;GetPicklistBusComp Method&quot; on page 206</td>
<td>Added new example code.</td>
</tr>
<tr>
<td>&quot;GetSearchExpr Method&quot; on page 208</td>
<td>Revised usage information.</td>
</tr>
<tr>
<td>&quot;GetService Method&quot; on page 136</td>
<td>Replaced examples with more detailed examples.</td>
</tr>
<tr>
<td>&quot;GetSharedGlobal Method&quot; on page 138</td>
<td>Replaced examples with more detailed examples.</td>
</tr>
<tr>
<td>&quot;GotoView Method&quot; on page 140</td>
<td>Replaced examples with more detailed examples.</td>
</tr>
<tr>
<td>&quot;LoadObjects Method&quot; on page 143</td>
<td>Revised usage information.</td>
</tr>
<tr>
<td>&quot;Login Method&quot; on page 145</td>
<td>Replaced examples with more detailed examples.</td>
</tr>
<tr>
<td>&quot;LookupMessage Method&quot; on page 150</td>
<td>Added new example code.</td>
</tr>
<tr>
<td>&quot;LookupValue Method&quot; on page 151</td>
<td>Added new example code.</td>
</tr>
</tbody>
</table>

Table 3. What’s New in Siebel Object Interfaces Reference Version 7.7 Rev A
What’s New in This Release

Table 3. What’s New in Siebel Object Interfaces Reference Version 7.7 Rev A

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“NewPropertySet Method” on page 152</td>
<td>Replaced examples with more detailed examples.</td>
</tr>
<tr>
<td>“PutFile” on page 216</td>
<td>Revised usage information.</td>
</tr>
<tr>
<td>“RaiseError Method” on page 156</td>
<td>Revised usage information. Added new example code.</td>
</tr>
<tr>
<td>“RaiseErrorText Method” on page 157</td>
<td>Revised usage information. Added new example code.</td>
</tr>
<tr>
<td>“Release Method” on page 226</td>
<td>Added new example code.</td>
</tr>
<tr>
<td>“Release Method” on page 278</td>
<td>Replaced examples with more detailed examples.</td>
</tr>
<tr>
<td>“SetMultipleFieldValues Method” on page 232</td>
<td>Added new example code.</td>
</tr>
<tr>
<td>“SetNameSearch Method” on page 233</td>
<td>Added new example code.</td>
</tr>
<tr>
<td>“SetProfileAttr Method” on page 160</td>
<td>Added new example code.</td>
</tr>
<tr>
<td>“SetSearchExpr Method” on page 235</td>
<td>Revised usage information.</td>
</tr>
<tr>
<td>“SetSharedGlobal Method” on page 162</td>
<td>Added new example code.</td>
</tr>
<tr>
<td>“SetValue Method” on page 293</td>
<td>Revised usage information. Replaced examples with more detailed examples.</td>
</tr>
<tr>
<td>“SiebelException Methods for Java Data Bean” on page 371</td>
<td>Revised file name and path for the Javadoc files.</td>
</tr>
<tr>
<td>“TraceOn Method” on page 169</td>
<td>Added new example code.</td>
</tr>
</tbody>
</table>

Table 4 lists changes in this version of the documentation to support release 7.7 of the software.

Table 4. What’s New in Siebel Object Interfaces Reference Version 7.7

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Applet_ChangeFieldValue Event” on page 104</td>
<td>Revised information on event triggers.</td>
</tr>
<tr>
<td>“BusComp_PreSetFieldValue Event” on page 260</td>
<td>Revised information on event triggers.</td>
</tr>
<tr>
<td>“ClearToQuery Method” on page 186</td>
<td>Revised example script.</td>
</tr>
</tbody>
</table>
Table 4. What’s New in Siebel Object Interfaces Reference Version 7.7

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“ExecuteQuery Method” on page 191</td>
<td>Removed mention of the workaround for bypassing the limit of 10,000 records returned. This workaround could cause the Siebel Object Managers to crash.</td>
</tr>
<tr>
<td>“GetFieldValue Method” on page 199</td>
<td>Revised the information on Return values.</td>
</tr>
<tr>
<td>“GotoView Method” on page 140</td>
<td>Revised information on applicability with standard interactivity.</td>
</tr>
<tr>
<td>“SetProfileAttr Method” on page 160</td>
<td>Added information on performance effects.</td>
</tr>
<tr>
<td>“SetSearchSpec Method” on page 237</td>
<td>Revised example script.</td>
</tr>
</tbody>
</table>
The Siebel applications include two programming languages. Siebel VB is a Visual Basic-like programming environment that includes an editor, debugger, interpreter and compiler. Siebel VB runs on the Windows operating system only. Siebel eScript is, similarly, a JavaScript-like programming environment, which uses the same tools that Siebel VB uses. Siebel eScript runs on both Windows and UNIX operating systems. With these built-in languages, you can extend and configure your Siebel application beyond the capabilities provided by declarative object property definition. The languages are integrated with other Siebel tools, such as the Applet Designer, Siebel CTI, and Siebel SmartScript. Using this integration you can define object properties both with the designer and by attaching scripts.

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

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

For more information, read the *Performance Tuning Guide*.

### Components of the Siebel Programming Environment

The individual components of the Siebel programming environment include:

- **Server Script:**
  - **Siebel VB language.** A programming language that is syntactically and semantically compatible with Microsoft Visual Basic™. Because the language uses most of the same commands and standards as Microsoft Visual Basic, you can extend your Siebel application and reduce training costs.
  - **Siebel eScript language.** A programming language that is syntactically and semantically compatible with Netscape JavaScript™. In parallel with Siebel VB, the language uses most of the same commands and standards as JavaScript, giving you the same advantages in an alternative language. Moreover, you can use Siebel eScript on all Siebel-supported operating systems. Siebel VB is supported on Windows only.
Browser Script: A new type of script in Siebel 7 that executes in and is interpreted by the Browser. Browser Scripts are written in JavaScript and interact with the Document Object Model (DOM) as well as with the Siebel Object Model available in the Browser through the Browser Interaction Manager. A developer can script the behavior of Siebel events as well as the Browser events that are exposed through the DOM. Be aware that the DOMs for Internet Explorer and Netscape Navigator are different. Browser Script may only be used with applications which run in high interactivity mode, except when scripting Control events supported by the Browser Document Object Model.

Siebel Script Editor. An integrated editor used to create, view, edit, and save custom program routines. The Siebel Editor is similar to the code editor that accompanies Microsoft’s Visual Basic program. The Siebel Editor is described in more detail in “The Siebel Script Editor” on page 22.

Siebel Debugger. Assists you in detecting errors contained within Siebel programming language routines. It does not assist in detecting errors outside of the context of custom program routines. The Siebel Debugger can be invoked automatically from Siebel applications when a run-time error occurs if the Siebel application was invoked with the debug option, /H, on the command start-up line. The Debugger can also be invoked from the Debug toolbar and Debug menu. The Debugger is described in more detail in “The Siebel Debugger” on page 25.

Compiler/Interpreter. A nonvisual component of the Siebel programming languages that compiles and executes Siebel custom program routines. It is similar to Microsoft’s Visual Basic Language Interpreter. Siebel language routines are compiled into p-code and stored with the other object definitions in the SRF file.

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

Supported Uses of Siebel Programming Languages

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

- “Business Rule Definition”
- “Custom Behavior for User Interface Components” on page 21

Business Rule Definition

The Siebel programming languages let you extend data validation beyond what is already provided for in the standard Siebel application. The unique validation requirements of a business can be satisfied by custom extension routines that implement the specific business rules prior to performing record manipulation operations, such as record write or record delete.
Data validation routines may incorporate validations based on data from sources within or outside the Siebel application. For example, a validation routine may verify that an opportunity revenue amount is greater than zero if the probability of the opportunity is more than 20 percent using internal Siebel data. Alternatively, an extension routine could verify the availability of a conference room prior to inserting a new activity record by reading the information from another application’s database table.

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

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

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

Custom Behavior for User Interface Components

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

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

Adding New Business Rules to a Business Component

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

To add business rules to a business component

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

The Siebel Script Editor

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

When creating Siebel custom programs, note the following:

■ Check out or lock the project containing the object definitions being modified. If the project is not locked, you are unable to add any text in the Editor window.

■ Choose Debug > Check Syntax to verify the syntax of your Basic or eScript program. The Siebel Compiler reports any syntax errors and indicates the lines where they occur.

■ Choose File > Save when you have finished entering and editing the custom statements to save your work. Closing the Siebel Script Editor without saving your work discards the changes.

■ Before you run the application, you must compile the projects that you have modified and generate a new SRF file. For information on the Object Compiler, read Siebel Developer’s Reference.

■ Run the application with the new application extensions by choosing Debug > Start or clicking the Start button in the Debug toolbar. The Siebel application executes with the new modifications incorporated.

■ You may inadvertently create programming errors that, when encountered, halts the execution of the extension routine. If you started Siebel applications in debug mode (/H option on the command start-up line), a message box opens indicating the nature of the error. You can then return to the Script Editor and choose Debug > Check Syntax. For further details, read “Checking Syntax” on page 28.

■ When a script error is encountered by an end user, or when the Siebel application is not running in Debug mode, the application displays an appropriate error message and returns control back to the point in the standard Siebel code just prior to the error.

See Also

“Using the Siebel Script Editor” on page 23
“Scripted Flag” on page 24
“Script Editing Preferences” on page 25
Using the Siebel Script Editor

To access the Siebel Script Editor (shown in Figure 1), select an object definition in the Object List Editor and click the right mouse button. If the editor is available from that object type, you can select Edit Scripts from the dialog box. Siebel scripts can be attached to the object types application, applet, business component, and business service.

![Siebel Script Editor](image)

Figure 1. Siebel Script Editor

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

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

- Cut, copy, and paste the text from one location to another location within or from outside the Editor. When pasting into the Editor, avoid having two code blocks with the same name by placing the code between the function `<Name> ()` (eScript) or Sub `<Name> / End Sub` block (VB).
- Import and export Siebel scripts.
- Associate a given Siebel script with a predefined object event, such as a PreSetFieldValue event for a Business Component.
- Debug a custom routine by invoking the Siebel Debugger.
- Compile a custom routine by invoking the Siebel Compiler from the Siebel Script Editor.

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

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

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

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

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

■ Remove extraneous comments.
■ Indent code using tabs instead of spaces.
■ Use short variable and method names.
■ Create subroutines and functions.

**Scripted Flag**

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

To access the script editing preferences, choose View > Options, and then click the Scripting tab (see Figure 2).

![Figure 2. Script Editing Preferences](image)

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

- **Font field.** Used to select the font for display of scripts.
- **Tab Width field.** Defines the number of spaces for a tab character. The default is four spaces.
- **Auto Indent field.** When checked, each succeeding line is indented to the position set by the current line.
- **Scripting Language.** Used to specify the default scripting language for server side components.
- **Compiling.** Used to specify the folder where \bscripts\all resides. This is where Browser Scripts are generated. For pure Web client, set this to siebsrvr\webmaster. For mobile Web client, set this to <client_root>\PUBLIC\<language_code>. For example, if you are using the mobile Web client, then you might set this to D:\sea770\client\PUBLIC\enu, then the Browser Script files are generated to D:\sea770\client\PUBLIC\enu\genbscript time stamped folder>\bscripts\all.

**The Siebel Debugger**

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

With the Siebel Debugger you can do the following:

- Set and clear breakpoints in your Siebel script. A breakpoint is a marker on a line of code that tells the interpreter to suspend execution at that line so that the state of the program can be examined using the Debugger.

- Step over a line of code. If the current line is a call to a subroutine or function, the Debugger stops at the next line in the current procedure (skipping the subroutine).

- Step into a subroutine of custom routine code. Step Into is used to execute one line of code in the Debugger. If the current line is a call to a subroutine or function, the Debugger stops at the first line of that function. Otherwise, the Debugger stops at the next line of the current procedure.

- View the value of custom routine variables. The Siebel Debugger includes a window in which variables and their values are displayed. This window can be used to monitor the values of specific variables as the custom routine executes.

- View the stack.

- View the Application Level Variables. This shows the instantiated business objects and business components. This also shows the business objects and business components that are available and the fields that are available.

- Debug hierarchical structures such as a propertyset.

See Also
"Using the Siebel Debugger"
"Debugging and Run-Time Preferences" on page 27
"Checking Syntax" on page 28
"Breakpoints" on page 29
"Variable Window" on page 30
"Siebel Calls Window” on page 30

Using the Siebel Debugger

You can access the Debugger in several ways:

- You can set breakpoints in the current routine and begin execution by clicking the Start button. Execution is suspended when one of the lines that contains a breakpoint is about to be executed. The Debugger is activated and it highlights the line containing the breakpoint.

- If an executing program encounters a run-time error, such as an unhandled Siebel VB or eScript error, execution is suspended, the Debugger is activated, and it highlights the line containing the error.

Debug options are available from the Debug title bar menu and the Debug toolbar.
Debugging and Run-Time Preferences

To access the debugging preferences, select View > Options and then click the Scripting tab (see Figure 3).

This dialog box has the following debugger settings:

- **Adjust breakpoint to next valid line.** When breakpoints are deleted on invalid lines, this option creates a breakpoint at the next valid line.

- **Make debugger window active when debugging.** The Siebel Debugger window appears whenever you are in debug mode.

- **Always enter the debugger when an error occurs.** The Siebel Debugger window appears whenever a script error occurs.
To access the run-time preferences, select Views > Options and then click the Debug tab (see Figure 4).

![Run-Time Preferences Dialog Box](image)

Figure 4. Run-Time Preferences

This dialog box has the following preference settings:

- **Executable.** The name of the Siebel Web Client executable (Siebel.exe).
- **CFG file.** Configuration file to be used by the client.
- **Browser.** The path to the browser executable.
- **Working directory.** Siebel root directory (location of DLLs).
- **Arguments.** Additional line options for starting the application. Common arguments are:
  - `/h` - to enable local debugging of Server scripts
  - `/s <filename>` - to enable SQL spooling
- **User name.** Login name of the user.
- **Password.** Password of the user name.
- **Data source.** Default data source. Values listed depend upon the configuration file specified in the CFG file field.

### Checking Syntax

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

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

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

2. Click Go to Line.

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

3. Correct the code and check the syntax again.

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

4. Repeat Step 2 and Step 3 until you see no more messages.

5. Choose File > Save to save your file, and close the Siebel application.

6. Press F7 to compile the SRF file.

7. When the compilation finishes, click Run or press F5 to restart the application.

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

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

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

Breakpoints

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

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

- When debugging, clicking on a line of Siebel code toggles a breakpoint on that line.
Variable Window

The variable window displays the contents of the variables associated with a Siebel script when debugging. It also shows business components, field values, and heretical structures such as property sets.

Siebel Calls Window

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

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

Script Tracing

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

To enable logging

1. Navigate to Server 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:
   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. Example: sadmin, mmasters, hkim, cconnors</td>
</tr>
</tbody>
</table>

The following is a sample trace.

```
2021 03 04 09 15:37:20 2003-04-09 16:40:52 -0700 00000022 001 001f 0001 09 SCCObj_Mgr_enu_47126.log 7.5.3 [16122] ENU
Obj MgrSessionInfoObj MgrLogin32003-04-09 15:37:20 Login name : SADMIN
Obj MgrSessionInfoObj MgrAuth32003-04-09 15:37:20 Authentication name : SADMIN
Obj MgrSessionInfoObj MgrLogin32003-04-09 15:37:20 Session Type: Regular Session
GenericLogGenericError12003-04-09 15:37:20 Invocation of Applet Menu New Service::NewExpense is not allowed.
Obj MgrSessionInfoObj MgrLogin32003-04-09 16:40:52 Username: SADMIN, Login Status: Attempt, Session Id: 11.690.b016.3e94a0a0, IP Address: 172.20.94.66
```

Script tracing is not the same as file-based tracing. For more information on file-based tracing, read “Trace Method” on page 166.
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.

Getting Started with 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 here.

<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>
</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. You can see other examples of `Me` in “ParentBusComp Method” on page 222, “SetViewMode Method” on page 245, “BusComp_PreQuery Event” on page 259, “BusComp_PreWriteRecord Event” on page 262, and “ActiveMode Method” on page 97.

```vba
Function BusComp_PreSetFieldValue(FieldName As String, FieldValue As String) As Integer
    If Val(Me.GetFieldValue("Rep %")) > 75 Then
        TheApplication.RaiseErrorText("You can set the Rep% to greater than 75")
        BusComp_PreSetFieldValue = CancelOperation
    Else
        BusComp_PreSetFieldValue = ContinueOperation
    End If
End Function
```

**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.

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

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

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

**Make effective use of the Select Case construct.** The Select Case construct chooses among any number of alternatives you require, based on the value of a single variable. This is greatly preferable to a series of nested `If` statements, because it simplifies code maintenance and also improves performance because the variable must be evaluated only once.
Use the With shortcut. Use the With statement to apply several methods to a single object. It makes the code easier to read, reduces typing, and improves performance. Instead of a series of statements such as:

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

use the following:

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

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

**To perform date arithmetic**

1. Retrieve the value of date fields with the GetFieldValue method. For more information, read “GetFieldValue Method” on page 199.
2. Convert it into a date variable using the DateSerial() function.
3. Perform the required date arithmetic.

The following example is in Siebel VB.
Dim strDate as String, varDate as Variant
strDate = oBC.GetFieldValue("Date Field")
varDate = DateSerial(Val(Mid(strDate,7,4)),Val(Left(strDate,2)),Val(Mid(strDate,4,2)))

[any date arithmetic]

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

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

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

[Your code here]

Set oBusComp = Nothing
Set oBusObj = Nothing
```

### A Few Notes About Siebel eScript

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

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

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

There are also some important differences between Siebel eScript and standard ECMAscript. Most important, Siebel eScript has no user interface functions. It cannot, therefore, be used to animate or control Web pages. Second, it contains two objects that are not part of standard ECMAscript: SELib and Clib. These objects implement a variety of C-like functions for interacting with the operating and file systems, and for file I/O. For details on these and other eScript functions not covered here, read *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. However, Siebel eScript requires you to declare variables with the `var` keyword. As a general rule, declare the variables used in a module before you use them. Doing so makes it easier for others to understand your code, and for you to debug the code. There is one notable exception to this standard. Declaring a variable inside a loop controller restricts the scope of that reference to within the loop. This prevents the variable from persisting; it can therefore be declared again inside another loop.

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

```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 makes the code easier to read, reduces typing, and improves performance. Instead of a series of statements such as:

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

use the following:

```javascript
var oBusObject = TheApplication().GetBusObject("Opportunity");
var oBusComp = oBusObject.GetBusComp("Opportunity");
with (oBusComp)
{
  ClearToQuery();
  ActivateField("Name");
  ActivateField("Quote Number");
  ActivateField("Account");
  SetSearchSpec("Name", varname);
  ExecuteQuery(ForwardOnly)
  if (FirstRecord())
  {
    var sQuoteNum = GetFieldValue("Quote Number");
  }
```
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.SetFieldVal("Sales Stage Date",sysdatestring);
        if (((FieldVal) == "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", FieldVal);
        }
        break;
    }
}
```

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

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

```javascript
var oBusObject = TheApplication().GetBusObject("Contact")
var oBusComp = oBusObject.GetBusComp("Contact")
[ Your code here ]
oBusComp = null;
oBusObject = null;
```
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.

Siebel Systems does not support the following:

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

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

This chapter describes the interface environments and object types. Chapter 4, "Interfaces Reference" describes the supported methods of the Siebel object interfaces and provides examples of how you can use them.
Installing Siebel Object Interfaces

Table 5 lists the installation procedure for each object interface.

### Table 5. 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. For more information, read the Siebel Installation Guide 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. COM Data Control is also installed with the OLE DB Provider and BizTalk Connector. For more information, read the Siebel Installation Guide for the operating system you are using.</td>
</tr>
<tr>
<td>COM Data Server</td>
<td>Installed by default with the Mobile/Dedicated Web Client.</td>
</tr>
<tr>
<td>Siebel Web Client Automation Server</td>
<td>Installed by default with the Siebel Mobile/Dedicated Web Client. Also installed by default with the Siebel Enterprise Server Installer.</td>
</tr>
</tbody>
</table>

### Siebel Object Interfaces

Siebel object interfaces provide:

- "Siebel COM Interfaces" on page 41
- "Siebel Java Interfaces" on page 43
- Built-in scripting of Siebel objects using Siebel VB, Siebel eScript, and Browser Script. For more information, read "Built-in Scripting" on page 44.

**See Also**

"Usage Evaluation Matrix” on page 44
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 applications or components that use Siebel COM Data Control connects and communicates with Siebel Application Object Manager. The Siebel Application Object Manager, which could be running on a remote Siebel Server, is a multi-threaded, multiprocess application server that hosts Siebel business objects and supports session-based connections by clients. **Figure 5** shows how external applications use *Siebel COM Data Control* to communicate with the Siebel application.

![Figure 5. Siebel COM Data Control](image-url)
**COM Data Server**

Figure 6 shows how external applications use Siebel COM Data Server without having to access the user interface objects.

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

**Siebel Web Client Automation Server**

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

To enable the Web Client Automation Server, make sure that the `EnableWebClientAutomation` parameter is set to TRUE in the [SWE] section of the application’s configuration file. With this parameter set to TRUE, a small ActiveX Control downloads to the desktop and the SiebelHTMLApplication process starts. This process terminates when the Siebel Web Client is gracefully terminated. You may need to modify the ActiveX controls and plug-ins security settings in the Browser to use the Web Client Automation Server.
Figure 7 shows how external applications can invoke business services and manipulate property sets in the Siebel Web Client Automation Server.

**Siebel Mobile/Dedicated Web Client Automation Server**
When accessing the Mobile/Dedicated Web Client Automation Server, Siebel Mobile Web Client must be running. Figure 8 shows how the Siebel Mobile/Dedicated Web Client Automation Server is used by external applications to control the Siebel application.

**Siebel Java Interfaces**
The Siebel Java Data Bean allows external applications to access Siebel objects without having to display the Siebel user interface. These objects are made available through the Siebel Java Data Bean, which can be used by an external application, component, or Java applet. The Java Data Bean provides functional access to the Siebel applications for both reading and writing data. The set of interfaces exposed through this interface is similar to that exposed by the Siebel COM Data Control.
Any external application that uses the Siebel Java Data Bean connects and communicates with a Siebel Application Object Manager. The Siebel Application Object Manager, which could be running on a remote Siebel Server, is a multithreaded, multiprocess application server that hosts Siebel objects and supports session-based connections by clients. The Siebel Application Object Manager specified in the connect string must be running on the specified Siebel Server.

**Using the Siebel Java Data Bean with Multiple Threads**

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

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

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

**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 6 to determine which types of Siebel object interface to use.

<table>
<thead>
<tr>
<th>Usage</th>
<th>Web Client Automation Server</th>
<th>Mobile/Dedicated 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></td>
<td>X</td>
<td>X</td>
<td>X</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 45
- “Business Object Object Type” on page 46
- “Business Component Object Type” on page 46
- “Business Service Object Type” on page 46
- “Applet Object Type” on page 47
- “Property Set Object Type” on page 47
- “User Interface Control Object Type” on page 47

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

**NOTE:** Interfaces may be subject to change.

### Application Object Type

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

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

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

Business Component Object Type

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

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

Business Service Object Type

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

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

There are two types of built-in business services:

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

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

Built-in services cannot be modified at runtime, and they cannot be overridden by run-time scripts.
User-created services can be created by adding a new record to the Business Service list applet in Siebel Tools. They can also be defined by administrators at run time by using views in the Siebel client. They can have whatever properties are needed to accomplish a particular task. They can be called either from scripts or from object interfaces.

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

### Applet Object Type

Because applet objects are part of the user interface, they are not accessible when using the Siebel object interfaces through the Siebel COM Data Server, Siebel COM Data Control, Siebel Web Client Automation Server, Siebel Mobile/Dedicated Web Client Automation Server, and 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/Dedicated 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>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Applet</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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></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></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></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>Argument</td>
<td>Description</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:

- **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 the COM Data Server, an additional argument, `errCode`, is always required as the last argument.

**Usage Syntax**

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

- **ObjectReference** is a variable name of type `ObjectType` that points to the object on which the method is invoked.
  
  **NOTE:** You do not need to explicitly specify an `ObjectReference` when you invoke a method on an object inside its event handler.

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

**Siebel VB**

If there is a return value,

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

If there are no arguments,

```
returnValue = ObjectReference.MethodName
```
If there is no return value,

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

Examples
acctName = acctBC.GetFieldValue("Name")
acctBC.SetViewMode AllView

Siebel eScript
If there is a return value,

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

If there are no arguments,

    returnValue = ObjectReference.MethodName();

If there is no return value,

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

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

Using 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, COM Controls, or ORBs:
- "Accessing Siebel COM Interfaces” on page 51
- "Accessing the Siebel Web Client Automation Server” on page 51
- "Accessing the Siebel Mobile/Dedicated Web Client Automation Server” on page 53
- "Instantiating the Siebel COM Data Server” on page 55
- “Instantiating the Siebel COM Data Control” on page 57
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 9 displays an example of an object browser in Microsoft Visual Basic 5.0.

![Object Browser](image)

Figure 9. Determining Correct COM Syntax in Microsoft Visual Basic

Accessing the Siebel Web Client Automation Server

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

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

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

```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
        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
    End If

    Set siebApp = Nothing
End Sub
```
Accessing the Siebel Mobile/Dedicated Web Client Automation Server

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

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

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

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

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

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

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

```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")
If Not siebBusObj Is Nothing Then
  ' Create a Business Component
  Set siebBusComp = siebBusObj.GetBusComp("Contact")
Else
  errCode = siebWebApp.GetLastErrCode
  errText = siebWebApp.GetLastErrText
  TheApplication().RaiseErrorText("Business Object Creation failed: " & errCode & ":" & errText);
End If

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

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

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

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

Set siebWebApp = Nothing
End Sub
Instantiating the Siebel COM Data Server

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

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

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

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

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

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

**NOTE:** This code must be written and executed outside of Siebel Tools, for example in Microsoft Visual Basic.
Private Sub Command1_Click()
'Siebel Application Object
Dim siebApp As SiebelApplication
Dim siebBusObj As SiebelBusObject
Dim siebBusComp As SiebelBusComp
Dim siebSvcs As SiebelService
Dim siebPropSet As SiebelPropertySet
Dim bool As Boolean
Dim errorCode 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, errorCode
  If errorCode = 0 Then
    'Log Into the Siebel Server
    siebApp.Login "username", "password", errorCode
    If errorCode = 0 Then
      'Create A Business Object
      Set siebBusObj = siebApp.GetBusObject("Contact", errorCode)
      If errorCode = 0 Then
        'Create a Business Component
        Set siebBusComp = siebBusObj.GetBusComp("Contact")
      Else
        errText = siebApp.GetLastErrText
        TheApplication().RaiseErrorText("Business Object Creation failed: " & errorCode & ":") & errText);
      End If
    Else
      errText = siebApp.GetLastErrText
      TheApplication().RaiseErrorText("Business Object Creation failed: " & errorCode & ":") & errText);
    End If
  Else
    errText = siebApp.GetLastErrText
    TheApplication().RaiseErrorText("Property Set Creation failed: " & errorCode & ":") & errText);
  End If
  'Create A New Property Set
  Set siebPropSet = siebApp.NewPropertySet(errCode)
  If errorCode = 0 Then
    Set siebPropSet = Nothing
  Else
    errText = siebApp.GetLastErrText
    TheApplication().RaiseErrorText("Property Set Creation failed: " & errorCode & ":") & errText);
  End If
  'Get A Siebel Service
  Set siebSvcs = siebApp.GetService("Workflow Process Manager", errCode)
  If Not siebSvcs Is Nothing Then
    Set siebSvcs = Nothing
  Else
errText = siebApp.GetLastErrText
TheApplication().RaiseErrorText("Could not Get Siebel Service: " & errCode & ":" & errText);
End If
If Not siebBusComp Is Nothing Then
    Set siebBusComp = Nothing
End If
If Not siebBusObj Is Nothing Then
    Set siebBusObj = Nothing
End If
Else
    errText = siebApp.GetLastErrText
    TheApplication().RaiseErrorText("Login Failed: " & errCode & ":" & errText);
End If
Else
    errText = siebApp.GetLastErrText
    TheApplication().RaiseErrorText("Load Objects Failed: " & errCode & ":" & errText);
End If
Set siebApp = Nothing
End If
End Sub

Instantiating the Siebel COM Data Control

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

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

1. Be sure you have installed the Siebel COM Data Control. Read “Installing Siebel Object Interfaces” on page 40.
2. Start Microsoft Visual Basic.
3. Select Standard EXE.
4. Choose Project > References.
5 In the list box, highlight (but do not check) Siebel BusObject Interfaces Type Library. Near the bottom of the dialog box, note the directory in which the file \sstchca.dll\ resides, as shown in the following illustration.

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

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

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

    Sub CreateDataControl()
        Dim errCode As Integer
        Set SiebelApplication = CreateObject("SiebelDataControl.SiebelDataControl.1")
        SiebelApplication.Login "host="&"siebel://hostname/EnterpriseServer/AppObjMgr",
        "CCONWAY", "CCONWAY"
        errCode = SiebelApplication.GetLastErrCode()
        If errCode <> 0 Then
            ErrText = SiebelApplication.GetLastErrText
            TheApplication().RaiseErrorText(ErrText);
            Exit Sub
        End If
        set OpptyBC = OpptyB0.GetBusComp("Opportunity", errCode)
    End Sub

See Table 21 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.

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

7884

Siebel Object Interfaces Reference Version 7.7, Rev. C
Set SiebelApplication = CreateObject("SiebelDataControl.SiebelDataControl.1")

' Test to see if object is created
If IsObject(SiebelApplication) = False then
    Response.Write "Unable to initiate Siebel Session.
Else
    connStr = "host=" & Chr(34) & "siebel.tcpip.none.none://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 82.

Java Data Bean

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

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

Supported Platforms and JDKs

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

Instantiating the Java Data Bean

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

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

```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();

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

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

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

        // logoff
        m_dataBean.logoff();
    }
    catch (SiebelException e)
    {
        System.out.println(e.getMessage());
    }
}

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 Connection Properties</td>
<td>siebel.conmgr.txtimeout</td>
<td>Indicates the transaction timeout (in seconds). Defaulted to 2700 = 45m.</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. Defaulted 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. Defaulted to 600 = 10m.</td>
</tr>
<tr>
<td></td>
<td>siebel.conmgr.retry</td>
<td>Indicates the number of open session retries. Defaulted to 3.</td>
</tr>
<tr>
<td></td>
<td>siebel.conmgr.jce</td>
<td>Indicates the usage of Java Cryptography Extension. 1 for jce usage and 0 for no usage.</td>
</tr>
<tr>
<td>Siebel Generated code for JCA/JDB properties</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 code page 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.

```plaintext
siebel.connection.string = siebel.tcpip.rsa.none://test.siebel.com/siebel/
                        sseobjmgr_enu/test
siebel.user.name = User1
siebel.user.password = password
```
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 9. Codepage Mappings for Java Data Bean

<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>unicodell</td>
<td>1200</td>
</tr>
<tr>
<td>unicodeltt</td>
<td>1200</td>
</tr>
<tr>
<td>utf8</td>
<td>65001</td>
</tr>
<tr>
<td>big5</td>
<td>950</td>
</tr>
<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>
</tbody>
</table>
Encrypting Communication Between JDB and Siebel Server

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

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

1. Enable encryption in the corresponding Object Manager Server Component. Please refer to Siebel 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, Siebel Systems supports the Java Cryptography Extension (JCE) v1.2.1 specification. JCE is designed so that other qualified cryptography libraries can be used as service providers.

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

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

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

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

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

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

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 are methods that allow the user to locate instances of objects so that they can be manipulated by other methods.
- **Accessing business components.** These are methods that provide the ability to access and modify data within Siebel applications.
Navigation. These are methods that provide a way to control the flow of the application as it is presented to the user by explicitly setting the focus of the application to the desired view, applet, or control. These methods are useful only when accessing the Siebel object interfaces from Siebel VB and when accessing Siebel as a Mobile/Dedicated Web Client Automation Server. When Siebel is accessed through the COM Data Control, COM Data Server, or Java Data Bean, no Siebel user interface is present.

Manipulating controls. These are the methods that get or set the value of a control. These methods are useful only when accessing controls from Browser Script.

Global state properties and functions. These are methods that get information on the current state.

User interaction. These are methods that provide user interface elements similar to those in standard Windows programs.

See Also
“Locating Objects”
“Accessing Business Components” on page 66
“Navigation Methods” on page 70
“User Interaction Methods” on page 71
“Global State Properties and Functions” on page 71

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 97
- "ActiveViewName Method” on page 124
- "BusComp Method” on page 288
- "BusObject Method” on page 98
- "GetBusObject Method” on page 131
- "GetValue Method” on page 304
- "Name Method” on page 290
- "TheApplication Method” on page 312
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.)
  - BusComp.FirstRecord
  - BusComp.LastRecord
  - BusComp.NextRecord
  - BusComp.PreviousRecord
- Closing a BusComp (Set BusComp = Nothing)

Scenarios for Business Components

The two scenarios that follow involve the use of Siebel scripts to work with business components.
The first example shows how to invoke methods on an existing business component when an event is triggered. In this example, the VB script is in the SetFieldValue event of a business component.

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

    theApplication.TraceOn "c:\temp\trace.txt", "Allocation", "All"
    If FieldName = "Type" Then
        newDesc = "Any valid string which contains the new description."
        desc = Me.GetFieldValue("Description")
        theApplication.Trace "The previous description is " & desc
        Me.SetFieldValue "Description", newDesc
        theApplication.Trace "The new description is " & newDesc
    End If
    theApplication.TraceOff
End Sub
```

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

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

1. GetBusComp
2. SetViewMode (optional, because if you are using Me or the current object, then the BusComp may already be in the correct mode)
3. ActivateField
4. ClearToQuery
5. SetSearchSpec or SetSearchExpr
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;
```

```vbnet
```
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)
{
    ActivateField("Job Title");
    ClearToQuery();
    SetSearchSpec("Job Title", "*VP*");
    ExecuteQuery(ForwardOnly);
    if (FirstRecord())
    {
        TheApplication().RaiseErrorText("Found a decision maker");
        RetValue = CancelOperation;
    }
    else
    {
        RetVal = ContinueOperation;
    }
}
break;
return(RetVal);

The following example shows how to instantiate objects in Siebel 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
  .ActivateField "Job Title"
  .ClearToQuery
  .SetSearchSpec "Job Title", "$VP$"
  .ExecuteQuery ForwardOnly
If (.FirstRecord = 0) Then
  TheApplication.RaiseErrorText "Found a decision maker"
  RetValue = CancelOperation
Else
  RetVal = ContinueOperation
End If
End With
End If
End Select
BusComp_PreSetFieldValue = RetValue
End Function

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

- "ActivateMultipleFields Method" on page 182
- "Associate Method" on page 183
- "ClearToQuery Method" on page 186
- "DeactivateFields Method" on page 188
- "DeleteRecord Method" on page 190
- "ExecuteQuery Method" on page 191
- "ExecuteQuery2 Method" on page 193
- "FirstRecord Method" on page 193
- "FirstSelected Method" on page 196
- "GetFieldValue Method" on page 199
- "GetFormattedFieldValue Method" on page 201
- "GetMultipleFieldValues Method" on page 204
- "GetMVGBusComp Method" on page 204
- "GetNamedSearch Method" on page 206
- "GetPicklistBusComp Method" on page 206
- "GetSearchExpr Method" on page 208
- "GetSearchSpec Method" on page 209
- "GetViewMode Method" on page 211
- "InvokeMethod Method" on page 212
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 10. Navigation Methods

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;InvokeMethod Method&quot; on page 101</td>
</tr>
<tr>
<td>&quot;GotoView Method&quot; on page 140</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 11. User Interaction Methods

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;RaiseError Method&quot; on page 156</td>
</tr>
<tr>
<td>&quot;RaiseErrorText Method&quot; on page 157</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 12. Global State Methods

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;CurrencyCode Method&quot; on page 127</td>
</tr>
<tr>
<td>&quot;EnableExceptions Method&quot; on page 129</td>
</tr>
<tr>
<td>&quot;GetLastError Code Method&quot; on page 134</td>
</tr>
<tr>
<td>&quot;GetLastErrorText Method&quot; on page 135</td>
</tr>
<tr>
<td>&quot;LoginId Method&quot; on page 147</td>
</tr>
<tr>
<td>&quot;LoginName Method&quot; on page 148</td>
</tr>
<tr>
<td>&quot;LookupMessage Method&quot; on page 150</td>
</tr>
<tr>
<td>&quot;PositionName Method&quot; on page 154</td>
</tr>
<tr>
<td>&quot;SetPositionId Method&quot; on page 159</td>
</tr>
<tr>
<td>&quot;SetPositionName Method&quot; on page 160</td>
</tr>
</tbody>
</table>

Variable Scoping for Siebel Script Variables

Three levels of scope exist for Siebel variables:

- "Local Variables"
- "Module Variables"
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.

```eScript
function WebApplet_Load ()
{
    var localStr;
}
```

The following example is in Siebel VB.

```vb
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.

```vb
(general) (declarations)
Dim ContactId as String
```
Code in the VB Editor in the (general) (declarations) section is illustrated in Figure 10.

Figure 10. 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.

```vbnet
TheApplication().gVar = "some value";
```
Inter-Application Variable Methods

Siebel provides two sets of methods to send values for variables back and forth between the Siebel application and external applications. Table 13 lists inter-application communication methods.

Table 13. Inter-Application Communication Methods

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>“GetUserProperty Method” on page 210</td>
</tr>
<tr>
<td>“SetUserProperty Method” on page 243</td>
</tr>
<tr>
<td>“GetLastErrCode Method” on page 134</td>
</tr>
<tr>
<td>“SetSharedGlobal Method” on page 162</td>
</tr>
<tr>
<td>“GetProfileAttr Method” on page 135</td>
</tr>
<tr>
<td>“SetProfileAttr Method” on page 160</td>
</tr>
</tbody>
</table>

Tracing

Table 14 lists Application event methods for controlling debug tracing.

Table 14. Debug Tracing Methods

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Trace Method” on page 166</td>
</tr>
<tr>
<td>“TraceOff Method” on page 168</td>
</tr>
<tr>
<td>“TraceOn Method” on page 169</td>
</tr>
</tbody>
</table>

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 79 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 75
- “How Your Script Affects Program Flow” on page 75
"When Events Occur" on page 79
"Siebel Business Component Events” on page 79
"Applet Events” on page 81
"Application Events” on page 82
"Application Events” on page 82
"Connect String” on page 82
"Error Handling” on page 85

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.
As Figure 11 illustrates, you can change the outcome of an event by specifying the return value on the preoperation events. The standard return value for preoperation events is ContinueOperation, which tells the calling Siebel object to continue processing the remaining operations associated with the event, as shown in Step 2 in Figure 11.

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

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

The following 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);
            iReturn = CancelOperation;
        } 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)
            BusComp_PreSetFieldValue = CancelOperation
        Else
            BusComp_PreSetFieldValue = ContinueOperation
        End If
    End If
End Function

Notice the logical structure of this routine.

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

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

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

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

```vbnet
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.

```javascript
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.

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.

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

For Siebel VB the syntax resembles the following.

```plaintext
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.

```plaintext
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 15 and Table 16 list BusComp events.

Table 15. Server Side BusComp Events

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>“BusComp_Associate Event” on page 250</td>
</tr>
<tr>
<td>“BusComp_ChangeRecord Event” on page 251</td>
</tr>
<tr>
<td>“BusComp_PreCopyRecord Event” on page 255</td>
</tr>
<tr>
<td>“BusComp_CopyRecord Event” on page 252</td>
</tr>
<tr>
<td>“BusComp_InvokeMethod Event” on page 254</td>
</tr>
<tr>
<td>“BusComp_NewRecord Event” on page 254</td>
</tr>
<tr>
<td>“BusComp_PreAssociate Event” on page 255</td>
</tr>
<tr>
<td>“BusComp_PreDeleteRecord Event” on page 256</td>
</tr>
<tr>
<td>“BusComp_PreGetFieldValue Event” on page 257</td>
</tr>
<tr>
<td>“BusComp_PreInvokeMethod Event” on page 258</td>
</tr>
<tr>
<td>“BusComp_PreNewRecord Event” on page 259</td>
</tr>
<tr>
<td>“BusComp_PreQuery Event” on page 259</td>
</tr>
<tr>
<td>“BusComp_PreSetFieldValue Event” on page 260</td>
</tr>
<tr>
<td>“BusComp_PreWriteRecord Event” on page 262</td>
</tr>
<tr>
<td>“BusComp_Query Event” on page 263</td>
</tr>
<tr>
<td>“BusComp_SetFieldValue Event” on page 265</td>
</tr>
<tr>
<td>“BusComp_WriteRecord Event” on page 265</td>
</tr>
</tbody>
</table>

Table 16. Browser Side BusComp Events

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>“BusComp_PreSetFieldValue Event” on page 260</td>
</tr>
</tbody>
</table>
Applet Events

Events are invoked in response to user interactions. These can be managed on a per-applet basis. Applet events are only supported in high interactivity mode. Table 17 and Table 18 list the User interface events.

Table 17. Server Side Applet Events

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;WebApplet_InvokeMethod Event&quot; on page 110</td>
</tr>
<tr>
<td>&quot;WebApplet_Load Event&quot; on page 111</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 18. Browser Side Applet Events

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Applet_ChangeFieldValue Event&quot; on page 104</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 109</td>
</tr>
</tbody>
</table>
Application Events

Application events are listed in Table 19 and Table 20.

Table 19. Server Side Application Events

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Application_InvokeMethod Event” on page 173</td>
</tr>
<tr>
<td>“Application_Navigate Event” on page 174</td>
</tr>
<tr>
<td>“Application_PreInvokeMethod Event” on page 174</td>
</tr>
<tr>
<td>“Application_PreNavigate Event” on page 176</td>
</tr>
<tr>
<td>“Application_Start Event” on page 177</td>
</tr>
</tbody>
</table>

Table 20. Browser Side Application Events

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Application_InvokeMethod Event” on page 173</td>
</tr>
<tr>
<td>“Application_PreInvokeMethod Event” on page 174</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:

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

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

```
Siebel Application.Login "host=""siebel://host/EnterpriseServer/AppObjMgr"",
"CCONWAY", "CCONWAY"
```

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

```
siebel.TCPIP.None.None://host/siebel/AppObjMgr
```

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://hhost/siebel/AppObjMgr
```
Protocols that are not specified receive their default values, as shown in Table 21.

Make the following substitutions for the placeholders in the example.

Table 21. 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>One of the following values:</td>
</tr>
<tr>
<td></td>
<td>■ tcpip (the default)</td>
</tr>
<tr>
<td></td>
<td>■ http</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>■ mscrypto (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</td>
</tr>
<tr>
<td></td>
<td>administrator changes the default during installation.</td>
</tr>
<tr>
<td></td>
<td>For information about load-balancing with SCBroker, see Deployment</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>the thin client to access; this can be a user-defined component or one</td>
</tr>
<tr>
<td></td>
<td>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, read Siebel System Administration Guide.</td>
</tr>
</tbody>
</table>

For more information about this method, read "Login Method" on page 145.

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

```plaintext
'COM Data Control : SERVER Mode
lstr = "host=" + "siebel://frashid/Siebel/SSEObjMgr"
'Format of the connect string is
```

Siebel Object Interfaces Reference Version 7.7, Rev. C 

83
Programming ■ Siebel Object Interface Events and Siebel Extension Events

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

**Leveraging Load Balancing with the Connect String**

Siebel COM Data Control operating in server mode and Java Data Beans 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.

The connect strings used to leverage Siebel native load balancing have the following requirements:

- **COM Data Control.** The connect string has the following structure:

  ```
  host="siebel://VirtualHost\EnterpriseServer\AppObjMgr\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 Beans.** The connect string has the following structure:

  ```
  host="siebel://VirtualHost\EnterpriseServer\AppObjMgr"
  ```

  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:
Error Handling

This section explains the Siebel COM Interfaces error handling differences.

COM Error Handling

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

Error Handling Example—COM Data Server only

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

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

GetBusObject (BusObjectName as string) -> businessObject

Java Error Handling

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

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

It is used as follows.

```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 \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.

**GetLastErrCode, GetLastErrText Method**

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

GetLastErrCode() ' retrieves errCode As Integer
GetLastErrText() ' retrieves text As String
4 Interfaces Reference

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

- “Object Interface Methods Tables” on page 87
- “Object Interface Events” on page 94
- “Siebel Constants” on page 96
- “Applet Methods” on page 97
- “Applet Events” on page 103
- “Application Methods” on page 119
- “Application Events” on page 172
- “Business Component Methods” on page 179
- “Business Component Events” on page 250
- “Business Object Methods” on page 266
- “Business Service Methods” on page 270
- “Business Service Events” on page 281
- “Control Methods” on page 287
- “Property Set Methods” on page 295
- “Miscellaneous Methods” on page 310

Object Interface Methods Tables

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

- “Applet Methods”
- “Application Methods” on page 88
- “Business Component Methods” on page 90
- “Business Object Methods” on page 91
- “Business Service Methods” on page 92
- “Control Methods” on page 92
- “Property Set Methods” on page 93
- “Miscellaneous Methods” on page 93
## Applet Methods

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveMode Method</td>
<td>X</td>
<td></td>
<td></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></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FindControl 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>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>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveApplet Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ActiveBusComp Method</td>
<td>X</td>
<td></td>
<td></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></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attach Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>CurrencyCode 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>Detach Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>EnableExceptions Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FindApplet Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetBusObject Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GetDataSource Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GetLastErrCode Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GetLastErrText Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GetProfileAttr 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>GetService 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>--------------------------</td>
<td>---------------</td>
<td>----------------</td>
<td>---------------------------------</td>
<td>-----------------------------------------------</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>X</td>
<td>X</td>
</tr>
<tr>
<td>GotoView 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>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>LoadObjects Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>LoadUserAttributes Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Login Method</td>
<td></td>
<td></td>
<td></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>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>LookupMessage Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>LookupValue Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Name Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>NewPropertySet 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>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>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RaiseError Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>RaiseErrorText Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>SetPositionId Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SetPositionName Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SetProfileAttr 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>SetSharedGlobal Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ShowModalDialog Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>SWEAlert Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Trace Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>TraceOff Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>TraceOn Method</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
## Business Component Methods

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ActivateField Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ActivateMultipleFields Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Associate Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusObject Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ClearToQuery Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeactivateFields Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeleteRecord Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ExecuteQuery Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ExecuteQuery2 Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FirstRecord Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FirstSelected Method</td>
<td>X</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></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GetFormattedFieldValue Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GetLastErrCode Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetLastErrText Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></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>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetNamedSearch Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GetPicklistBusComp Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GetSearchExpr Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GetSearchSpec Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GetUserProperty Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GetViewMode Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>LastRecord Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Name 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>NewRecord Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
### Business Object Methods

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GetBusComp Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetLastErrCode Method</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetLastErrText 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>
<tr>
<td>Release Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

## Interfaces Reference

**Object Interface Methods Tables**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NextRecord Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</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>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pick Method</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PreviousRecord Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RefineQuery Method</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>SetFieldSetValue Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetFormattedFieldValue Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetMultipleFieldValues Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetNamedSearch Method</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetSearchExpr Method</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetSearchSpec Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetSortSpec Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetUserProperty Method</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetViewMode Method</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UndoRecord Method</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WriteRecord Method</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Business Service Methods

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetLastErrText Method</td>
<td></td>
<td></td>
<td></td>
<td></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>InvokeMethod 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>Name 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>PropertyExists 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>Release Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RemoveProperty 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>SetProperty 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>

### Control Methods

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Applet Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BusComp Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetProperty Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetValue 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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetLabelProperty Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetProperty Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetValue Method</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Property Set Methods

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></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>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>GetFirstChild 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>GetNextChild 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>GetParent 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>RemoveChild 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>SetProperty 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>SetType 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>SetValue 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>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GetErrorCode 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>
Object Interface Events

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

- "Applet Events"
- "Application Events” on page 95
- "Business Component Events” on page 95
- "Business Service Events” on page 96

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>
<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>
<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>
<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>
</tbody>
</table>
Siebel Constants

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

<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>
<tr>
<td>Search Methods</td>
<td>ForwardBackward</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ForwardOnly</td>
<td>1</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>
</tbody>
</table>
Applet Methods

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

- "ActiveMode Method"
- "BusComp Method" on page 98
- "BusObject Method" on page 98
- "FindActiveXControl Method" on page 99
- "FindControl Method" on page 100
- "InvokeMethod Method" on page 101
- "Name Method" on page 102

**ActiveMode Method**

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

**Syntax**

```java
oApplet.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();
    alert("The active mode for the selected applet is: " + currMode);
}
```

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

**Syntax**
```javascript
oApplet.BusComp();
```

<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 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
oApplet.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 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();
    alert("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.

```vbscript
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**
`oApplet.FindActiveXControl(controlName)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>controlName</code></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`.

**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
```javascript
oApplet.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.

Used With
Browser Script

Example
To use this example, read the notes for the "SetLabelProperty Method" on page 290.

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

**InvokeMethod Method**

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

**Browser Script Syntax**

```javascript
oApplet.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**

```javascript
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 methodName</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. Siebel Systems 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;
}
return(iReturn);
}
```

**Name Method**
The Name method returns the name of the applet.

**Syntax**

```javascript
oApplet.Name()
```

**Argument**
Not applicable

**Returns**
A string containing the applet object name.

**Used With**
Browser Script, Server Script
Example
The following example is in Browser Script.

```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();
    alert("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.

```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 104
- "Applet_ChangeRecord Event" on page 105
- "Applet_InvokeMethod Event" on page 106
- "Applet_Load Event" on page 107
- "Applet_PreInvokeMethod Event" on page 109
- "WebApplet_InvokeMethod Event" on page 110
- "WebApplet_Load Event" on page 111
- "WebApplet_PreCanInvokeMethod Event" on page 112
- "WebApplet_PreInvokeMethod Event" on page 113
- "WebApplet_ShowControl Event" on page 115
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.

Example

The following example is in Browser Script.

```javascript
function Applet_ChangeFieldValue (fieldname, fieldValue)
{
  try
  {
    switch (fieldname)
    {
      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");
    alert("Changing the Committed Flag to NO as $500,000 in Revenue +
    Upside amount is required");
}
break;
}

See Also
"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.

    function Applet_ChangeRecord ()
    {
        try
        {
            var thisBC = this.BusComp();
            var sFlag = thisBC.GetFieldValue("Primary Revenue Committed Flag");
            if (total < 500000)
            {
                thisBC.SetFieldValue("Primary Revenue Committed Flag", "N");
                alert("Changing the Committed Flag to NO as $500,000 in Revenue +
                Upside amount is required");
            }
            break;
        }
    }
if (sFlag == "Y")
{
    alert("This record cannot be update as its been Committed");
}

catch(e)
{
    alert("Error in ChangeFieldValue and error is " + e.toString() + " + " + e.errText());
}

See Also
"Applet_ChangeFieldValue Event” on page 104

Applet_InvokeMethod Event

The 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");
        . . . . .
    }
}
```

See Also
"Applet_PreInvokeMethod Event" on page 109
"Application_InvokeMethod Event" on page 173

**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.

**Used With**

Browser Script

**Examples**

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:** These examples are 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");
}
```

This event can also be used to filter records.

```javascript
Function Applet_Load()
{
    var bc = this.BusComp();
    bc.SetSearchExpr("<new expression>");
    bc.ExecuteQuery();
}
```
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
Browser Script

```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");
    } else
        return ("ContinueOperation");
}
```
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>methodName</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.

```
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.

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

See Also

"How Your Script Affects Program Flow" on page 75
The Application.SetSharedGlobal "EnableButton", ""
break
End Select

See Also
"Applet_InvokeMethod Event” on page 106
"Application_InvokeMethod Event” on page 173
"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);
```
The following example is in Siebel VB.

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

**See Also**

"Applet_InvokeMethod Event” on page 106  
"Application_InvokeMethod Event” on page 173  
"WebApplet_PreCanInvokeMethod Event” on page 112

**WebApplet_PreCanInvokeMethod Event**

The PreCanInvokeMethod event is called before the PreInvokeMethod and also when an applet is loaded, allowing the script to determine whether or not the user has the authority to invoke the Applet method.

**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>

**Returns**

CancelOperation or ContinueOperation
**Used With**
Server Script

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

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

The following example is in Siebel VB.

```plaintext
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)`

**Argument** | **Description**
--- | ---
`methodName` | String variable or literal containing the name of the method invoked

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

CancelOperation does not stop the execution of the code following it, but it does prevent the execution of any built-in code associated with this event. WebApplet_PreInvokeMethod should return CancelOperation when you are handling the event entirely through scripting and you 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.

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

- The <swe:control> tag and <swe:this> tag are used to show a control:
  
  `<swe:control id="1">
  
  ```
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: Base, Edit, New, Query, 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.

```vb
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 133
- “GetLastError Method” on page 134
- “GetLastErrText Method” on page 135
- “GetProfileAttr Method” on page 135
- “GetService Method” on page 136
- “GetSharedGlobal Method” on page 138
ActiveApplet Method

ActiveApplet returns a reference to the applet that currently has focus.
Syntax
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();
      alert("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
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();
    alert(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/Dedicated 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.GetField("Email Address")
            SendEmail(sEmail)
```
ActiveViewName Method

ActiveViewName returns the name of the active view.

Syntax

```
Application.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/Dedicated Web Client Automation Server, Server Script

Example

The following example is in Siebel eScript.

```
function BusComp_PreSetFieldValue (FieldName, FieldValue)
{
    var iReturn = ContinueOperation;
    switch(FieldName)
    {
    case "Name":
    case "Location":
    case "Account Status":
    case "Alias":
    case "City":
    case "Country":
```
case "Currency Code":
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");
        iReturn = CancelOperation;
    } else {
        break;
    }
    return (iReturn);
}

### Attach Method

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

#### Syntax

```java
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 or a value returned from the Siebel cookie.</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:

```vba
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
Else
    ErrText = SiebelApplication_first.GetLastErrText
    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("siebel.TCPIP.none.none://<virtualip>: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></td>
<td>Not applicable</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></td>
<td>Not applicable</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**

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

**Argument**

<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/Dedicated 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 GetLastErrorCode method.

```vba
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.TCPIPNONE/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.GetLastErrorCode()
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 Dedicated/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
GetLastError 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.GetLastErrorCode()
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**

```plaintext
declareApplet.appletName
```

**Argument**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>appletName</code></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:

```plaintext
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();
    alert(entryappletvalue);
  }
}
```

**GetBusObject Method**

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

Application.GetBusObject(busObjectName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>busObjectName</td>
<td>String variable or literal containing the name of the business object to instantiate.</td>
</tr>
</tbody>
</table>

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/Dedicated 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.

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

[ Your code here ]

oBusComp = null;
oBusObject = null;
```

The following example is in Siebel VB.

```vbnet
Dim AccntBO as BusObject
Dim AccntBC as BusComp
Dim AddrBC as BusComp
Set AccntBO = theApplication.GetBusObject("Account")
Set AccntBC = AccntBO.GetBusComp("Account")

[ your code here]

Set AccntBO = Nothing
Set AccntBC = 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 DataBean.

```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)
```

**GetDataSource Method**

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

**Syntax**

```
dataSrc = Application.InvokeMethod("GetDataSource")
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

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

**Used With**

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

**Example**

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

```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)

**GetLastErrorCode Method**

The GetLastErrorCode method returns the last error execution status.

**Syntax**

*Application*.GetLastErrorCode

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not applicable</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 GetLastErrorCode can be invoked to check if any error was returned from the previous operation. GetLastErrorText method can be invoked to retrieve the text of the error message. Each method invocation resets the execution status.

**Used With**


**Example**

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

```vbscript
errcode = SiebelApplication.GetLastErrorCode
If errcode <> 0 Then
    ErrText = SiebelApplication.GetLastErrorText
    MsgBox ErrText
    Exit Sub
End If
```

**See Also**

“GetLastErrorText Method” on page 135
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


Example

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

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

See Also

“GetLastErrCode Method” on page 134

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.

Used With
Browser Script, COM Data Control, COM Data Server, Java Data Bean, Mobile/Dedicated 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")
```

See Also
“SetProfileAttr Method” on page 160

GetService Method
The GetService method returns a specified service. If the 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 service is not found, GetService searches through services defined in the run-time Business Services table.

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

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

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

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

Used With

Examples
The following examples instantiate a business service named Workflow Process Manager.

The following example is in Browser Script.

```javascript
function Applet_PrevInvokeMethod (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.
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.

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("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, Java Data Bean, Mobile/Dedicated 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 in 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.

```vbnet
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.

```vbnet
```
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

See Also
"SetSharedGlobal Method" on page 162

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, 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.

```javascript
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 a excellent lead,
        //go to the account for this opportunity
        actName = this.GetFieldValue("Account");
        actBO = TheApplication().GetBusObject("Account");
        actBC = actBO.GetBusComp("Account");

        with (actBC)
        {
            SetViewMode(AllView);
            ClearToQuery();
            SetSearchSpec("Name", actName);
            ExecuteQuery();
        }
        TheApplication().GotoView("All Account List View",actBO);
    }
}
```

```
```
The following example is in Siebel VB.

```vbnet
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
            .SetViewMode AllView
            .ClearToQuery
            .SetSearchSpec "Name", actName
            .ExecuteQuery
        End With
        TheApplication.GotoView "All Account List View", actBO
    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**

`Application.InvokeMethod(methodName, methodArgs_PropSet);`

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

**Usage**

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

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

**Used With**

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

**Example**

For an example, read “InvokeMethod Method” on page 101.

**LoadObjects Method**

The LoadObjects method is used to start the COM Data Server object, and returns a reference to the Application object. This method must be the first call to the COM Data Server.
Syntax
Application.LoadObjects(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. example: C:\siebel\bin\uagent.cfg</td>
</tr>
</tbody>
</table>

Returns
The Application object opened on start-up

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.

```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.GetLastErrText  
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 person whose profile needs to be loaded.

Returns
Not applicable

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

Used With
Server Script

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

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

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

```vbnet
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

```csharp
Application.Login([connectString,] userName, password)
```

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

Returns

A string containing the error code

Usage

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

For information on formatting the connect string, read “Connect String” on page 82.

Used With

COM Data Control, COM Data Server, Java Data Bean

Example

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

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

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:

```csharp
m_dataBean.login("siebel.tcpip.none.none://gateway:gatewayport/enterprise/SCCObjMgr", 'username', 'password');
```

To use the COM Data Control in C++:

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

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

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

//Check for errors
If SiebelAppControl.GetLastErrCode <> 0 Then
    frmMain.txtStatus.Text = SiebelAppControl.GetLasErrText
```
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.getErrorMessage());
        }
    }
}
```

**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 row ID of the user’s login in the Employee table. Once obtained, the login ID can be conveniently used as a search specification.

Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile/Dedicated 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
    Dim iReturn as integer
    iReturn = ContinueOperation
    Select Case FieldName
        Case "Account Status"
            if Me.GetFieldValue("Created By") <> _
                TheApplication.LoginId then
                TheApplication.RaiseErrorText("*** You cannot change Account Status _
                because you did not create the record***")
                iReturn = CancelOperation
            end if
    End Select
    BusComp_PreSetFieldValue = iReturn
End Function
```

LoginName Method

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

**Syntax**

```
Application>LoginName
```

**Returns**

A string containing the user’s login name

**Used With**

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

**Example**

For examples, read ”ExecuteQuery Method” on page 191 and ”TheApplication Method” on page 312.

**See Also**

”Login Method” on page 145

---

### Logoff Method

The Logoff method disconnects the client from the server.

**Syntax**

```
Application.Logoff
```

**Argument** | **Description**
--- | ---
Not applicable |

**Returns**

Not applicable

**Usage**

For clients with user interfaces, Logoff destroys every window except for the topmost window. Logoff also deletes every object, except for the topmost object, on both client and server.

Logoff is called automatically if you destroy the main object.
Used With
COM Data Control, Java Data Bean, Mobile/Dedicated Web Client Automation Server

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");
```
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

COM Data Control, Java Data Bean, Mobile/Dedicated Web Client Automation Server, 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.

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

Name Method

The Name method returns name of the application.

Syntax

Application.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 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

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

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();
```
Siebel Object Interfaces Reference Version 7.7, Rev. C

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

```vbs
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");
oBS.InvokeMethod("New Value Method", inpPS, outPS);
    inpPS = null;
    outPS = null;
oBS = null;
    return {"CancelOperation"};
}
else
{
    return {"ContinueOperation"};
}
}
```

The following example is in Siebel VB.

```vbs
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"};
}
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

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

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

<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 user’s position

Used With

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

Example

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

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

Application.RaiseError(key, [arg1], [arg2], [arg3], ..., [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.

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

If you have implemented error handling in Siebel VB, remember that 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.

**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.

```javascript
function BusComp_PredeleteRecord ()
{
    try {
        var status = this.getFieldValue("Account Status");
```
if (status == "Gold") {
    TheApplication().RaiseError (<user defined error name>);
    return (CancelOperation);
}
else {
    return (ContinueOperation);
}
} catch (e) {
    throw e;
}

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 predefined 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 'RaiseError 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");
            return (CancelOperation);
        }
        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>value</td>
<td>The error text message.</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

Not applicable

Usage

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

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

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');
            return (CancelOperation);
        }
        else {
            return (ContinueOperation);
        }
    }
}
```
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);
            return (CancelOperation);
        }
        else
        {
            return (ContinueOperation);
        }
    }
    catch (e)
    {
        throw e;
    }
}

The following eScript example raises an error when deleting an opportunity with the “Pipeline” revenue class.

### SetPositionId Method

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

**Syntax**

`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.
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.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/Dedicated 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><code>name</code></td>
<td>A string indicating the name of the attribute</td>
</tr>
<tr>
<td><code>value</code></td>
<td>The value of <code>name</code></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 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.

Used With
Browser Script, COM Data Control, COM Data Server, Server Script, Java Data Bean, Mobile/Dedicated 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"};
    } else
        return {"ContinueOperation"};
}
```

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
```

See Also
"Name Method” on page 151. For more information on user profile attributes, read *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 may be accessed using GetSharedGlobal.

#### Syntax

`Application.SetSharedGlobal(varName, value)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>varName</code></td>
<td>String variable or literal containing the name of the shared global variable to set</td>
</tr>
<tr>
<td><code>value</code></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/Dedicated Web Client Automation Server, Server Script

#### Example

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

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

The following example is in Siebel VB.
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.

```vbscript
dim oleVar 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.

```vbscript
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
```

See Also
"GetLastErrCode Method” on page 134

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.

This method is implemented in Siebel Business Applications release 7.7.2.2 and later.
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 optional 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.</td>
</tr>
<tr>
<td>options</td>
<td>String that specifies the attributes of the window that displays the dialog box. This optional parameter may include one or more of the following semicolon-delimited values:</td>
</tr>
<tr>
<td></td>
<td>■ dialogHeight:sHeight sets the height of the dialog window, where sHeight 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>
</tr>
<tr>
<td></td>
<td>■ dialogLeft:sXPos sets the left position of the dialog window relative to the upper-left corner of the desktop.</td>
</tr>
<tr>
<td></td>
<td>■ dialogTop:sYPos sets the top position of the dialog window relative to the upper-left corner of the desktop.</td>
</tr>
<tr>
<td></td>
<td>■ dialogWidth:sWidth sets the width of the dialog window.</td>
</tr>
<tr>
<td></td>
<td>■ center:{ yes</td>
</tr>
<tr>
<td></td>
<td>■ dialogHide:{ yes</td>
</tr>
<tr>
<td></td>
<td>■ edge:{ sunken</td>
</tr>
<tr>
<td></td>
<td>■ help:{ yes</td>
</tr>
<tr>
<td></td>
<td>■ resizable:{ yes</td>
</tr>
<tr>
<td></td>
<td>■ scroll:{ yes</td>
</tr>
<tr>
<td></td>
<td>■ status:{ yes</td>
</tr>
<tr>
<td></td>
<td>■ unadorned:{ yes</td>
</tr>
</tbody>
</table>
Returns
The value of the returnValue property, as set by the window of the document specified by the url parameter.

Used With
Browser Script

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, popup applets such as Mvg and Pick applets are hidden (sent to the background) when a JavaScript Alert() is raised by a Browser side event. With SWEAlert, the dialog's parent applet is not sent to the foreground.

Used With
Browser Script

Example
The following 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
  return ("ContinueOperation");
  

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 is the application’s CFG file. For more information, read “Script Tracing” on page 30.

Syntax

Application.Trace(message)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String variable or literal containing message text to append to the trace file</td>
</tr>
</tbody>
</table>

Returns

Not applicable

Used With

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

Example

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

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.CREATED_BY,
  T1.LAST_UPD_BY,
  T1.CREATED,
  T1.LAST_UPD,
  T1.CONFLICT_ID,
  T1.NAME,
  T1.DESC_TEXT,
  T1.PRIV_FLG,
  T1.QUERY_STRING
FROM
  DEV32.S_APP_QUERY T1
WHERE
  (T1.CREATED_BY = :1 OR T1.PRIV_FLG = :2) AND
  ((T1.NAME LIKE :3 OR T1.NAME LIKE :4 OR T1.NAME LIKE :5 OR
    T1.NAME LIKE :6) AND UPPER(T1.NAME) = UPPER(:7))
ORDER BY
  T1.NAME, T1.DESC_TEXT"
01/22/98, 21:04:10, SQLBIND, 1, 1, 1-6NF
01/22/98, 21:04:10, SQLBIND, 1, 2, N
01/22/98, 21:04:10, SQLBIND, 1, 3, ac%
01/22/98, 21:04:10, SQLBIND, 1, 4, Ac%
01/22/98, 21:04:10, SQLBIND, 1, 5, aC%
01/22/98, 21:04:10, SQLBIND, 1, 6, AC%
01/22/98, 21:04:10, SQLBIND, 1, 7, Account
```

See Also

"TraceOff Method"
"TraceOn Method" on page 169
**TraceOff Method**

TraceOff turns off the tracing started by the TraceOn method.

**Syntax**

```
Application.TraceOff
```

**Returns**

Not applicable

**Used With**

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

**Example**

This Siebel VB example sets the value in the Sales Stage field to the default, that is, to the first value in the field’s picklist, and uses tracing to track the result.

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

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

Syntax

Application.TraceOn(filename, type, selection)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>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 $t substitutes the thread id to the file name. For example: TheApplication().TraceOn(&quot;d:\temp\trace_$p_$t.txt&quot;, &quot;Allocation&quot;, &quot;All&quot;); 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_112.txt If user B had a process id of 11, and a thread id of 2, their tracing file would be d:\temp\trace_112.txt As a result, both users would attempt to log to the same file. Adding a separator between the process and thread id keeps the filenames unique. d:\temp\trace_1_12.txt d:\temp\trace_11_2.txt</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of tracing to start. This can have the following values: * Allocation. Traces allocations and deallocations of Siebel objects. This option is useful if you suspect memory leaks in your code. * SQL. Traces SQL statements generated by the Siebel application.</td>
</tr>
<tr>
<td>selection</td>
<td>Indicates which Siebel objects should be traced for the Allocation trace type. This argument should be &quot;&quot; if the trace type is SQL: * Script. Traces VB and eScript objects. * OLE. Traces allocations for data server or automation server programs. * All. Traces all objects. The All value does not trace the Siebel objects managed implicitly by Siebel's declarative configuration use. All traces the Siebel objects constructed by scripting.</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. You can issue multiple TraceOn statements to the same trace file.

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

**Example**
The following example is for COM Data Server. SiebelApplication 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.

```sql
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"
```

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(3);
    ClearToQuery();
    ActivateField("Sales Stage Order");
    SetSortSpec("Sales Stage Order(ASCENDING)");
    ExecuteQuery(1);
  }
```
The following example is in Siebel VB.

```vbscript
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(ASCENDING)"
    With oBC
        .SetViewMode AllView
        .ClearToQuery
        .ActivateField "Sales Stage Order"
        .SetSortSpec "Sales Stage Order"
        .ExecuteQuery ForwardOnly
        If .FirstRecord Then
            .Pick
        End If
    End With
    Set oBC = Nothing
    TheApplication.Trace "End of tracing!"
    TheApplication.TraceOff
End Sub
```

**See Also**

"Trace Method" on page 166
"TraceOff Method" on page 168

**Application Events**

The following topics describe application events:

- "Application_Close Event"
- "Application_InvokeMethod Event" on page 173
- "Application_Navigate Event" on page 174
- "Application_PreInvokeMethod Event" on page 174
- "Application_PreNavigate Event" on page 176
Application_Close Event

The Close event is called before the application exits. This allows Basic scripts to perform last-minute cleanup (such as cleaning up a connection to a COM server). It is called when Windows notifies the application that it should close, but not if the process is terminated directly.

Syntax
Application_Close

Returns
Not applicable

Used With
Server Script

NOTE: Siebel Business Processes invokes this event. For more information, read Siebel Business Process Designer Administration Guide.

Application_InvokeMethod 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

See Also
“How Your Script Affects Program Flow” on page 75
“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.

```
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.

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

## Interfaces Reference ■ Application Events

**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
```

**Siebel Object Interfaces Reference** Version 7.7, Rev. C
Dim i As Integer
Dim iReturn As Integer
iReturn = ContinueOperation

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

Application_PreInvokeMethod = iReturn

End Function

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":
            Clib.system("C:\\Program Files\\Microsoft Office \\
Office\\WINWORD.EXE",1);
            iReturn = CancelOperation;
            break;
        case "LaunchExcel":
            Clib.system("C:\\Program Files\\Microsoft Office \\
Office\\EXCEL.EXE",1);
            iReturn = CancelOperation;
    
    return (iReturn)
}

See Also
"How Your Script Affects Program Flow" on page 75

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
```

**Argument** | **Description**
--- | ---
DestViewName | Name of the View to which the user is navigating
DestBusObjName | Business object of the destination view

**Returns**
CancelOperation or ContinueOperation

**Used With**
Server Script

**Example**
In the following 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).
```
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 invokes this event. For more information, read Siebel Business Process Designer Administration Guide.

Returns

Not applicable

Used With

Server Script

Example

This Siebel VB code should be placed in the Application_Start procedure for the application of your choice. This example retrieves the first and last name of the user logging into the Siebel application.

```vbnet
Sub Application_Start(CommandLine As String)
    Dim oEmpBusObj As BusObject
    Dim oEmpBusComp As BusComp
    Dim sLoginName As String
    Dim sUser Name As String

    sLoginName = TheApplication.LoginName
    Set oEmpBusObj = TheApplication.GetBusObject("Employee")
    Set oEmpBusComp = oEmpBusObj.GetBusComp("Employee")
    With oEmpBusComp
        .ActivateField("Login Name")
        .ActivateField("First Name")
        .ActivateField("Last Name")
        .ClearToQuery
        .SetSearchSpec "Login Name", sLoginName
        .ExecuteQuery
        If .FirstRecord Then
            sUserName = .GetFieldValue("First Name")
            sUserName = sUserName + " " + .GetFieldValue("Last Name")
        End If
    End With

    Set oEmpBusComp = Nothing
    Set oEmpBusObj = Nothing
End Sub
```
Business Component Methods

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

- "ActivateField Method” on page 180
- "ActivateMultipleFields Method” on page 182
- "Associate Method” on page 183
- "BusObject Method” on page 185
- "ClearToQuery Method” on page 186
- "DeactivateFields Method” on page 188
- "DeleteRecord Method” on page 190
- "ExecuteQuery Method” on page 191
- "ExecuteQuery2 Method” on page 193
- "FirstRecord Method” on page 193
- "FirstSelected Method” on page 196
- "GetAssocBusComp Method” on page 197
- "GetFieldValue Method” on page 199
- "GetFormattedFieldValue Method” on page 201
- "GetLastErrCode Method” on page 202
- "GetLastErrText Method” on page 203
- "GetMultipleFieldValues Method” on page 204
- "GetMVGBusComp Method” on page 204
- "GetNamedSearch Method” on page 206
- "GetPicklistBusComp Method” on page 206
- "GetSearchExpr Method” on page 208
- "GetSearchSpec Method” on page 209
- "GetUserProperty Method” on page 210
- "GetViewMode Method” on page 211
- "InvokeMethod Method” on page 212
- "LastRecord Method” on page 218
- "Name Method” on page 218
- "NewRecord Method” on page 219
- "NextRecord Method” on page 220
- "NextSelected Method” on page 221
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 prior to executing a query for the business component.

NOTE: If you are writing an event handler on a business component, you must make sure that the field has already been activated by specifying the ForceActive user property on the control.
By default, fields are inactive except when:

- They are displayed on the applet and the business component is the instance on which the applet is based.
- They are System fields (which include Id, Created, Created By, Updated, and Updated By).
- Their ForceActive property is set to TRUE.
- The method ActivateField has been invoked with the FieldName.
- They have the Link Specification property set to TRUE.

After a business component has been executed, if additional fields are activated, the business component must be requeried before field values can be accessed. Failure to requery the business component results in a value of 0 being returned. The ActivateField method destroys the context of a query when it is used after the ExecuteQuery method.

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/Dedicated 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 186.

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

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

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

Syntax

```java
BusComp.ActivateMultipleFields(SiebelPropertySet 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/Dedicated 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 ps = new mdata_bean.NewPropertySet();
ps.setProperty("Account Products","");
ps.setProperty("Agreement Name","");
ps.setProperty("Project Name","");
ps.setProperty("Description","");
ps.setProperty("Name","");
siebBusComp.ActivateMultipleFields(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)
{
    ClearToQuery();
    SetViewMode(AllView);
    var fieldsPS = TheApplication().NewPropertySet();
    var valuesPS = TheApplication().NewPropertySet();
    fieldsPS. SetProperty("Last Name", ");
    fieldsPS. SetProperty("First Name", ");
    ActivateMultipleFields(fieldsPS);
    ExecuteQuery();
    if (FirstRecord())
    {
        GetMultipleFieldValues(fieldsPS, valuesPS);
        var slName = valuesPS.GetProperty("Last Name");
        var sfName = valuesPS.GetProperty("First Name");
    }
}
```

**See Also**

"SetMultipleFieldValues Method" on page 232
"GetMultipleFieldValues Method" on page 204

### 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(whereIndicator)`

**Argument**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>whereIndicator</code></td>
<td>This argument should be one of the following predefined constants or the corresponding integer: NewBefore (0) or NewAfter (1), 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/Dedicated 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.

```vbnet
Dim oParentBC as BusComp
Dim oMvgBC as BusComp
Dim oAssocBC as BusComp

Set oParentBC = me.BusComp
Set oMvgBC = OpBC.GetMVGBusComp("Sales Rep")
Set oAssocBC = oMvgBC.GetAssocBusComp
With oAssocBC
    .SetSearchSpec "Id", newPosId
    .ExecuteQuery
    .Associate NewAfter
End With

oMvgBC.SetFieldValue "Opportunity Assignment Type", "NewType"
oMvgBC.WriteRecord
Set oAssocBC = Nothing
Set oMvgBC = Nothing
Set oParentBC = Nothing
```

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.

```escript
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();
    if (FirstRecord())
    {
        // Instatiates 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;
      TheApplication().RaiseErrorText("Error Associating new Organization");
   }
}
} // if oAssocBC.FirstRecord
} // if FirstRecord
} // With ContactBC

See Also
"NewRecord Method" on page 219
"FirstSelected Method" on page 196
"GetMVGBusComp Method" on page 204

**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>Not applicable</td>
<td></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/Dedicated Web Client Automation Server, Server Script
Example
For an example, read “SetViewMode Method” on page 245.

See Also
“ActiveBusObject Method” on page 122

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 180.

Search and sort specifications sent to the business component are cumulative; the business component retains and logically ANDs query qualifications since the last ClearToQuery, except for new search specifications on a field for which a search specification has previously been set. In that circumstance, the new specification replaces the old.

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

Example
The following example is in Siebel eScript. For Siebel VB examples, read “Applet_PreInvokeMethod Event” on page 109, “ActivateField Method” on page 180, and “ExecuteQuery Method” on page 191. For another eScript example, read “GotoView Method” on page 140.

```javascript
var oEmpBusObj = TheApplication().ActiveBusObject();
var oEmpBusComp = oEmpBusObj().GetBusComp("Employee");
var sLoginName;
```
oEmpBusComp.ActivateField("Login Name");
oEmpBusComp.ClearToQuery();
oEmpBusComp setSearchSpec("Login Name", sLoginName);
oEmpBusComp.ExecuteQuery();

oEmpBusComp = null;
oEmpBusObj = null;

See Also
“RefineQuery Method” on page 225

CountRecords Method
CountRecords uses database aggregation to count the records in a business component.

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
Not Applicable

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();
      ActivateField("Name");
      setSearchSpec("Name", "A*");
      ExecuteQuery();
      var count = CountRecords();
    }
```
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
```

<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 must activate fields using ActivateField prior to executing a query for the business component. By default, fields are inactive except when:

- They are displayed on the applet and the business component is the instance on which the applet is based.
- They are System fields (which include Id, Created, Created By, Updated, and Updated By).
- Their ForceActive property is set to TRUE.
- The method ActivateField has been invoked with the FieldName.
- They have the Link Specification property set to TRUE.

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

Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script
Examples
The following example is for COM. Siebel Application is an Application instance.

```
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.

```
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.

```
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
```

See Also
“ActivateField Method” on page 180
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/Dedicated Web Client Automation Server, Server Script

Example

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

```vbnet
Sub DeleteInactiveAccounts()
    Dim objBO as BusObject
    Dim objBC as BusComp
    Set objBO = theApplication.GetBusObject("Account")
    Set objBC = objBO.GetBusComp("Account")
    With objBC
        .ClearToQuery
        .SetSearchSpec "Status", "Inactive"
        .ExecuteQuery ForwardBackward
        Do While .FirstRecord
            .DeleteRecord
        Loop
    End With
    Set objBC = Nothing
    Set objBO = Nothing
End Sub
```

NOTE: The cursor is moved to the next record after DeleteRecord is executed. Therefore, it is not necessary to execute NextRecord after DeleteRecord. Do not use NextRecord after DeleteRecord in a loop because this causes the deletion of the last record in the loop to be skipped. If you use DeleteRecord on the last record, the cursor points to nothing.
ExecuteQuery Method

ExecuteQuery returns a set of BusComp 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>cursorMode</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.

Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile/Dedicated 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 109, "GotoView Method" on page 140, and "ClearToQuery Method" on page 186.

```vbscript
Option Explicit
Function BusComp_PreSetFieldValue (FieldName As String,FieldValue As String) As Integer
Dim iReturn As Integer, i As Integer
Dim iFoundP As Integer ' 1 = found (TRUE), 0 = not found (FALSE)
Dim oMVGBC as BusComp

iReturn = ContinueOperation
iFoundP = FALSE
Select Case FieldName
Case "SSA Primary Field"
    set oMVGBC = me.ParentBusComp.GetMVGBusComp("Sales Rep")
    With oMVGBC ' this is the position BC
        .ActivateField "Active Login Name"
        .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")
                    iReturn = CancelOperation
                end if
            end if
            i = .NextRecord
        Loop
    if iFoundP = FALSE then
        .FirstRecord
        TheApplication.RaiseErrorText("No Primary Found - Contact an Administrator")
    end if
End Select

set oMVGBC = Nothing
BusComp_PreSetFieldValue = iReturn
End Function
```

See Also
"ClearToQuery Method" on page 186
"SetSearchSpec Method" on page 237
ExecuteQuery2 Method

ExecuteQuery2 returns a set of BusComp 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>■ <strong>ForwardBackward.</strong> 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>■ <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>
<tr>
<td>ignoreMaxCursorSize</td>
<td>■ <strong>TRUE.</strong> Retrieves every row from a business component. This option may result in lower performance.</td>
</tr>
<tr>
<td></td>
<td>■ <strong>FALSE.</strong> Retrieves the number of rows specified by the MaxCursorSize argument in the CFG file.</td>
</tr>
</tbody>
</table>

Returns

Not applicable

Usage

ExecuteQuery2 is specific to Microsoft SQL Server.

Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile/Dedicated 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.
Syntax

`BusComp.FirstRecord`

<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: 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/Dedicated 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.

```javascript
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);
            ActivateField("Account Id");
            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.

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
            .ActivateField("Account Id")
            .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
        iRtn = CancelOperation
        End If
    BusComp_PreInvokeMethod = iRtn
End Function
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.

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
            While iRecord
                .DeleteRecord
                iRecord = .NextSelected
            Wend
        End With
        iRtn = CancelOperation
    End If
    BusComp_PreInvokeMethod = iRtn
End Function

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

BushComp.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 N: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/Dedicated 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.

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

The following is the equivalent Siebel eScript code.

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

See Also
"GetMVGBusComp Method" on page 204
"GetPicklistBusComp Method" on page 206
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 180. 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 child of the current business component is desired, the Link Specification property for that field must be set to TRUE in Siebel Tools. Otherwise, the parent business component cannot access the value in the child business component. For more information, read Object Types Reference.

The FieldName must be enclosed in double quotes and must be spelled exactly as the field name appears in Siebel Tools, with the correct case; for example,

GetFieldValue("ActivityCreatedByName")

The name 'Person who created the activity', as shown in the status bar, does not work; nor does the column head 'Created By'.

**NOTE:** In Browser Script, GetFieldValue can be used only for the fields exposed in the applet and for the system Id field.

Used With

Browser Script, COM Data Control, COM Data Server, Java Data Bean, Mobile/Dedicated 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.

```vbnet
Function BusComp_PreSetFieldValue (FieldName As String, FieldValue As String) As Integer
    Dim bcOppty As BusComp
    Dim boBusObj As BusObject
    Dim srowid As String
    srowid = GetFieldValue("Id")
    Set boBusObj = TheApplication.GetBusObject("Opportunity")
    Set bcOppty = boBusObj.GetBusComp("Opportunity")
    With bcOppty
        .SetViewMode SalesRepView
        .ActivateField "Sales Stage"
        .SetSearchSpec "Id", srowid
        .ExecuteQuery ForwardOnly
    End With
    Set bcOppty = Nothing
    Set boBusObj = Nothing
End Function
```

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;
}
```

See Also
“ActivateField Method” on page 180
"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

\[
\text{BusComp.GetFormattedFieldValue(FieldName)}
\]

<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:

```plaintext
phone = bc.GetFieldValue("Main Phone Number")
TheApplication.Trace "The number is " & phone
```

Result:

The number is 8869629123

Example 2:

```plaintext
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 22 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/Dedicated 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.

```vba
Sub Button_Click
    Dim DateDiff as Integer
    Dim oBC as BusComp
    Set oBC = me.BusComp
    x = oBC.GetFormattedFieldValue("Start Date")
    y = oBC.GetFormattedFieldValue("Done")
    dx = DateValue(x)
    dy = DateValue(y)
    DateDiff = dy - dx
End Sub
```

**See Also**
"ActivateField Method" on page 180
"GetFieldValue Method" on page 199
"SetFieldValue Method" on page 228
"SetFormattedFieldValue Method" on page 230

**GetLastErrCode Method**
The GetLastErrCode method returns the most recent error code on the business component level.
Syntax

BusComp.GetLastErrorCode

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. The text retrieved using GetLastErrText also includes a Siebel error number that can be used to search Siebel SupportWeb for additional information about the error.

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

GetLastErrorText Method

The GetLastErrorText method returns the last error text message on the business component level.

Syntax

BusComp.GetLastErrorText

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/Dedicated Web Client Automation Server
GetMultipleFieldValues Method

GetMultipleFieldValues returns values for the fields specified in the property set.

Syntax

BusComp.GetMultipleFieldValues(SiebelPropertySet fieldNames, SiebelPropertySet fieldValues)

Arguments

<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 fieldNames argument</td>
</tr>
</tbody>
</table>

Returns

TRUE if success; FALSE if failure

Used With

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

See Also

"SetMultipleFieldValues Method" on page 232

GetMVGBusComp Method

GetMVGBusComp returns the MVG business component associated with the business component field specified by FieldName. This business component can be used to operate on the multi-value group using the normal business component mechanisms.

Syntax

BusComp.GetMVGBusComp(FieldName)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</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>

See Also

"GetLastErrCode Method"
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 should be set to Nothing after use.

**NOTE:** In the context of a many-to-many relationship, you can use Siebel VB to either add a new record, that is, associate a new child record, or insert a record, that is, create a 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/Dedicated 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 191 and “FirstSelected Method” on page 196.

```vbnet
Dim AccntBO as BusObject
Dim AccntBC as BusComp
Dim AddrBC as BusComp
Set AccntBO = TheApplication.GetBusObject("Account")
Set AccntBC = AccntBO.GetBusComp("Account")

With AccntBC
    .SetViewMode SalesRepView
    .ActivateField "Name"
    .ClearToQuery
    .SetSearchSpec "Name", "Hong Kong Flower Shop"
    .ExecuteQuery
    Set AddrBC = .GetMVGBusComp("Street Address")
End With

With AddrBC
    .NewRecord NewAfter
    .SetFieldValue "City", "Denver"
    .WriteRecord
End With

Set AccntBO = Nothing
Set AccntBC = Nothing
Set AddrBC = Nothing
```
GetNamedSearch Method
GetNamedSearch returns the named search specification specified by searchName.

Syntax
BusComp.GetNamedSearch(searchName)

Arguments
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchName</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/Dedicated Web Client Automation Server, Server Script

See Also
"GetSearchExpr Method" on page 208
"GetSearchSpec Method" on page 209
"SetNamedSearch Method" on page 233

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

Usage

The returned pick business component can be used to operate on the picklist. The `GetPickListBusComp` should be destroyed after use by using the `Nothing` function.

**NOTE:** When a record is picked on a constrained picklist using the `GetPickListBusComp` and `Pick` methods, the constraint is active. Therefore, the retrieved picklist business component contains only those records that fulfill the constraint.

**To pick a value from a picklist in Siebel VB**

1. Use `GetPicklistBusComp` to create an instance of the pick list business component.
2. Navigate in the picklist business component to the record you want to pick.
3. Use `Pick` to pick the value.
4. Use `Set objBCPickList = Nothing` to explicitly destroy the picklist business component instance.

Used With

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

Example

The following example is in Siebel eScript.

```eScript
if (this.GetFieldValue("City") == "San Mateo")
{
    var oBCPick = this.GetPicklistBusComp("State");
    with (oBCPick)
    {
        ClearToQuery();
        SetSearchSpec("Value", "CA");
        ExecuteQuery(ForwardOnly);
        if(FirstRecord())
            Pick();
```
The following example is for Java Data Bean. It selects a product from a picklist.

```java
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.activateField("Name");
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.

```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
```

See Also
- "FirstSelected Method" on page 196
- "GetMVGBusComp Method" on page 204

**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 "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/Dedicated Web Client Automation Server, Server Script

See Also
"GetNamedSearch Method" on page 206
"GetSearchSpec Method"
"SetSearchExpr Method" on page 235

GetSearchSpec Method
GetSearchSpec returns the search specification for the field specified by the FieldName argument.

Syntax
BusComp.GetSearchSpec(FieldName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</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/Dedicated Web Client Automation Server, Server Script

See Also
"GetNamedSearch Method" on page 206
"GetSearchExpr Method" on page 208
"SetSearchSpec Method" on page 237

**GetUserProperty Method**

GetUserProperty returns the value of a named user property.

**Syntax**

`BusComp.GetUserProperty(propertyName)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>propertyName</code></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/Dedicated Web Client Automation Server, Server Script

**See Also**
"SetUserProperty Method" on page 243
**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, read “SetViewMode Method” on page 245.

**Syntax**

`BusComp.GetViewMode`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>Where <code>mode</code> 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 245.</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 some internal rules.

**Used With**

COM Data Control, COM Data Server, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script
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 “InvokeMethod Methods” on page 214.</td>
</tr>
<tr>
<td>methodArgs</td>
<td>A single string or a string array (object interfaces) containing arguments to 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, methArg2, ..., methArgn</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 property set.

**Usage**

Use InvokeMethod to call methods on a business component object that are not exposed directly through the object interface.

Specialized methods are typically methods implemented in applet or business component classes other than CSSFrame and CSSBusComp, respectively, that is, specialized classes.

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

**Used With**

COM Data Control, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script
Example
The following example is in Siebel VB.

```vbnet
(general) (declarations)
Option Explicit

Sub Button1_Click
Me.BusComp.InvokeMethod "Select All"
End Sub

Function BusComp_PreInvokeMethod (MethodName As String) As Integer
BusComp_PreInvokeMethod = ContinueOperation
On Error GoTo Leave
If MethodName = "Select All" Then
    Dim oCurBC as BusComp
    Set oCurBC = Me
    If oCurBC is not nothing Then
        oCurBC.ClearToQuery
        oCurBC.ExecuteQuery
        BusComp_PreInvokeMethod = CancelOperation
    End If
End If
Leave:
End Function
```

The following is the equivalent example in Siebel eScript.

```javascript
function BusComp_PreInvokeMethod (MethodName)
{
    var iReturn = ContinueOperation;
    if (Clib.errno() != 0)
        return(CancelOperation);
    if (MethodName = "Select All")
    {
        var oCurBC = this;
        if (oCurBC != null)
        {
            oCurBC.ClearToQuery();
            oCurBC.ExecuteQuery();
            return(CancelOperation);
        }
    }
    return (iReturn);
}
```
InvokeMethod Methods

Siebel applications provide multiple methods for manipulating files stored in the Siebel File System. These methods may be invoked using server script (Siebel VB, eScript) or using one of our programmatic interfaces (Mobile/Dedicated Web Client Automation Server – connected mode only, COM Data Control, Java Data Bean). The methods available for manipulating the file system always store or retrieve the file to and from the local file system. For example, if you construct a Java client using the Java Data Bean to manipulate the file system, all files must be accessible from the Siebel Server. You can use UNC naming conventions (for example: `\server\dir\file.txt`) or standard DOS directories (for example: `D:\dir\file.txt`) for file access, but the UNC path or mounted file system must be accessible to the Siebel Server. These methods do not serialize the files from a remote client and place them in the Siebel file system.

Methods that manipulate files are available for business components whose Class is ‘CSSBCFile’. The methods can be accessed using COM Data Control, Java Data Bean, Mobile/Dedicated Web Client Automation Server, and Server Script.

The following methods are available for use with InvokeMethod:

- "CreateFile"
- "GenerateProposal" on page 215
- "GetFile" on page 215
- "PutFile" on page 216
- "RefreshRecord" on page 217
- "SetAdminMode" on page 217

CreateFile

To create a file in the Siebel file system from an external source, use the business component CreateFile method. Before calling CreateFile, make sure that a new business component record has been created using the NewRecord method for the business component.

Syntax

```
BusComp.InvokeMethod("CreateFile", SrcFilePath, KeyField, KeepLink)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SrcFilePath</code></td>
<td>The fully qualified path of the file on the Siebel Server or Mobile Web Client.</td>
</tr>
<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> field in the Account Attachment business component.</td>
</tr>
<tr>
<td><code>KeepLink</code></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 values of “Success” or “Error” depending on whether or not the operation succeeded.

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

GenerateProposal
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)</td>
</tr>
<tr>
<td></td>
<td>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
Browser Script, COM Data Control, COM Data Server, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script

GetFile
Obtains a file from the Siebel file system and places that file on the local file system of the Siebel Server or Mobile Client. Note that you must be properly positioned on the desired file attachment record to get the file and have it placed on the local file system's temporary directory.
Syntax

BusComp.InvokeMethod("GetFile", KeyFieldName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyFieldName</td>
<td>The name of the field in the business component that contains the File Name. For example: AccntFileName field 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

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

PutFile

Updates a file in the Siebel file system with a newer file. Note that you must be properly positioned on the desired file attachment record to update the file in the file system.

Syntax

BusComp.InvokeMethod("PutFile", SrcFilePath, KeyFieldName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SrcFilePath</td>
<td>This is the fully qualified path of the file on the Siebel Server or Mobile Web Client.</td>
</tr>
<tr>
<td>KeyFieldName</td>
<td>This is the name of the field in the business component that contains the FileName. For example: AccntFileName 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 248.
**RefreshRecord**
This method refreshes the business component, which triggers an update of the business component fields in the client display and positions the cursor on the context record.

**Syntax**
```java
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

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

---

**SetAdminMode**
This method is particularly useful if you need to replicate the behavior enforced by the ‘Admin’ property of the View object by disabling all visibility rules for the business component.

**Syntax**
```java
BusComp.InvokeMethod("SetAdminMode", flag)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flag</td>
<td>“TRUE” or “FALSE”. Flag to specify whether the business component should be executed in Admin mode.</td>
</tr>
</tbody>
</table>

**Returns**
Not Applicable

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

LastRecord moves to the last record in the business component.

Syntax

\[ BusComp\text{.}LastRecord \]

<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 ActiveX, COM, Java Data Bean, Siebel eScript.

Used With

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

Example

The following example is for Mobile/Dedicated Web Client Automation Server.

```vba
Private Sub LastRecord_Click()
    Dim errCode As Integer
    Dim oBusComp as SiebelBusComp
    FieldValue.Text = ""
    HourGlassStart
    oBusComp.LastRecord errCode
    If errCode = 0 Then
        FieldValue.Text = oBusComp.GetFieldValue(FieldName.Text, _
            errCode)
    End If
    HourGlassStop
    Status.Text = SiebelApplication.GetLastErrText
End Sub
```

See Also

"FirstRecord Method" on page 193
"NextRecord Method" on page 220

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>Not applicable</td>
<td></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/Dedicated 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(whereIndicator)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>whereIndicator</td>
<td>Predefined constant or corresponding integer indicating where the new row is added. This value should be one of the following:</td>
</tr>
<tr>
<td></td>
<td>0 (or NewBefore)</td>
</tr>
<tr>
<td></td>
<td>1 (or NewAfter)</td>
</tr>
<tr>
<td></td>
<td>2 (or NewBeforeCopy)</td>
</tr>
<tr>
<td></td>
<td>3 (or NewAfterCopy)</td>
</tr>
<tr>
<td></td>
<td>With Java Data Bean the values are:</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.

```
Object.NewRecord NewBeforeCopy
```

To place the copy after the original record, use the following command.

```
Object.NewRecord NewAfterCopy
```

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

**Example**
The following example is in Siebel VB.

```
Dim oBusObj as BusObject
Dim oBC as BusComp

Set oBusObj = theApplication.ActiveBusObject
Set oBC = oBusObj.GetBusComp("Action")
oBC.NewRecord NewAfter
oBC.SetFieldValue "Type", "To Do"
oBC.SetFieldValue "Description", "Find Decision Makers")
oBC.WriteRecord

set oBC = Nothing
set oBusObj = Nothing
```

**NextRecord 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**
`BusComp.NextRecord`

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

"220 ■ Siebel Object Interfaces Reference" Version 7.7, Rev. C
Returns
An integer in Siebel VB; a Boolean in Siebel eScript and COM: 1 if the current record was moved to the next record, 0 if the current record was already the last record.

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

Example
The following example is in Siebel eScript. For the equivalent Siebel VB example, read "FirstRecord Method" on page 193.

```javascript
var i = 0;
var isRecord;
with (this)
{
    ClearToQuery();
    setSearchSpec("Name", "*");
    ExecuteQuery(ForwardBackward);
    isRecord = FirstRecord();
}
while (isRecord)
{
    i++;
    isRecord = BusComp.NextRecord();
}
```

See Also
"FirstRecord Method" on page 193

NextSelected Method
NextSelected moves the focus to the next record of the current multiple selection.

Syntax
```javascript
BusComp.NextSelected
```

<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: 1 if there is another record in the multiple selection, 0 otherwise.
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.

Used With

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

Example

The following example is in Siebel VB. For another example, read “ExecuteQuery Method” on page 191.

```
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:** In Siebel Business Applications v.7.5.3 and later releases, 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/Dedicated Web Client Automation Server, Server Script

**Example**

This Siebel VB example sorts the values in the Sales Stage field.

```
Sub BusComp_NewRecord
  Dim oBC as BusComp
  set oBC = me.GetPickListBusComp("Sales Stage")
  With oBC
    .ClearToQuery
    .ActivateField "Sales Stage Order"
    .SetSortSpec "Sales Stage Order"
    .ExecuteQuery ForwardOnly
    if .FirstRecord then .Pick
  End With
  set oBC = Nothing
End Sub
```

The following is the equivalent example in Siebel eScript.

---

**Interfaces Reference**  ■  Business Component Methods

**Siebel Object Interfaces Reference**  Version 7.7, Rev. C  ■  223
```javascript
function BusComp_NewRecord ()
{
    var oBC = this.GetPickListBusComp("Sales Stage");
    with (oBC)
    {
        ClearToQuery();
        ActivateField("Sales Stage Order");
        SetSortSpec("Sales Stage Order");
        ExecuteQuery(ForwardOnly);
        if (FirstRecord())
            Pick();
    }
    oBC = null;
}
```

See Also

"GetPicklistBusComp Method" on page 206

**PreviousRecord Method**

PreviousRecord moves to the previous record in the business component, invoking any associated Basic events.

**Syntax**

`BusComp.PreviousRecord`

<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; Siebel eScript, a Boolean in COM, and ActiveX: 1 or nonzero if the current record was moved to the previous record, 0 if the current record was already the first record.

**Usage**

PreviousRecord may be used only on a business component that has been queried using the FowardBackward CursorMode.

**Used With**

COM Data Control, COM Data Server, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script
Example
The following example is for Mobile/Dedicated Web Client Automation Server. *SiebelApplication* is an Application instance.

```vbnet
Option Explicit

Private Sub PreviousRecord_Click()
    Dim errCode As Integer
    Dim oBusComp as BusComp
    FieldValue.Text = ""
    HourClassStart
    oBusComp.PreviousRecord errCode
    If errCode = 0 Then
        FieldValue.Text = oBusComp.GetFieldValue(FieldName.Text, _
            errCode)
    End If
    HourClassStop
    Status.Text = SiebelApplication.GetLastErrText
End Sub
```

See Also
“ExecuteQuery Method” on page 191

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/Dedicated Web Client Automation Server, Server Script

Example
The following Siebel VB code fragment shows how RefineQuery might be used.

```vbnet
me.ActivateField "Status"
me.SetSearchSpec "Status", "Open"
me.ClearToQuery
me.ExecuteQuery
me.RefineQuery
me.SetSearchSpec "Substatus", "Assigned"
me.ExecuteQuery
```

See Also
"ClearToQuery Method" on page 186
"GetNamedSearch Method" on page 206

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>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
```
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");
        }
        // Be sure to release the business component and its resources on the server side
        m_busComp.release();
        // release the resources occupied by Siebel Bus Object and Siebel Data Bean after their use.
    }
}
```
// 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());
}

See Also
“Logoff Method” on page 149

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 180. For applications in standard interactivity mode, write the record immediately after using SetFieldValue by calling WriteRecord.

*FieldName* must be enclosed in double quotes, and must be spelled exactly as the field name appears in Siebel Tools (not in the status line of the application or the column head), with the correct case; for example,

```
SetFieldValue 'Name', 'Acme'
```

*FieldValue* must not have a length that exceeds the defined length of the field. For example, passing a 20 character string into a field that is defined as being 16 characters long results in the runtime error "Value too long for field 'xxxxx' (maximum size nnn)." A good practice is to check the length of the string against the length of the destination field before using SetFieldValue.

To set a field to null, follow this example.

```
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 picklistbuscomp.Pick to emulate the user picking the record.

**NOTE:** SetFieldValue cannot be used with calculated fields and cannot be used recursively.

**Used With**

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

**Example**

The following example is in Siebel VB.

```vbnet
Dim CurrOppty as BusComp
Set CurrOppty = Me
If Val(CurrOppty.GetFieldValue("Rep %")) < 75 Then
    CurrOppty.SetFieldValue "Rep %", "75"
End If
```

The following is the equivalent example in Siebel eScript.

```javascript
var CurrOppty = this;
if (ToInteger(CurrOppty.GetFieldValue("Rep %")) < 75)
    CurrOppty.SetFieldValue("Rep %", "75");
```
See Also
“ActivateField Method” on page 180
“SetFormattedFieldValue Method”
“Pick Method” on page 223
“GetPicklistBusComp Method” on page 206

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/Dedicated 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
Option Explicit

Dim OpportunityBO as BusObject, StageBC as BusComp
Dim OppStageId as String, SalesRep as String, Stage as String
Dim StagePrev As String, StageDate as String, StageDatePrev as String
Dim Dx as Double, Dy as Double, Diff as Double, DiffStr as String
Dim OppID As String, OppStageId as String, StageID As String
Dim SalesStageBO as BusObject, SalesStageBC as BusComp
```
Set SalesStageBO = TheApplication.GetBusObject("Sales Cycle Def")
Set SalesStageBC = SalesStageBO.GetBusComp("Sales Cycle Def")

With SalesStageBC
    .SetViewMode AllView
    .ClearToQuery
    .SetSearchSpec "Sales Cycle Stage", StagePrev
    .ExecuteQuery ForwardOnly
    .FirstRecord
    StageId = .GetFieldValue("Id")
End With

' Instantiate stage BC
Set StageBC = OpportunityBO.GetBusComp("Opportunity Stage")

' Check that we do not already have a record for the stage

With StageBC
    .SetViewMode AllView
    .ClearToQuery
    .SetSearchSpec "Sales Stage Id", StageId
    .ExecuteQuery ForwardOnly

' Proceed further only if we do not already have record
' opportunity sales stage

If (.FirstRecord = 0) Then
    ' Create a new stage record and write it out
    .NewRecord 1
    ' Record Id for future use
    OppStageId = .GetFieldValue("Id")
    .SetFieldValue "Opportunity Id", OppId
    .SetFieldValue "Sales Stage Id", StageId
    .SetFieldValue "Sales Rep", SalesRep
    .SetFormattedFieldValue "Entered Date", StageDatePrev
    .SetFormattedFieldValue "Left Date", StageDate
    Dx = DateValue (StageDatePrev)
    Dy = DateValue (StageDate)
    Diff = Dy - Dx
    DiffStr = Str(Diff)
    .SetFieldValue "Days In Stage", DiffStr
    .WriteRecord
End If
End With

See Also
"ActivateField Method" on page 180
"SetFieldValue Method" on page 228
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".

Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile/Dedicated 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) ;
b.c.WriteRecord;
```

The following Java example sets multiple fields using SetMultipleFieldValues
SiebelDataBean sieb_dataBean = null;
SiebelBusObject sieb_busObject = null;
SiebelBusComp sieb_busComp = null;
SiebelPropertySet ps = null;

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());
}

See Also
"ActivateMultipleFields Method" on page 182
"GetMultipleFieldValues Method" on page 204

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. Once set, the named search specification is applied every time ExecuteQuery is called. ClearToQuery does not clear the named search specification. To clear it, explicitly set the searchSpec argument to "". Note that when a new instance of the BusComp is created, the named search specification is cleared.

The searchSpec argument assigned to SetNamedSearch is the same argument that is used after the equal sign in a predefined query. The maximum length of a predefined query is 2000 characters. For details on how to set up the search specification, read "SetSearchExpr Method" and "SetSearchSpec Method" on page 237.

Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile/Dedicated 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", "[Status] LIKE 'Candidate'
    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.

See Also
"GetNamedSearch Method" on page 206
"SetSearchSpec Method" on page 237

SetSearchExpr Method

SetSearchExpr sets an entire search expression on the business component, rather than setting one search specification per field. Syntax is similar to that on the Predefined Queries screen.

Syntax
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.

The maximum length of a predefined query is 2000 characters. The argument assigned to SetSearchExpr is the same as that used after the equal sign in a predefined query. For example, the first line following is a search specification in a predefined query; the second is the equivalent search specification used with the various interface methods. Note that Name is a field on the business component and therefore must be enclosed in brackets, [ ].

'Account'.Search = "[Name] ~ LIKE "'A. C. Parker'"
BC.SetSearchExpr @[Name] ~ LIKE "'A. C. Parker'"
If field values have search keywords such as NOT, AND, and OR, use two pairs of double quotes around the field value. For example, if a field Sub-Status can have the string "Not an Issue" as a field value, then use the following syntax to avoid an SQL error:

```java
substatus = GetFieldValue("Sub-Status")
searchst = "[Value] = "" & substatus & ""
BC.SetSearchExpr searchst
```

The following syntax generates an SQL error.

```java
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 SetSortSpec 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/Dedicated Web Client Automation Server, Server Script

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

```java
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();
```

**See Also**
"ClearToQuery Method" on page 186
"ExecuteQuery Method" on page 191
"SetSearchSpec Method" on page 237
"SetSortSpec Method" on page 241
SetSearchSpec Method

SetSearchSpec sets the search specification for a particular field. This method must be called before ExecuteQuery.

Syntax

\[ \text{BusComp.SetSearchSpec FieldName, searchSpec} \]

<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 unpredicted compound search specification on a business component, it is recommended to call ClearToQuery before calling 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:

  ```java
  myBusComp.SetSearchSpec("Status", '<> 'Renewal');
  myBusComp.SetSearchSpec("Status", '<> 'Dropped');
  ```

  results in the following WHERE clause:

  ```sql
  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:

  ```java
  myBusComp.SetSearchSpec("Type", '<> 'Renewal');
  myBusComp.SetSearchSpec("Status", '<> 'Sold' AND [Status] <> 'Cancelled' AND [Status] <> 'Renewed');
  ```

  results in the following WHERE clause:

  ```sql
  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:

BC.SetSearchSpec "Status", ' <> 'Closed' AND ([Owner] = LoginName () OR [Refer To] = LoginName ()) OR ([Owner] IS NULL AND [Support Group] = 'TS-AE')"

Using special characters. If the search specification contains any of the following characters.

= > < {} ~ " '

it must be enclosed in quotes. This rule applies to operators that are part of the search expression as well as text to search for. If the search expression contains quotes, those quotes must be doubled. For example, in the preceding line of code, notice that the entire search specification is enclosed in double quotes, whereas fields and values referred to within the specification each have single quotes.

If the search object includes a single double quote, that quote must be doubled; for example, if you wanted to search for text containing:

"We must"

the search specification would take this form:

'SetSearchSpec 'Comments', " We must"
so that the initial quote is doubled, and the string containing it is placed within single quotes, and the entire expression, including the single quotes, is placed within double quotes.

If the search specification includes single quotes (including apostrophes), the expression must be placed within single quotes, apostrophes must be doubled, and double quotes must be placed around the entire string. Thus, for example, if you wanted to search for “Phillie’s Cheese Steaks” in the Name field, you would have to enter the specification as follows:

```
SetSearchSpec "Name", "'Phillie''s Cheese Steaks"
```

**NOTE:** eScript and Browser Script require backslashes instead of double quotes for marking special characters. For example:

```
SetSearchSpec("Comments", "'"We must\"""); and
SetSearchSpec("Name", "'Phillie\"\"s Cheese Steaks\"");
```

**Searching for text in non-text fields.** If the search expression queries a field of any type other than text, or if it is an expression other than a field-level query, text must be placed within quotes if it contains any characters other than the following:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz _ ? ", ', [ 
```

As with text field search expressions, quotes must be doubled.

**Retrieving all records.** To retrieve all records efficiently, use ClearToQuery followed by ExecuteQuery, without using SetSearchSpec.

**Searching for a null field.** To search for null fields, use the following form:

```
SetSearchSpec "Account", "is NULL"
```

If your search specification requests an empty string, then the search returns every record. For example:

```
SetSearchSpec "Account", ""
```

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/Dedicated Web Client Automation Server, Server Script

**Example**
For Siebel VB examples, read “FirstRecord Method” on page 193, “SetFormattedFieldValue Method” on page 230, and “BusComp_PreQuery Event” on page 259. For a Siebel eScript example, read “ClearToQuery Method” on page 186.

**Example**
This Siebel VB code searches for a contact by name and then navigates to the record displayed in a view.
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.ActivateField "Last Name"
NewComp.ActivateField "First Name"
NewComp.ActivateField "Account Id"
NewComp.ClearToQuery
NewComp setSearchSpec "First Name", RecId1
NewComp setSearchSpec "First Name", RecId2
NewComp setSearchSpec "Account Id", RecId3
NewComp.ExecuteQuery ForwardBackward

TheApplication.GotoView TargetView , NewBusObj

End Sub

The following example is in Siebel eScript.

```javascript
var oAccntBO = TheApplication().GetBusObject("Account");
var oAccntBC = oAccntBO.GetBusComp("Account");
var oAddrBC;

with (oAccntBC)
{
    SetViewMode(SalesRepView);
    ActivateField("Name");
    ClearToQuery();
    setSearchSpec("Name", "Hong Kong Flower Shop");
    ExecuteQuery();
    oAddrBC = GetMVGBusComp("Street Address");
}

with (oAddrBC)
{
```
NewRecord(NewAfter);
SetFieldValue("City", "Denver");
WriteRecord();
}

oAddrBC = null;
oAccntBC = null;
oAccntBO = null;

See Also
"ExecuteQuery Method" on page 191
"ClearToQuery Method" on page 186
"SetSearchExpr Method" on page 235
"SetSortSpec Method"

SetSortSpec Method
SetSortSpec sets the sorting specification for a query.

Syntax
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/Dedicated 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.

```vbscript
Function BusComp_PreQuery As Integer
With Me
  .ActivateField("Account")
  .ActivateField("Account Location")
  .ClearToQuery
  .SetSortSpec "Account(DESCENDING), Account Location(ASCENDING)"
  .ExecuteQuery
End With

  BusComp_PreQuery = ContinueOperation
End Function
```

The following is the equivalent example in Siebel eScript.

```javascript
Function BusComp_PreQuery
with (this) {  
     ActivateField("Account");
     ActivateField("Account Location");
     ClearToQuery();
     SetSortSpec("Account(DESCENDING), Account Location(ASCENDING)");
     ExecuteQuery();
 }

return (ContinueOperation);
```
SetUserProperty Method

Sets the value of a named business component user property. The user properties are similar to instance variables of a BusComp.

Syntax

\[ \text{BusComp.SetUserProperty } \text{propertyName}, \text{newValue} \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>propertyName</td>
<td>String containing the name of the user property to set</td>
</tr>
<tr>
<td>newValue</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 Server, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script

Example

The following example is in Siebel VB.

```vb
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');
    }
}
```

**See Also**

"GetUserProperty Method" on page 210
SetViewMode Method

SetViewMode sets the visibility type for the business component. This is used prior to a query.

Syntax

`BusComp.SetViewMode mode`

where `mode` is a Siebel ViewMode constant or its corresponding integer value. The constants shown are defined in three environments.

<table>
<thead>
<tr>
<th>SiebelViewMode 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>
</tbody>
</table>
### SiebelViewMode Constant

<table>
<thead>
<tr>
<th>SiebelViewMode Constant</th>
<th>Integer Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<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>
<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>
</tbody>
</table>
| SubOrganizationView     | 9             | If the business component on which the view is based uses single organization access control, then this constant displays records associated directly with the user’s active organization or with a descendent organization. Descendent organizations are defined by the organization hierarchy. To use this visibility applet type, the business component must have a view mode with an Owner Type of Organization.  
   
   If the business component on which the view is based uses multiple organization access control, then this constant displays records for which the user’s active organization or a descendent organization is the primary organization.  
   
   Example: All Opportunities Across My Organization. Typically used by executives. |

### Returns

Not applicable

### Used With

COM Data Control, COM Data Server, Java Data Bean, Mobile/Dedicated Web Client Automation Server, Server Script
See Also
"GetViewMode Method" on page 211

Example
The following example is in Siebel VB. For another example, see “BusComp_PreDeleteRecord Event” on page 256.

```vbnet
Option Explicit
Dim oBO as BusObject
Dim oBC as BusComp

Set oBO = theApplication.GetBusObject(Me.BusObject.Name)
Set oBC = oBO.GetBusComp(Me.Name)
With oBC
   .SetViewMode SalesRepView
   .ClearToQuery
   .ActivateField "Name"
   .SetSearchSpec "Name", Me.GetFieldValue("Name")
   .SetSearchSpec "Id", "<> " & Me.GetFieldValue("Id")
   .ExecuteQuery ForwardOnly
   If .FirstRecord Then
      theApplication.Trace("Entry for name " & Me.GetFieldValue("Name") & " exists.")
   End If
End With

Set oBC = Nothing
Set oBO = Nothing
```

The following is the equivalent example in Siebel eScript.

```escript
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();
   ActivateField("Name");
   SetSearchSpec("Name", this.GetFieldValue("Name"));
   SetSearchSpec("Id", "<> " + this.GetFieldValue("Id");
   ExecuteQuery(ForwardOnly);
   if (FirstRecord)
      TheApplication().Trace("Entry for name " + this.GetFieldValue("Name") + " exists.");
}

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
```

**Argument**

<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/Dedicated Web Client Automation Server, Server Script

**See Also**

"NewRecord Method" on page 219

**WriteRecord Method**

Commits to the database any changes made to the current record.
Syntax

$oBusComp.WriteRecord

| Argument | Description
<table>
<thead>
<tr>
<th></th>
<th></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/Dedicated 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 204 and “NewRecord Method” on page 219.

```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
                .SetFieldValue "Type", "Event"
                .SetFieldValue "Description", "THRU SVB, Stage changed to 02"
                .SetFieldValue "Done", Format(Now(), "mm/dd/yyyy hh:mm:ss")
                .SetFieldValue "Status", "Done"
                .WriteRecord
            End With
            set oBCact = Nothing
        end if
    End Select
End Sub
```
Business Component Events

The following topics describe business component events:

- “BusComp_Associate Event” on page 250
- “BusComp_ChangeRecord Event” on page 251
- “BusComp_CopyRecord Event” on page 252
- “BusComp_DeleteRecord Event” on page 253
- “BusComp_InvokeMethod Event” on page 254
- “BusComp_NewRecord Event” on page 254
- “BusComp_PreAssociate Event” on page 255
- “BusComp_PreCopyRecord Event” on page 255
- “BusComp_PreDeleteRecord Event” on page 256
- “BusComp_PreGetFieldValue Event” on page 257
- “BusComp_PreInvokeMethod Event” on page 258
- “BusComp_PreNewRecord Event” on page 259
- “BusComp_PreQuery Event” on page 259
- “BusComp_PreSetFieldValue Event” on page 260
- “BusComp_PreWriteRecord Event” on page 262
- “BusComp_Query Event” on page 263
- “BusComp_SetFieldValue Event” on page 265
- “BusComp_WriteRecord Event” on page 265

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

**See Also**
“BusComp_NewRecord Event” on page 254

---

**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.

**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**

```java
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**

```java
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 223.

**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**
BusComp_PreAssociate

<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_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></td>
<td>Not applicable</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></td>
<td>Not applicable</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.
Used With
Server Script

Example
This Siebel VB example prevents the deletion of an account that has associated opportunities.

```vbnet
(general) (declarations)
Option Explicit

Function BusComp_PreDeleteRecord As Integer
    Dim iReturn as integer
    Dim oBC as BusComp
    Dim oBO as BusObject
    Dim sAcctRowId as string

    iReturn = ContinueOperation
    sAcctRowId = me.GetFieldValue("Id")
    set oBO = TheApplication.GetBusObject("Opportunity")
    set oBC = oBO.GetBusComp("Opportunity")

    With oBC
        .SetViewMode AllView
        .ActivateField "Account Id"
        .ClearToQuery
        .SetSearchSpec "Account Id", sAcctRowId
        .ExecuteQuery ForwardOnly
        If (.FirstRecord) = 1 Then
            RaiseErrorText("Opportunities exist for the Account - _
            Delete is not allowed")
            iReturn = CancelOperation
        End If
    End With

    BusComp_PreDeleteRecord = iReturn

    Set oBC = Nothing
    Set oBO = Nothing
End Function
```

**BusComp_PreGetFieldValue Event**
The PreGetFieldValue event is called when the value of a business component field is accessed.
## Business Component Events

### 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.

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
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 before a value is pushed down into the business component from the user interface or through a call to SetFieldValue.
Syntax
BusComp_PreSetFieldValue(FieldName, FieldValue)

<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 147, and "ExecuteQuery Method" on page 191.

```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
```

Argument

- **FieldName**
  - String containing the name of the changed field
- **FieldValue**
  - String containing the changed value
BusComp_PreSetFieldValue = CancelOperation
Else
    BusComp_PreSetFieldValue = ContinueOperation
End If
End If
End Function

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);
            return (CancelOperation);
        }
        else
        {
            return (ContinueOperation);
        }
    }
    else
    {
        return (ContinueOperation);
    }
}
```

**BusComp_PreWriteRecord Event**

The PreWriteRecord event is called before a row is written out to the database. The event may perform any final validation necessary before the actual save occurs.

**Syntax**

BusComp_PreWriteRecord

<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 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.

Used With

Server Script

Example

Function BusComp_PreWriteRecord As Integer
    ' This code resets the probability before the write
    ' if necessary
    if Me.GetFieldValue("Sales Stage") LIKE '07*' then
        ' Resets the Probability to 75 if less than 75
        if Val(Me.GetFieldValue("Rep %")) < 75 then
            Me.SetFieldValue "Rep %", "75"
        end if
    end if
    BusComp_PreWriteRecord = ContinueOperation
End Function

BusComp_Query 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></td>
<td>Not applicable</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.

```
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 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 BusComp 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 262).

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>

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.

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 267
- “GetLastErrText Method” on page 268
- “Name Method” on page 269
- “Release Method” on page 269

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>BusCompName</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.

The Nothing function should be used to destroy the instantiated business component when it is no longer necessary.

**NOTE:** In Browser Script, the GetBusComp() method can only access business components in the current view; in Server Script, the GetBusComp() method can access every business component that has been instantiated in the active business object.

Used With

Browser Script, COM Data Control, COM Data Server, Java Data Bean, Server Script

Examples

To access a business component in the UI context:

\[
\begin{align*}
  &\text{var ActiveBO = TheApplication().ActiveBusObject();} \\
  &\text{var ConBC = ActiveBO.GetBusComp("Contact");}
\end{align*}
\]

To access a business component in the non-UI context:

\[
\begin{align*}
  &\text{var BO = TheApplication().GetBusObject("Account");} \\
  &\text{var ConBC = BO.GetBusComp("Contact");}
\end{align*}
\]

GetLastErrorCode Method

The GetLastErrorCode method returns the last error code.
Syntax

\texttt{oBusObj.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

COM Data Control, Mobile/Dedicated Web Client Automation Server

See Also

“GetLastErrText Method” on page 268

GetLastErrText Method

The GetLastErrText method returns the last error text.

Syntax

\texttt{oBusObj.GetLastErrText}

<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 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/Dedicated Web Client Automation Server

**See Also**
“GetLastErrCode Method” on page 267

**Name Method**
The Name method returns the name of the business object.

**Syntax**

```
obusObj.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 object name

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

**Example**
For an example, read “Name Method” on page 218.

**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></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
...

// Use the business Object
// Release the business object resources
...

busObj.release();
}
```

Business Service Methods
In the method descriptions, the placeholder `oSService` represents a business service instance:

- “GetFirstProperty Method”
- “GetLastErrCode Method” on page 272
- “GetLastErrText Method” on page 273
- “GetNextProperty Method” on page 273
- “GetProperty Method” on page 275
- “InvokeMethod Method” on page 276
- “Name Method” on page 277
- “PropertyExists Method” on page 278
- “Release Method” on page 278
GetFirstProperty Method

This method retrieves the name of the first property of a business service.

**Syntax**

```
aService.GetFirstProperty()
```

**Argument**

<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 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/Dedicated 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 != null && propSetName != '')
    {
        count++;
        propSetName = busService.GetNextProperty();
    }
}
```java
return count;
}
```

The following example is in Java.

```java
public int countPropSets(SiebelService busService)
{
    int count = 0;
    try
    {
        String propSetName = busService.getFirstProperty();
        while(propSetName != null && propSetName != "")
        {
            count++;
            propSetName = busService.getNextProperty();
        }
    }
    catch(SiebelException sExcept)
    {
        return 0;
    }
    return count;
}
```

See Also

"GetNextProperty Method" on page 273
"GetProperty Method" on page 275
"SetProperty Method" on page 280

**GetLastErrorCode Method**

The GetLastErrorCode method returns the most recent error code.

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

Used With
Mobile/Dedicated Web Client Automation Server

See Also
“GetLastErrText Method”

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

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
Mobile/Dedicated Web Client Automation Server

See Also
“GetLastErrCode Method” on page 272

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 a null string (""") if no more properties exist.

Usage

After retrieving the name of the first property with GetFirstProperty, the GetNextProperty method should be used in a loop, to be terminated when a null string (""") is returned. When property names have been retrieved, they can be used as 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/Dedicated 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 != null && 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 != null && propSetName != "")
{
    count++;
    propSetName = busService.getNextProperty();
}

} catch(SiebelException sExcept)
{
    return 0;
}
return count;

See Also
"GetFirstProperty Method” on page 300
"GetProperty Method”
"SetProperty Method” on page 280

GetProperty Method
The GetProperty method returns the value of the property whose name is specified in its argument.

Syntax
oService.GetProperty(propName)

Argument | Description
--- | ---
propName | The name of the property whose value is to be returned

Returns
A string containing the value of the property indicated by propName or NULL if the property does not exist.

Usage
You must know the name of a property to retrieve its value. To retrieve property names, use the GetFirstProperty and GetNextProperty methods.

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

```javascript
oService.InvokeMethod(methodName, InputArguments, OutputArguments)
```

**Siebel VB Syntax**

```vbscript
oService.InvokeMethod methodName, InputArguments, OutputArguments
```

**Argument** | **Description**
--- | ---
`methodName` | A string representing the name of the method to execute
`InputArguments` | A property set containing the arguments required by the method
`OutputArguments` | A property set containing the arguments to be returned by the method (passed by reference)

**Browser Script Syntax**

```javascript
outputPropSet=Service.InvokeMethod(MethodName, inputPropSet)
```

**Argument** | **Description**
--- | ---
`methodName` | The name of the method
`inputPropSet` | A property set containing the method arguments
`outputPropSet` | A property set containing the output arguments of the Property Set

**Returns**

Not applicable

**Usage**

Built-in business services work the same way as business component invoke methods. That is, you can call specialized methods on the service that are not exposed directly through the object interface.
Run-time business services can hold user-defined methods, which must be implemented in scripts written in Siebel VB or Siebel eScript. The scripts must be written in these languages within Siebel Tools; however, they can be called through external interfaces.

Although the InvokeMethod function does not return a value, the properties in the `OutputArguments` property set may have been modified.

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

**Used With**

**See Also**
"Service_InvokeMethod Event" on page 281
"Service_PreInvokeMethod Event" on page 284

**Name Method**
The Name property contains the name of the service.

**Syntax**
```
Service.Name
```

**Argument**
<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<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**

```plaintext
oService.PropertyExists(propName)
```

**Argument**  
**Description**

<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 of the specified service</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 a null string ("") for nonexistent properties, you should use PropertyExists() in an if statement to determine whether a specific property has been set.

**Used With**

Browser Script, COM Data Server, Java Data Bean, Mobile/Dedicated 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**

```plaintext
oBusSvc.release()
```

**Argument**  
**Description**

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

**Returns**

Not applicable

**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.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 ");
        }
        catch(SiebelException ex)
        {
            System.out.println(Ex.getMessage());
        }
    }
}
```
// 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**

```java
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/Dedicated Web Client Automation Server, Server Script

**See Also**

“PropertyExists Method” on page 278

**SetProperty Method**

This method assigns a value to a property of a business service.
Syntax

\`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>PropertyValue</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/Dedicated Web Client Automation Server, Server Script

Example

For an example, read "Service_PreInvokeMethod Event" on page 284.

See Also

"GetProperty Method" on page 275

Business Service Events

The following topics describe business service events:

- "Service_InvokeMethod Event"
- "Service_PreCanInvokeMethod Event" on page 283
- "Service_PreInvokeMethod Event" on page 284

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>

**Returns**
Not applicable

**Usage**
Although this event does not return a value, it may add properties to, or alter the values of the properties in, the property set `OutputArguments`.

When you invoke business service methods through Browser Script, the business service may be implemented as a browser-based business service (written in JavaScript) or a server-based business service. Initially, the high interactivity mode framework checks if the business service resides in the browser, and if it does not find it, it sends the request to the server for execution.

**NOTE:** Browser Script may invoke a browser-based or server-based business service, but Server Script can only invoke a server-based business service.

**NOTE:** Although the `InvokeMethod` function does not return a value in Server Script, it may modify the properties in the `OutputArguments` property set.

**Used With**
Browser Script, Server Script
Example
This Browser Script example invokes the Shipping Engine business service created in "Service_PreInvokeMethod Event" on page 284 in response to a button click. The InvokeMethod property on the Button is set to "CalcShipping". It gets values from the keyboard through prompts (JavaScript method), passes a property set to the service, and gets return values by means of another property set.

```javascript
function Applet_PreInvokeMethod (name, inputPropSet)
{
  if (name == 'CalcShipping') {
    var svc = theApplication().GetService('Shipping Engine');
    var inputs = theApplication().NewPropertySet();
    var outputs = theApplication().NewPropertySet();

    var size = prompt("Enter the sum of H+W+D in inches");
    var shipper = prompt("Enter the shipping company");
    var weight = prompt("Enter the shipping weight in pounds");

    with (inputs) {
      SetProperty("Size", size);
      SetProperty("Shipping Company", shipper);
      SetProperty("Ship Method", shipMethod);
      SetProperty("Weight", weight);
    }

    outputs = svc.InvokeMethod("CalculateShipping", inputs);
    var cost = outputs.GetProperty("Cost");
    var delDate = outputs.GetProperty("Delivery Date");

    TheApplication().SWEAlert("Shipping by " + shipper + ";\n Shipping Cost is " +
                          cost + "\n Estimated delivery date is " +
                          delDate);

    return (CancelOperation);
  }
  else

    return (ContinueOperation);
}
```

See Also
"Service_PreInvokeMethod Event" on page 284

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.
Interfaces Reference ▪ Business Service Events

Server 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</td>
</tr>
<tr>
<td></td>
<td>invoked. Valid values are TRUE and FALSE.</td>
</tr>
</tbody>
</table>

Browser 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.

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>

Returns
“ContinueOperation” or “CancelOperation”
Usage
If implementing a new method, or overriding the behavior of a method implemented in a specialized business service, the script should return CancelOperation to avoid invoking an "Unknown method name" error. As Figure 12 illustrates, this error is predictable if the PreInvokeMethod event is scripted. This occurs because there is no native code to execute in the InvokeMethod event. CancelOperation tells the Siebel application to cancel the remaining operations associated with the event.

NOTE: The example in Figure 12 applies only to new and user-defined methods. For existing standard Siebel methods, it is not necessary to use CancelOperation.

Service_InvokeMethod is rarely scripted, but can be used for such post-operation events as posting a notice to a log when the event completes successfully.

Used With
Browser Script, Server Script
Example
This Siebel VB example creates the new service "Shipping Engine."

Function Service_PreInvokeMethod (MethodName As String, Inputs As PropertySet, Outputs As PropertySet) As Integer
    If MethodName = "CalculateShipping" Then
        Dim sShipper As String, sShipMethod As String
        Dim dWeight As Double, dSize As Double, dCost As Double
        Dim sZone As String, DelDate As Variant
        Dim sCost As String, iReturn As Integer
        iReturn = ContinueOperation
        sShipper = Inputs.GetProperty("Shipping Company")
        sShipMethod = Inputs.GetProperty("Ship Method")
        dWeight = Val(Inputs.GetProperty("Weight"))
        dSize = Val(Inputs.GetProperty("Total Dimensions"))
        iZone = Val(Inputs.GetProperty("Zone"))
        DelDate = DateValue(Now)
        Select Case sShipper
            Case "GlobalEx"
                Select Case sShipMethod
                    Case "Next-Day Air"
                        dCost = 14 + dWeight
                        DelDate = DelDate + 1
                    Case "Second-Day Air"
                        dCost = 11 + (dWeight * .54)
                        DelDate = DelDate + 2
                End Select
            Case "Airline"
                Select Case sShipMethod
                    Case "Next-Day Air"
                        dCost = 5 + (dWeight * .3) + (dSize * .33) + _
                        (Val(sZone) * .5)
                        DelDate = DelDate + 1
                    Case "Second-Day Air"
                        dCost = 4 + (dWeight * .3) + (dSize * .2) + _
                        (Val(sZone) * .3)
                        DelDate = DelDate + 2
                    Case "Ground"
                        dCost = 3 + (dWeight * .18) + (dSize * .1) + _
                        (Val(sZone) * .1)
                        DelDate = DelDate + 2 + Int(Val(sZone) * .8)
                End Select
        End Select
        sCost = Format(dCost, "Currency")
        Outputs.SetProperty "Cost", sCost
        Outputs.SetProperty "Delivery Date", DelDate
        iReturn = CancelOperation
    End If
End Function
End If
    Service_PreInvokeMethod = iReturn
End Function

See Also
“Service_InvokeMethod Event” on page 281

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 287
■ "BusComp Method” on page 288
■ "GetProperty Method” on page 288
■ "GetValue Method” on page 289
■ "Name Method” on page 290
■ "SetProperty Method” on page 292
■ "SetValue Method” on page 293

Applet Method
The Applet method returns the parent applet object for a control.

**Syntax**

```
colorVar.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 218.

**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, Shown, Read Only, Visible.
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

**Returns**

A string containing the object name

**Used With**

Browser Script

**Example**

For an example, read "Name Method" on page 218.

**SetLabelProperty Method**

The SetLabelProperty method sets visual properties of a label.

**Syntax**

controlVar.SetLabelProperty(propName, propValue)

**Argument** | **Description**
---|---
propName | The name of the property to be set, as described in the following table
propValue | The value to assign to the property, as described in the following table

**Returns**

Not applicable

**Usage**

If more than one property is to be set, each must be set in a separate statement.
The following table lists the properties that can be set for a label, and the values that can be assigned to them.

<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 FontColor for a label; for example, green is “00ff00”.</td>
</tr>
<tr>
<td>FontType</td>
<td>string</td>
<td>Determines FontType for a label; for example, “Times Roman”.</td>
</tr>
<tr>
<td>FontSize</td>
<td>string</td>
<td>Determines FontSize for a label; for example, “12 pt”.</td>
</tr>
<tr>
<td>FontStyle</td>
<td>string</td>
<td>Determines FontStyle for a label; for example, “Italic”.</td>
</tr>
<tr>
<td>FontWeight</td>
<td>string</td>
<td>Determines FontWeight 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; for example, 500.</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</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”.

### Example

**Browser Script**

The following code shows the use of SetLabelProperty.

```javascript
function Applet_PreInvokeMethod (name, inputPropSet) {
    // example of changing the Font Size of the Location label
    if (name == 'fontsize') {
        var ctl = this.FindControl("Location");
        var fSize = prompt("Please specify the desired label font size (numeric value only). ");
        ctl.SetLabelProperty("FontSize", fSize);
        return ('CancelOperation');
    }

    // example of changing the Background Color of the Location label
    else if (name == 'bgcolor') {
        var ctl = this.FindControl("Location");
        var bgColor = prompt("Specify the background color of the label. Please enter");
        ctl.SetLabelProperty("BackgroundColor", bgColor);
        return ('CancelOperation');
    }
}
```
```javascript
ctl.SetLabelProperty("BgColor", bgColor);
return "CancelOperation";
}

// example of changing the Font Type of the Location label
else if (name == "fonttype") {
    var ctl = this.FindControl("Location");
    var fontType = prompt("Please specify the font type for the label");
    ctl.SetLabelProperty("FontType", fontType);
    return "CancelOperation";
}

// example of changing the Font Color of the Location label
else if (name == "fontcolor") {
    var ctl = this.FindControl("Location");
    var fontColor = prompt("Specify the font color of the label. Please enter a valid six hexadecimal digit RGB value");
    ctl.SetLabelProperty("FontColor", fontColor);
    return "CancelOperation";
} else
    return "ContinueOperation";
}
```

### SetProperty Method

The `SetProperty` method sets visual properties of a control.

**Syntax**

```javascript
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 control; for example, red is “ff0000”, green is “00ff00”, and blue is “0000ff”.</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 FontColor for a control; for example, green is “00ff00”.</td>
</tr>
<tr>
<td>FontType</td>
<td>string</td>
<td>Determines FontType for a control; for example, “Times Roman”.</td>
</tr>
<tr>
<td>FontSize</td>
<td>string</td>
<td>Determines FontSize for a control; for example, “12 pt”.</td>
</tr>
<tr>
<td>FontStyle</td>
<td>string</td>
<td>Determines FontStyle for a control; for example, “Bold”.</td>
</tr>
<tr>
<td>Height</td>
<td>string</td>
<td>Determines Height for a control, in pixels; for example, “5”.</td>
</tr>
<tr>
<td>Shown</td>
<td>TRUE or FALSE</td>
<td>Is the control shown? (Unless explicitly modified by using SetProperty, default is as defined in the repository.)</td>
</tr>
<tr>
<td>ReadOnly</td>
<td>TRUE or FALSE</td>
<td>Determines whether the control is read-only. Defaults to repository definition unless explicitly modified by using SetProperty.</td>
</tr>
<tr>
<td>Visible</td>
<td>TRUE or FALSE</td>
<td>Determines whether the control is visible. Defaults to repository definition unless explicitly modified by using SetProperty.</td>
</tr>
<tr>
<td>Width</td>
<td>string</td>
<td>Determines Width for a control, in pixels; for example, “80”.</td>
</tr>
</tbody>
</table>

**Used With**

Browser Script

**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", "25pt");
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

```plaintext
controlVar.SetValue (controlValue)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlValue</td>
<td>String containing the value to which to set the control</td>
</tr>
</tbody>
</table>

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) {
    // Example of changing the value of the Abstract control to uppercase
    if(name == "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
    if(name == "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
if(name == "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");
}

Property Set Methods

In the method descriptions, the placeholder oPropSet refers to a variable containing a property set:

- "AddChild Method" on page 296
- "Copy Method" on page 297
- "GetChild Method" on page 298
- "GetChildCount Method" on page 299
- "GetFirstProperty Method" on page 300
- "GetNextProperty Method" on page 301
- "GetProperty Method" on page 302
- "GetPropertyCount Method" on page 303
- "GetType Method" on page 303
- "GetValue Method" on page 304
- "InsertChildAt Method" on page 305
- "PropertyExists Method" on page 305
- "RemoveChild Method" on page 306
- "RemoveProperty Method" on page 307
- "Reset Method" on page 307
- "SetProperty Method" on page 308
- "SetType Method" on page 309
- "SetValue Method" on page 310
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

\[ o\text{PropSet}.\text{AddChild}(child\text{PropSet} \text{as PropertySet}) \]

<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 ( o\text{PropSet} )</td>
</tr>
</tbody>
</table>

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.

```javascript
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

End Sub

GetChild Method

Syntax
GetChild returns a specified child property set of a property set.

oPropSet.GetChild(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 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 every property set after it, have their indexes increased by 1. Similarly, a property set removed using RemoveChild decreases the indexes of following child property sets by 1.

**NOTE:** This method returns the number of direct descendants only. That is, if the child property sets have children of their own, these grandchildren are not included in the computation of the return value.

Used With

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');
    }
}
```

See Also
"AddChild Method" on page 296
"InsertChildAt Method" on page 305

**GetChildCount Method**
This method returns the number of child property sets attached to a parent property set.
Property Set Methods

Syntax

\( o\text{PropSet}.\text{GetChildCount}() \)

<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 \( o\text{PropSet} \)

Usage

This method returns the actual number of child property sets of \( o\text{PropSet} \). Because index numbers for child property sets start at 0, a child count of 3 indicates that there are child property sets at indexes 0, 1, and 2.

Used With


Example

For an example, read “GetChild Method” on page 298.

GetFirstProperty Method

This method returns the name of the first property in a property set.

Syntax

\( o\text{PropSet}.\text{GetFirstProperty}() \)

<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 null
    // or a null string
    while ((propName != "") && (propName != null)) {
        propVal = Inputs.GetProperty(propName);
        if ( Outputs.PropertyExists(propName) ) {
            Outputs.SetProperty(propName, propVal);
        }
        propName = Inputs.GetNextProperty();
    }
    return (CancelOperation);
}
```

See Also
“GetNextProperty Method”
“GetProperty Method” on page 302

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></td>
<td>Not applicable</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 a null 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 300.

**See Also**
“GetFirstProperty Method” on page 300
“GetProperty Method”

---

**GetProperty Method**
This method returns the value of a property when given the property name.

**Syntax**

```
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 GetFirstProperty or GetNextProperty</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 284.

```javascript
var Inputs = TheApplication().NewPropertySet();
var sShipper = Inputs.GetProperty("Shipping Company");
var dWeight = Val(Inputs.GetProperty("Weight"));
var dSize = Val(Inputs.GetProperty("Total Dimensions"));
var iZone = Val(Inputs.GetProperty("Zone"));
```

The following example is in C++.
```cpp
char typeBuffer[40];
strcpy(typeBuffer, inputPS.type);
```

See Also
"GetFirstProperty Method" on page 300
"GetNextProperty Method" on page 301
"SetProperty Method" on page 308

GetPropertyCount Method
This method returns the number of properties attached to a property set.

Syntax
`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 contained within a property set

Used With

GetType Method
This method retrieves the data value stored in the type attribute of a property set.
Syntax

\texttt{oPropSet.GetType}

<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 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**


**See Also**

“GetValue Method”

“SetType Method” on page 309

---

**Get Value Method**

This method retrieves the data value stored in the value attribute of a property set.

Syntax

\texttt{oPropSet.GetValue}

<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 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**

See Also
"GetProperty Method" on page 302
"GetType Method" on page 303
"SetValue Method" on page 310

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

See Also
"AddChild Method" on page 296

PropertyExists Method
This method returns a Boolean value indicating whether a specified property exists in a property set.
**Syntax**

\texttt{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 \texttt{GetProperty} returns a null string ("") for every nonexistent property, use \texttt{PropertyExists()} in an if statement to determine whether a specific property has been set.

**Used With**


**Example**

For an example, read “\texttt{GetFirstProperty Method}” on page 300.

---

**RemoveChild Method**

This method removes a child property set from a parent property set.

**Syntax**

\texttt{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
```

**See Also**
“AddChild Method” on page 296
“InsertChildAt Method” on page 305

**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 every properties and child property set from a property set.
Syntax

```csharp
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 every property and children from a property set, allowing the property set to be reused with new properties.

Used With


SetProperty Method

This method assigns a data value to a property in a property set.

Syntax

```csharp
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 284.
Dim Svc As Service
Dim Inputs As PropertySet, Outputs As PropertySet
Set Svc = theApplication.GetService("Shipping Engine")
Set Inputs = theApplication.NewPropertySet()

With Inputs
    .SetProperty "Shipping Company", "Airline"
    .SetProperty "Weight", "12"
    .SetProperty "Total Dimensions", "48"
    .SetProperty "Shipping Method", "Second-Day Air"
End With

See Also
"GetProperty Method” on page 302

SetType Method
This method assigns a data value to the type attribute of a property set.

Syntax

```
PropSet.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

See Also
"GetType Method” on page 303
"SetValue Method” on page 310
SetValue Method
This method assigns a data value to the value attribute of a property set.

Syntax
\texttt{oPropSet.SetValue value}

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{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

See Also
"GetValue Method" on page 304
"SetProperty Method" on page 308
"SetValue Method"

Miscellaneous Methods
The following methods do not belong to any other category:

- "GetErrorCode Method" on page 310
- "GetErrorMessage Method" on page 312
- "TheApplication Method" on page 312

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></td>
<td>Not applicable</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);
}
...
GetErrorMessage Method

This method is used with the Java Data Bean to display error messages.

Syntax

```java
public string getErrorMessage()
```

<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 an error message

Used With

Java Data Bean

See Also

“GetErrorCode Method”

TheApplication Method

TheApplication is a global method that returns the unique object of type Application. This is the root of objects within the Siebel Applications object hierarchy. Use this method to determine the object reference of the application, which is later used to find other objects or to invoke methods on the application object.

Browser Script Syntax

```javascript
theApplication()
```

VB Syntax

```vb
TheApplication
```

See Also

“GetErrorMessage Method”
**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 convenience, the Siebel applications provide the shortcut constant theApplication.

To determine if you are logged in to a server database or local database, use
TheApplication.invokemethod("GetDataSource").

**Used With**
Browser Script, Server Script

**Example**
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. Use this to build real-time interfaces to Siebel using C++ for integration purposes.

## Building the Siebel COM Client in C++

Use the following procedure to build a Siebel COM client in C++ that uses the Microsoft Foundation Class (MFC) library.

**To build the Siebel COM client in C++**

2. Select the MFC AppWizard (exe) project type.
3. In the Project name field, enter SiebelCOM, and then click OK. The MFC AppWizard starts.
4. 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, as shown in the following illustration.
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 C:\Sea750\client\bin folder. Choose sobjsrv.tlb.

12 In the Confirm Classes dialog box, make sure all five Siebel classes are selected, and then click OK. Click OK again to close the Class Wizard.

13 Add code to communicate with the Siebel COM Server.

   a In the workspace window, click the FileView tab.
b Expand the Source Files and Header Files folders, as shown in the following illustration.

c Double-click the SiebelCOMDlg.h file.

The code window opens, as shown in the following illustration.
Enter the code that is highlighted in boldface in Figure 13 into the SiebelCOMDlg.h file.

```cpp
#include "sobjsrv.h" // include Siebel wrapper classes

class CSiebelCOMDlgAutoProxy;

class CSiebelCOMDlg : public CDialog{
    DECLARE_DYNAMIC(CSiebelCOMDlg);
    friend class CSiebelCOMDlgAutoProxy;
    SiebelApplication sApp;  // declare Siebel object

    // Construction
    public:
        CSiebelCOMDlg(CWnd* pParent = NULL); // standard constructor
        virtual ~CSiebelCOMDlg();
}

Figure 13. Code for SiebelCOMDlg.h
Choose File > Open and select the SiebelCOMDlg.cpp file. Add the code that is highlighted in boldface in Figure 14 to the OnInitDialog procedure.

```cpp
BOOL CSiebelCOMDlg::OnInitDialog()
{
CDialog::OnInitDialog();
// Add "About..." menu item to system menu
// IDM_ABOUTBOX must be in the system command range.
ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
ASSERT(IDM_ABOUTBOX < 0xF000);
CMenu* pSysMenu = GetSystemMenu(FALSE);
if (pSysMenu != NULL)
{
CString strAboutMenu;
strAboutMenu.LoadString(IDS_ABOUTBOX);
if (!strAboutMenu.IsEmpty())
{
    pSysMenu->AppendMenu(MF_SEPARATOR);
    pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
}
}
// Set the icon for this dialog. The framework does this
// automatically when the application's main window
// is not a dialog
setIcon(m_hIcon, TRUE); // Set big icon
setIcon(m_hIcon, FALSE); // Set small icon
// TODO: Add extra initialization here
// Start the Siebel Data Server
if (!sApp.CreateDispatch(_T("SiebelDataServer.ApplicationObject")))
{
    AfxMessageBox("Cannot start Siebel Data Server.");
    EndDialog(-1); // fail
} else
{
    AfxMessageBox("Siebel Data Server initialized.");
}
return TRUE; // return TRUE unless you set the focus to a control
}
```

Figure 14. Code to Be Added to OnInitDialog Routine in SiebelCOMDlg.cpp
In the same file, add the code that is highlighted in boldface in Figure 15 and Figure 16 to the OnOKDialog procedure. Make sure that the line beginning with sApp.LoadObjects points to the location of the CFG file you intend to use. In the line beginning with sApp.Login, make sure that you have entered a valid logon name and password.

```cpp
void CSiebelCOMDlg::OnOK()
{
    short sErr;

    //Load Configuration File
    // Make sure that the following line points to the configuration file you intend to use!
    sApp.LoadObjects("C:\\siebel\\bin\\siebel.cfg", &sErr);
    if(sErr)
    {
        AfxMessageBox("LoadObjects failed.");
        return;
    } else
    {
        AfxMessageBox("CFG file loaded.");
    }

    //Login as Sadmin
    sApp.Login("SADMIN", "SADMIN", &sErr);
    if (sErr)
    {
        AfxMessageBox("Login failed.");
        return;
    } else
    {
        AfxMessageBox("Logged into Siebel database.");
    }

    //Get Account BusObject
    LPDISPATCH lpdBo;
    lpdBo = sApp.GetBusObject("Account", &sErr);
    if (sErr)
    {
        AfxMessageBox("GetBusObject failed.");
        return;
    } else
    {
        AfxMessageBox("Account BusObject retrieved.");
    }
}
```

Figure 15. Code to be Added to OnOKDialog Routine in SiebelCOMDlg.cpp
When you have finished creating your program, test it to make sure it works properly.

**Testing Your Program**

**To test your program**

1. Start your Siebel client application using the same CFG file and login 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 Data Source key indicates the database source you specified at logon in Step 2.

4 In Microsoft Visual C++, choose Build > Build SiebelCOM.exe, or press F7. If there are any errors or warnings reported in the output window, correct the errors and repeat this step.

5 Choose Build > Execute SiebelCOM.exe, or press F5.
   A message box displays the message “Siebel Data Server initialized.”

6 Click OK.
   The customized dialog box opens.

7 The application displays a series of message boxes, with the following messages:
   “CFG file loaded.”
   “Logged into Siebel database.”
   “Account BusObject retrieved.”
   “Account BusComp retrieved.”
   The application displays the name of the first account in the All Accounts view.
This quick reference has the following topics:

- “Application Methods for COM Data Control” on page 324
- “Business Component Methods for COM Data Control” on page 326
- “Business Object Methods for COM Data Control” on page 330
- “Business Service Methods for COM Data Control” on page 330
- “Property Set Methods for COM Data Control” on page 331

### Application Methods for COM Data Control

Table 23 lists a summary of the Application 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><code>Dim application as SiebelDataControl</code> <code>Dim status as Boolean</code> <code>status = application.Attach(sessionID As String)</code></td>
</tr>
<tr>
<td>CurrencyCode Method</td>
<td>Returns the three-letter operating currency code.</td>
<td><code>Dim application as SiebelDataControl</code> <code>Dim sCur as String</code> <code>sCur = Application.CurrencyCode</code></td>
</tr>
<tr>
<td>Detach Method</td>
<td>Returns a string containing the Siebel session ID.</td>
<td><code>Dim application as SiebelDataControl</code> <code>Dim sessionId as String</code> <code>sessionId = application.Detach()</code></td>
</tr>
<tr>
<td>EnableExceptions Method</td>
<td>Enables/disables native COM error handling.</td>
<td><code>Dim application as SiebelDataControl</code> <code>Dim bEnable as Boolean</code> <code>bEnable = application.EnableExceptions(bEnable)</code></td>
</tr>
<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 SiebelDataControl</code> <code>Dim busObject as SiebelBusObject</code> <code>set busObject = application.GetBusObject(busobjName as String)</code></td>
</tr>
</tbody>
</table>
### Table 23. Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetLastErrCode       | Returns the last error code.             | Dim application as SiebelDataControl  
| Method               |                                          | iErr = application.GetLastErrCode                                       |
| GetLastErrText       | Returns the last error text message.     | Dim application as SiebelDataControl  
| Method               |                                          | sText = application.GetLastErrText                                       |
| GetProfileAttr       | Returns the value of an attribute in a user profile. | Dim application as SiebelDataControl  
| Method               |                                          | sText = application.GetProfileAttr(profileAttributeName as string)      |
| GetService           | Instantiates and returns a new instance of the argument-specified service. | Dim application as SiebelDataControl  
| Method               |                                          | set service = application.GetService(serviceName as String)              |
| GetSharedGlobal      | Returns the shared user-defined global variables. | Dim application as SiebelDataControl  
| Method               |                                          | sText = application.GetSharedGlobal(globalVariableName as string)       |
| InvokeMethod         | Calls the named specialized method.      | Dim application as SiebelDataControl  
| Method               |                                          | sReturn = application.InvokeMethod(methodName as String, methodArgs as String or StringArray) |
| Login                | Allows external applications to log in to the COM Data Server. | Dim application as SiebelDataControl  
| Method               |                                          | sErr = application.Login(connectString as String, userName as String, password as String) |
| LoginId              | Returns the login ID of the user who started the Siebel application. | Dim application as SiebelDataControl  
| Method               |                                          | sID = application.LoginId                                                |
| LoginName            | Returns the login name of the user who started the Siebel application. | Dim application as SiebelDataControl  
| Method               |                                          | sUser = application.LoginName                                             |
| Logoff               | Disconnects the client from the server.  | Dim SiebApp as SiebelDataControl  
| Method               |                                          | boolVal = siebApp.LogOff()                                               |
| NewPropertySet       | Constructs and returns a new property set object. | Dim application as SiebelDataControl  
| Method               |                                          | oPropSet = oApplication.NewPropertySet()                                 |
Table 23. Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| 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  
|                       | Returns a Boolean value indicating whether or not method succeeded.         | 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 application as SiebelDataControl  
|                       | GetSharedGlobal.                                                             | Dim SiebApp as SiebelDataControl  
|                       |                                                                             | boolVal=SetSharedGlobal(varName As String, value As String)              |
| Trace Method          | Appends a message to the trace file.                                        | Dim application as SiebelDataControl  
|                       |                                                                             | Dim SiebApp as SiebelDataControl  
|                       |                                                                             | boolVal=siebApp.TraceOn(msg As String) As Boolean                      |
| TraceOff Method       | Turns off the tracing started by the TraceOn method.                         | Dim application as SiebelDataControl  
|                       |                                                                             | Dim SiebApp as SiebelDataControl  
|                       |                                                                             | boolVal=siebApp.TraceOff as Boolean                                    |
| TraceOn Method        | Turns on the tracking of allocations and deallocations of Siebel objects,    | Dim application as SiebelDataControl  
|                       | and SQL statements generated by the Siebel application.                     | Dim SiebApp as SiebelDataControl  
|                       |                                                                             | boolVal=siebApp.TraceOn(fileName As String, category As String, src As String) As Boolean |
Business Component Methods for COM Data Control

Table 24 lists a summary of the Business Component methods’ syntax.

Table 24. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| ActivateField Method | Allows queries to retrieve data for the specified field.                    | Dim busComp as SiebelBusComp
|                      |                                                                             | BusComp.ActivateField(fieldName as String)                             |
| ActivateMultipleFields Method | Allows queries to retrieve data for the fields specified in the property set. | Dim busComp as SiebelBusComp
|                      |                                                                             | busComp.ActivateMultipleFields(oPropSet as SiebelPropertySet)          |
| Associate Method     | Creates a new many-to-many relationship for the parent object through an association business component. | Dim busComp as SiebelBusComp
|                      |                                                                             | busComp.Associate(whereIndicator as Integer)                          |
| BusObject Method     | Returns the business object that contains the business component.           | Dim busComp as SiebelBusComp
|                      |                                                                             | Dim busObject as SiebelBusObject
|                      |                                                                             | Set busObject = busComp.BusObject                                      |
| ClearToQuery Method  | Clears the current query and sort specifications on the business component. | Dim busComp as SiebelBusComp
|                      |                                                                             | busComp.ClearToQuery                                                   |
| DeactivateFields Method | Deactivates every currently activated field.                                | Dim busComp as SiebelBusComp
|                      |                                                                             | busComp.DeactivateFields                                               |
| DeleteRecord Method  | Removes the current record from the business component.                    | Dim busComp as SiebelBusComp
|                      |                                                                             | busComp.DeleteRecord                                                   |
| ExecuteQuery Method  | Retrieves a set of BusComp records.                                         | Dim buscomp as SiebelBusComp
|                      |                                                                             | buscomp.ExecuteQuery(cursorMode As Integer) As Boolean                 |
| ExecuteQuery2 Method | Retrieves a set of BusComp records.                                         | Dim buscomp as SiebelBusComp
|                      |                                                                             | buscomp.ExecuteQuery2(cursorMode As Integer, ignoreMaxCursorSize As Integer) As Boolean |
| FirstRecord Method   | Moves to the first record in the business component.                        | Dim busComp as SiebelBusComp
|                      |                                                                             | Dim bIsRecord as Boolean
|                      |                                                                             | bIsRecord = busComp.FirstRecord                                         |
| 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)                     |
### Table 24. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetFormattedFieldValue     | Returns a formatted value for the field specified in the argument. | Dim busComp as SiebelBusComp  
                           | Dim sValue as String  
                           | sValue = busComp.GetFormattedFieldValue(FieldName as String) |
| GetLastErrCode            | Returns the most recent error code.       | Dim errCode As Integer  
                           | Dim SiebApp as SiebelDataControl  
                           | errCode = siebApp.GetLastErrCode |
| GetLastErrText            | Returns the most recent error text message. | Dim busComp as SiebelBusComp  
                           | Dim sErr as String  
                           | busComp.GetLastErrText |
| GetMultipleFieldValues    | Returns a value for the fields specified in the property set. | Dim busComp as SiebelBusComp  
                           | Dim busComp.GetMultipleFieldValues(FieldName as String,  
                           | FieldValues as SiebelPropertySet)  
                           | set mVGBusComp = busComp.GetMVGBusComp(FieldName as String) |
| GetMVGBusComp             | Returns the MVG business component associated with the field specified in the argument. | Dim busComp as SiebelBusComp  
                           | Dim mVGBusComp as SiebelBusComp  
                           | set mVGBusComp = busComp.GetMVGBusComp(FieldName as String) |
| GetNamedSearch            | Returns the argument-named search specification. | Dim busComp as SiebelBusComp  
                           | Dim sValue as String  
                           | sValue = busComp.GetNamedSearch(SearchName as String) |
| GetPicklistBusComp        | Returns the pick business component associated with the field specified in the argument. | Dim busComp as SiebelBusComp  
                           | Dim pickBusComp as SiebelBusComp  
                           | set pickBusComp = busComp.GetPicklistBusComp(FieldName as String) |
| GetSearchExpr             | Returns the current search expression.    | Dim busComp as SiebelBusComp  
                           | Dim sExpr as String  
                           | sExpr = busComp.GetSearchExpr |
| GetSearchSpec             | Returns the current search specification for the field specified in the argument. | Dim busComp as SiebelBusComp  
                           | Dim sSpec as String  
                           | sSpec = busComp.GetSearchSpec(FieldName as String) |
| GetUserProperty           | Returns the value of a named user property. | Dim buscomp as SiebelBusComp  
                           | Dim retStr as String  
                           | retStr = buscomp.GetUserProp(prop As String) As String |
### Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| 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
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 |
| 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) |
### Table 24. Business Component 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>Dim busComp as SiebelBusComp&lt;br&gt;busComp.SetFormattedFieldValue(FieldName as String, FieldValue as String)</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>Dim busComp as SiebelBusComp&lt;br&gt;busComp.SetMultipleFieldValues(propSet as SiebelPropertySet)</td>
</tr>
<tr>
<td><strong>SetNameSearch Method</strong></td>
<td>Sets a named search specification on the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.SetNameSearch(searchName as String, searchSpec as String)</td>
</tr>
<tr>
<td><strong>SetSearchExpr Method</strong></td>
<td>Sets the search specification for the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.SetSearchExpr(searchSpec as String)</td>
</tr>
<tr>
<td><strong>SetSearchSpec Method</strong></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><strong>SetSortSpec Method</strong></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><strong>SetViewMode Method</strong></td>
<td>Sets the visibility type for the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim boolVal as Boolean&lt;br&gt;boolVal=buscomp.SetViewMode(mode As Integer) As Boolean</td>
</tr>
<tr>
<td><strong>UndoRecord Method</strong></td>
<td>Reverses any uncommitted changes made to the record.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.UndoRecord</td>
</tr>
<tr>
<td><strong>WriteRecord Method</strong></td>
<td>Commits to the database any changes made to the current record.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.WriteRecord</td>
</tr>
</tbody>
</table>
Business Object Methods for COM Data Control

Table 25 lists a summary of the Business Object methods’ syntax.

Table 25. Business Object Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetBusComp   | Returns the specified business component.        | Dim busObject as SiebelBusObject  
| Method       |                                                  | Dim busComp as SiebelBusComp  
|              |                                                  | set busComp = BusObject.GetBusComp(BusCompName as String)              |
| GetLastErrCode| Returns the most recent error code.              | Dim busObject as SiebelBusObject  
| Method       |                                                  | Dim iErr as Integer  
|              |                                                  | busObject.GetLastError()                                                 |
| GetLastErrText| Returns the most recent error text.              | Dim busObject as SiebelBusObject  
| Method       |                                                  | Dim sErr as String  
|              |                                                  | busObject.GetLastErrorText()                                              |
| Name Method  | Returns the name of the control.                 | Dim busObject as SiebelBusObject  
|              |                                                  | Dim sName as String  
|              |                                                  | sName = busObject.Name                                                    |

Business Service Methods for COM Data Control

Table 26 lists a summary of the Business Service methods’ syntax.

Table 26. Business Service Methods Syntax Summary

<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  
| Method           |                                                      | Dim sName as String  
|                  |                                                      | sName = oService.GetFirstProperty()                                  |
| GetNextProperty  | Once the name of the first property has been retrieved, retrieves the name of the next property of a business service. | Dim oService as SiebelService  
| Method           |                                                      | Dim sName as String  
|                  |                                                      | sName = oService.GetNextProperty()                                  |
| GetProperty      | Retrieves the value stored in the specified property. | Dim oService as SiebelService  
| Method           |                                                      | Dim sValue as String  
|                  |                                                      | sValue = oService.GetProperty(propName as String)                    |
### Property Set Methods for COM Data Control

Table 27 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>Name Method</td>
<td>Returns the name of the business service.</td>
<td>Dim oService as SiebelService&lt;br&gt;Dim sName as String&lt;br&gt;sName = oService.Name</td>
</tr>
<tr>
<td>InvokeMethod</td>
<td>Calls a specialized method or a user-created method on the business service.</td>
<td>Dim oService as SiebelService&lt;br&gt;Dim Return&lt;br&gt;Return = oService.InvokeMethod(methodName as String, InputArguments as SiebelPropertySet, OutputArguments as SiebelPropertySet)</td>
</tr>
<tr>
<td>PropertyExists</td>
<td>Returns a Boolean value indicating whether the property specified in the argument exists.</td>
<td>Dim oService as SiebelService&lt;br&gt;Dim propExists as Boolean&lt;br&gt;propExists = oService.PropertyExists(propName as String)</td>
</tr>
<tr>
<td>RemoveProperty</td>
<td>Removes a property from a business service.</td>
<td>Dim oService as SiebelService&lt;br&gt;oService.RemoveProperty(propName as String)</td>
</tr>
<tr>
<td>SetProperty</td>
<td>Assigns a value to a property of a business service.</td>
<td>Dim oService as SiebelService&lt;br&gt;oService.SetProperty(propName as String, propValue as String)</td>
</tr>
</tbody>
</table>

### Property Set Methods for COM Data Control

Table 26 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>Name Method</td>
<td>Returns the name of the business service.</td>
<td>Dim oService as SiebelService&lt;br&gt;Dim sName as String&lt;br&gt;sName = oService.Name</td>
</tr>
<tr>
<td>InvokeMethod</td>
<td>Calls a specialized method or a user-created method on the business service.</td>
<td>Dim oService as SiebelService&lt;br&gt;Dim Return&lt;br&gt;Return = oService.InvokeMethod(methodName as String, InputArguments as SiebelPropertySet, OutputArguments as SiebelPropertySet)</td>
</tr>
<tr>
<td>PropertyExists</td>
<td>Returns a Boolean value indicating whether the property specified in the argument exists.</td>
<td>Dim oService as SiebelService&lt;br&gt;Dim propExists as Boolean&lt;br&gt;propExists = oService.PropertyExists(propName as String)</td>
</tr>
<tr>
<td>RemoveProperty</td>
<td>Removes a property from a business service.</td>
<td>Dim oService as SiebelService&lt;br&gt;oService.RemoveProperty(propName as String)</td>
</tr>
<tr>
<td>SetProperty</td>
<td>Assigns a value to a property of a business service.</td>
<td>Dim oService as SiebelService&lt;br&gt;oService.SetProperty(propName as String, propValue as String)</td>
</tr>
<tr>
<td>AddChild Method</td>
<td>Adds subsidiary property sets to a property set.</td>
<td>Dim oPropSet as SiebelPropertySet&lt;br&gt;Dim iIndex as Integer&lt;br&gt;iIndex = oPropSet.AddChild(childObject as PropertySet)</td>
</tr>
<tr>
<td>Copy Method</td>
<td>Returns a copy of a property set.</td>
<td>Dim oPropSet1 as SiebelPropertySet&lt;br&gt;Dim oPropSet2 as SiebelPropertySet&lt;br&gt;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&lt;br&gt;Dim sPropVal as String&lt;br&gt;sPropVal = oPropSet.GetChild(index as Integer)</td>
</tr>
</tbody>
</table>
### Method | Description | Syntax
--- | --- | ---
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()

GetFirstChild Method | Returns the name of the first property in a property set. | Dim oPropSet as SiebelPropertySet  
Dim sPropName as String  
sPropName = oPropSet.GetFirstChild()

GetNextChild Method | Returns the name of the next property in a property set. | Dim oPropSet as SiebelPropertySet  
Dim sPropName as String  
sPropName = oPropSet.GetNextChild()

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)

GetPropertyCount Method | Returns the number of properties attached to a property set. | Dim oPropSet as SiebelPropertySet  
Dim count as Long  
count = oPropSet.GetPropertyCount()

GetType Method | Returns the value stored in a type in a property set. | Dim oPropSet as SiebelPropertySet  
Dim sTypeVal as String  
sTypeVal = oPropSet.GetType()

GetValue Method | Returns a value stored as part 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 PropertySet  
dim propExists as Boolean  
propExists = 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()
### Table 27. Property Set Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| 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 quick reference has the following topics:

- "Application Methods for COM Data Server"
- "Business Component Methods for COM Data Server" on page 338
- "Business Object Methods for COM Data Server" on page 342
- "Business Service Methods for COM Data Server" on page 343
- "Property Set Methods for COM Data Server" on page 344

### Application Methods for COM Data Server

Table 28 lists a summary of the Applications methods’ syntax.

<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  
| Method           |                                                                             | Dim sCur as String  
|                  |                                                                             | sCur = Application.CurrencyCode(ErrCode as Integer)                                   |
| GetBusObject     | Instantiates and returns a new instance of the business object specified in the argument. | Dim application as SiebelApplication  
| Method           |                                                                             | Dim busObject as SiebelBusObject  
|                  |                                                                             | set busObject = application.GetBusObject(busobjName as String, ErrCode as Integer)     |
| GetLastErrCode   | Returns the last Siebel error number.                                        | Dim application as SiebelApplication  
| Method           |                                                                             | Dim iErrNum as Integer  
|                  |                                                                             | iErrNum = application.GetLastErrCode(ErrCode as Integer)                              |
| GetLastErrText   | Returns the last error text message.                                         | Dim application as SiebelApplication  
| Method           |                                                                             | Dim sText as String  
|                  |                                                                             | sText = application.GetLastErrText(ErrCode as Integer)                                |
### Table 28. 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) |
| LoadObjects Method   | Starts the COM Data Server object and returns a reference to the Application object. | Dim application as SiebelApplication  
                      Dim returned as SiebelApplication  
                      application.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 PropertySet  
                      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) |
| PositionName Method  | Returns the position name of the user’s current position.                   | Dim application as SiebelApplication  
                      Dim sPosition as String  
                      sPosition = application.PositionName(ErrCode as Integer) |
### Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| SetPositionId        | Sets the active position to the position ID specified in the argument. Returns a Boolean value indicating if the method succeeded. | Dim application as SiebelApplication  
Dim posId as String  
Dim status as Boolean  
status = application.SetPositionId(posId as String, ErrCode as Integer) |
| SetPositionName      | Sets the active position to the position name specified in the argument. Returns a Boolean value indicating if the method succeeded. | Dim application as SiebelApplication  
Dim posName as String  
Dim status as Boolean  
status = application.SetPositionName(posName as String, ErrCode as Integer) |
| SetProfileAttr       | Used in personalization to assign values to attributes in a user profile. | Dim application as SiebelApplication  
application.SetProfileAttr(name as String, value as String, ErrCode as Integer) |
| SetSharedGlobal      | Sets a shared user-defined global variable.                   | Dim application as SiebelApplication  
application.SetSharedGlobal(varName as String, value as String, ErrCode as Integer) |
| Trace 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 29 lists a summary of the Business Component methods’ syntax.

Table 29. 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 busComp as SiebelBusComp&lt;br&gt;busComp.ActivateField(fieldName as String, ErrCode as Integer)</td>
</tr>
<tr>
<td>ActivateMultipleFields Method</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, ErrCode as Integer)</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 busComp as SiebelBusComp&lt;br&gt;busComp.Associate(whereIndicator as Integer, ErrCode as Integer)</td>
</tr>
<tr>
<td>BusObject Method</td>
<td>Returns the business object that contains the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim busObject as BusObject&lt;br&gt;Set busObject = busComp.BusObject(ErrCode as Integer)</td>
</tr>
<tr>
<td>ClearToQuery Method</td>
<td>Clears the current query and sort specifications on the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.ClearToQuery(ErrCode as Integer)</td>
</tr>
<tr>
<td>DeactivateFields Method</td>
<td>Deactivates every currently activated field.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.DeactivateFields(ErrCode as Integer)</td>
</tr>
<tr>
<td>DeleteRecord Method</td>
<td>Removes the current record from the business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.DeleteRecord(ErrCode as Integer)</td>
</tr>
<tr>
<td>ExecuteQuery Method</td>
<td>Retrieves a set of BusComp records.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.ExecuteQuery(cursorMode as Boolean, ErrCode as Integer)</td>
</tr>
<tr>
<td>ExecuteQuery2 Method</td>
<td>Retrieves a set of BusComp records.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;busComp.ExecuteQuery2(cursorMode as Boolean, ignoreMaxCursorSize as Boolean, ErrCode as Integer)</td>
</tr>
</tbody>
</table>
Table 29. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<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 blsRecord as Boolean&lt;br&gt;blsRecord = busComp.FirstRecord(ErrCode as Integer)</td>
</tr>
<tr>
<td><strong>FirstSelected Method</strong></td>
<td>Returns the association business component.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim AssocBusComp as BusComp&lt;br&gt;Set AssocBusComp = busComp.GetAssocBusComp(ErrCode as Integer)</td>
</tr>
<tr>
<td><strong>GetFieldvalue Method</strong></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, ErrCode as Integer)</td>
</tr>
<tr>
<td><strong>GetFormattedFieldValue Method</strong></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, ErrCode as Integer)</td>
</tr>
<tr>
<td><strong>GetMultipleFieldValues Method</strong></td>
<td>Returns a value for the fields specified in the property set.</td>
<td>Dim busComp as SiebelBusComp&lt;br&gt;Dim retValue as Boolean&lt;br&gt;retValue = busComp.GetMultipleFieldValues(oPropSetName as SiebelPropertySet, oPropSetValue as SiebelPropertySet, ErrCode as Integer)</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>Dim busComp as SiebelBusComp&lt;br&gt;Dim mVGBusComp as SiebelBusComp&lt;br&gt;Set mVGBusComp = busComp.GetMVGBusComp(FieldName as String, ErrCode as Integer)</td>
</tr>
<tr>
<td><strong>GetNamedSearch Method</strong></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(SearchName as String, ErrCode as Integer)</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>Dim busComp as SiebelBusComp&lt;br&gt;Dim pickBusComp as SiebelBusComp&lt;br&gt;Set pickBusComp = busComp.GetPicklistBusComp(FieldName as String, ErrCode as Integer)</td>
</tr>
</tbody>
</table>
### Table 29. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| 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) |
| ParentBusComp Method        | Returns the parent business component.                                      | Dim busComp as SiebelBusComp  
  Dim parentBusComp as SiebelBusComp  
  Set parentBusComp =  
  busComp.ParentBusComp(ErrCode as Integer) |
Table 29. 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 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 Dim bReturn as Boolean 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 busComp.RefineQuery(ErrCode as Integer)</td>
</tr>
<tr>
<td>SetFieldValume 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 SetFieldValume(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 busComp.SetFormattedFieldValume(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 busComp.SetMultipleFieldValues(oPropSet 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 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 busComp.SetSearchExpr(searchSpec as String, ErrCode as Integer)</td>
</tr>
<tr>
<td>SetSearchSpec Method</td>
<td>Sets the search specification for the specified field.</td>
<td>Dim busComp as SiebelBusComp busComp.SetSearchSpec(FieldName as String, searchSpec as String, ErrCode as Integer)</td>
</tr>
<tr>
<td>SetSortSpec Method</td>
<td>Sets the sort specification for a query.</td>
<td>Dim busComp as SiebelBusComp busComp.SetSortSpec(sortSpec as String, ErrCode as Integer)</td>
</tr>
</tbody>
</table>
Table 30 lists a summary of the Business Object methods’ syntax.

### Table 30. Business Object Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetUserProperty Method</td>
<td>Sets the value of the specified User Property.</td>
<td><code>Dim busComp as SiebelBusComp</code> <code>busComp.SetUserProperty(propertyName as String, newValue as String, ErrCode as Integer)</code></td>
</tr>
<tr>
<td>SetViewMode Method</td>
<td>Sets the visibility type for the business component.</td>
<td><code>Dim buscomp as SiebelBusComp</code> <code>buscomp.SetViewMode(mode As Integer, errCode As Integer)</code></td>
</tr>
<tr>
<td>UndoRecord Method</td>
<td>Reverses any uncommitted changes made to the record.</td>
<td><code>Dim busComp as SiebelBusComp</code> <code>busComp.UndoRecord(ErrCode as Integer)</code></td>
</tr>
<tr>
<td>WriteRecord Method</td>
<td>Commits to the database any changes made to the current record</td>
<td><code>Dim busComp as SiebelBusComp</code> <code>busComp.WriteRecord(ErrCode as Integer)</code></td>
</tr>
</tbody>
</table>

## Business Object Methods for COM Data Server

Table 30 lists a summary of the Business Object methods’ syntax.
# Business Service Methods for COM Data Server

Table 31 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 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 | Once the name of the first property has been retrieved, retrieves the name of the next property of a business service. | `Dim oService as SiebelService`  
`Dim sName as String`  
`sName = oService.GetNextProperty(ErrCode as Integer)` |
| GetProperty 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`  
`oService.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`  
`oService.RemoveProperty(propName as String, ErrCode as Integer)` |
| SetProperty Method | Assigns a value to a property of a business service.                        | `Dim oService as SiebelService`  
`oService.SetProperty(propName as String, propValue as String, ErrCode as Integer)` |
# Property Set Methods for COM Data Server

Table 32 lists a summary of the Property Set methods’ syntax.

## Table 32. 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 SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim iIndex as Integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>iIndex = oPropSet.AddChild(childObject as Property Set, errCode as Integer)</td>
</tr>
<tr>
<td>Copy Method</td>
<td>Returns a copy of a property set.</td>
<td>Dim oPropSet1 as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim oPropSet2 as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet2 = oPropSet1.Copy(ErrCode as Integer)</td>
</tr>
<tr>
<td>GetChild Method</td>
<td>Returns a specified child property set.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim oChildPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oChildPropSet = oPropSet.GetChild(index as Integer, ErrCode as Integer)</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 SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim iCount as Integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>iCount = oPropSet.GetChildCount(ErrCode as Integer)</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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim sPropName as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sPropName = oPropSet.GetFirstProperty(ErrCode as Integer)</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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim sPropName as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sPropName = oPropSet.GetNextProperty(ErrCode as Integer)</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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim sPropVal as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sPropVal = oPropSet.GetProperty(propName as String, ErrCode as Integer)</td>
</tr>
<tr>
<td>GetPropertyCount Method</td>
<td>Returns the number of properties contained within the property set.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim propCount as Integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>propCount = oPropSet.GetPropertyCount(ErrCode 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>GetType Method</td>
<td>Returns the value stored in a type in a property set.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim sTypeVal as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sTypeVal = oPropSet.GetType(value as String)</td>
</tr>
<tr>
<td>GetValue Method</td>
<td>Returns a value stored as part of a property set.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim sValVal as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sValVal = oPropSet.GetValue(ErrCode as Integer)</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 SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet.InsertChildAt(childObject as String, index as Integer, ErrCode as Integer)</td>
</tr>
<tr>
<td>PropertyExists Method</td>
<td>Returns a Boolean value indicating whether the property specified in the argument exists.</td>
<td>Dim oPropSet as Property Set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim propExists as Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>propExists = oPropSet.PropertyExists(propName as String, ErrCode as Integer)</td>
</tr>
<tr>
<td>RemoveChild Method</td>
<td>Removes a child property set as a specified index from a parent property set.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet.RemoveChild(index as Integer, errCode as Integer)</td>
</tr>
<tr>
<td>RemoveProperty Method</td>
<td>Removes the property specified in its argument from a property set.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet.RemoveProperty(propName as String, ErrCode as Integer)</td>
</tr>
<tr>
<td>Reset Method</td>
<td>Removes every property and child property set from a property set.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet.Reset(ErrCode as Integer)</td>
</tr>
<tr>
<td>SetProperty Method</td>
<td>Assigns a value to the property of a property set specified in its argument.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet SetProperty(propName as String, propValue as String, ErrCode as Integer)</td>
</tr>
<tr>
<td>SetType Method</td>
<td>Assigns a data value to a type member of a property set.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet.SetType(value as String, ErrCode as Integer)</td>
</tr>
<tr>
<td>SetValue Method</td>
<td>Assigns a data value to a value member of a property set.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet.SetValue(value as String, errCode as Integer)</td>
</tr>
</tbody>
</table>
This quick reference has the following topics:

- "Application Methods for Mobile/Dedicated Web Client Automation Server"
- "Property Set Methods for Mobile/Dedicated Web Client Automation Server” on page 356

## Application Methods for Mobile/Dedicated Web Client Automation Server

Table 33 lists a summary of the Application methods’ syntax.

<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 of the active applet.</td>
<td><code>Dim application as SiebelWebApplication</code>&lt;br&gt;<code>Dim busObject as SiebelBusObject</code>&lt;br&gt;<code>Set busObject = application.ActiveBusObject</code></td>
</tr>
<tr>
<td>ActiveViewName Method</td>
<td>Returns the name of the active view.</td>
<td><code>Dim application as SiebelWebApplication</code>&lt;br&gt;<code>Dim sView as String</code>&lt;br&gt;<code>sView = application.ActiveViewName</code></td>
</tr>
<tr>
<td>CurrencyCode Method</td>
<td>Returns the three-letter operating currency code.</td>
<td><code>Dim application as SiebelWebApplication</code>&lt;br&gt;<code>Dim sCur as String</code>&lt;br&gt;<code>sCur = Application.CurrencyCode</code></td>
</tr>
<tr>
<td>EnableExceptions Method</td>
<td>Enables or disables native COM error handling.</td>
<td><code>Dim application as SiebelWebApplication</code>&lt;br&gt;<code>EnableExceptions(bEnable as Boolean)</code>&lt;br&gt;<code>Call application.EnableExceptions(bEnable as Integer)</code></td>
</tr>
<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>
</tbody>
</table>
### Table 33. Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastErrCode Method</td>
<td>Gets the last error code.</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim iErr as Integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>iErr = application.GetLastErrCode</td>
</tr>
<tr>
<td>GetLastErrText Method</td>
<td>Returns the last error text message.</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim sText as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sText = application.GetLastErrText</td>
</tr>
<tr>
<td>GetProfileAttr Method</td>
<td>Returns the value of an attribute in a user profile.</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim profValue as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>profValue = application.GetProfileAttr(profName as String)</td>
</tr>
<tr>
<td>GetService Method</td>
<td>Instantiates and returns a new instance of the argument-specified service.</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim oService as SiebelService</td>
</tr>
<tr>
<td></td>
<td></td>
<td>set oService = Application.GetService(serviceName as String)</td>
</tr>
<tr>
<td>GetSharedGlobal Method</td>
<td>Returns the shared user-defined global variables.</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim sName as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sName = application.GetSharedGlobal(sName as String)</td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>Calls the named specialized method.</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim sReturn as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sReturn = application.InvokeMethod(methodName as String, methodArgs as String or StringArray)</td>
</tr>
<tr>
<td>LoginId Method</td>
<td>Returns the login ID of the user who started the Siebel application.</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim sID as string</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sID = application.LoginId</td>
</tr>
<tr>
<td>LoginName Method</td>
<td>Returns the login name of the user who started the Siebel application.</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim sUser as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sUser = application.LoginName</td>
</tr>
<tr>
<td>Logoff Method</td>
<td>Terminates the Mobile Web Client session.</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim status as Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>status = application.Logoff</td>
</tr>
<tr>
<td>NewPropertySet Method</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 current position.</td>
<td>Dim application as SiebelWebApplication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim sRow as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sRow = application.PositionId</td>
</tr>
</tbody>
</table>
### Table 33. 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 SiebelWebApplication  
                        |                                                                     |                        |  
                        |                                                                 | Dim sPosition as String  
                        |                                                                 | sPosition = application.PositionName |
| **SetPositionId Method** | Sets the active position to the Position ID specified in the argument.       | Dim application as SiebelWebApplication  
                        |                                                                     |                        |  
                        |                                                                 | Dim posId as String  
                        |                                                                 | Dim status as Boolean  
                        |                                                                 | status = application.SetPositionId(posId) |
| **SetPositionName Method** | Sets the active position to the position name specified in the argument. | Dim application as SiebelWebApplication  
                        |                                                                     |                        |  
                        |                                                                 | Dim posName as String  
                        |                                                                 | Dim status as Boolean  
                        |                                                                 | status = application.SetPositionName(posName) |
| **SetProfileAttr Method** | Used in personalization to assign values to attributes in a user profile. | Dim oApplication as SiebelWebApplication  
                        |                                                                     |                        |  
                        |                                                                 | Dim bool as Boolean  
                        |                                                                 | bool = oApplication.SetProfileAttr(name as String, value as String) |
| **SetSharedGlobal Method** | Sets a shared user-defined global variable.                               | Dim application as SiebelWebApplication  
                        |                                                                     |                        |  
                        |                                                                 | Dim bool as Boolean  
                        |                                                                 | bool = application.SetSharedGlobal(varName as String, value as String) |
| **Trace Method**       | Appends a message to the trace file.                                      | Dim application as SiebelWebApplication  
                        |                                                                     |                        |  
                        |                                                                 | application.Trace(message as String) |
| **TraceOff Method**    | Turns off the tracing started by TraceOn.                                  | Dim application as SiebelWebApplication  
                        |                                                                     |                        |  
                        |                                                                 | Dim bool as Boolean  
                        |                                                                 | bool = application.TraceOff |
| **TraceOn Method**     | Turns tracing on.                                                          | Dim application as SiebelWebApplication  
                        |                                                                     |                        |  
                        |                                                                 | Dim bool as Boolean  
                        |                                                                 | bool = application.TraceOn(filename as String, type as String, Selection as String) |
## Business Component Methods for Mobile/Dedicated Web Client Automation Server

Table 34 lists a summary of the Business Component methods’ syntax.

<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;Dim bool as Boolean&lt;br&gt;bool = 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;Dim bool as Boolean&lt;br&gt;bool = 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>
</tbody>
</table>
Table 34. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| ExecuteQuery2 Method   | Retrieves a set of BusComp records.                                          | Dim busComp as SiebelBusComp  
                          |                                                                  | Dim bool as Boolean  
                          |                                                                  | bool = busComp.ExecuteQuery2(cursorMode as Integer, ignoreMaxCursorSize as Boolean) |
| FirstRecord Method     | Moves to the first record in the business component.                        | Dim busComp as SiebelBusComp  
                          |                                                                  | Dim bIsRecord as Boolean  
                          |                                                                  | bIsRecord = busComp.FirstRecord |
| GetAssocBusComp Method | Returns the association business component.                                 | Dim busComp as SiebelBusComp  
                          |                                                                  | Dim AssocBusComp as SiebelBusComp  
                          |                                                                  | Set AssocBusComp = busComp.GetAssocBusComp |
| 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 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) |
| GetLastErrCode Method  | Returns the last Siebel error number.                                       | Dim busComp as SiebelBusComp  
                          |                                                                  | Dim iErr as Integer  
                          |                                                                  | iErr = busComp.GetLastErrCode |
| GetLastErrText Method  | Returns the last error text message.                                        | Dim busComp as SiebelBusComp  
                          |                                                                  | Dim sErr as String  
                          |                                                                  | sErr = busComp.GetLastErrText |
| GetMultipleFieldValues Method | Returns a value for the fields specified in the property set.  | Dim busComp as SiebelBusComp  
                          |                                                                  | busComp.GetMultipleFieldValues(oPropSet as SiebelPropertySet, PValues 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) |
### Table 34. 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>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pick Method</strong></td>
<td>Places the currently selected record in a picklist business component into the appropriate fields of the parent business component.</td>
<td><code>Dim busComp as SiebelBusComp</code> <code>busComp.Pick</code></td>
</tr>
<tr>
<td><strong>PreviousRecord Method</strong></td>
<td>Moves to the previous record in the business component.</td>
<td><code>Dim busComp as SiebelBusComp</code> <code>Dim bReturn as Boolean</code> <code>bReturn = busComp.PreviousRecord</code></td>
</tr>
<tr>
<td><strong>RefineQuery Method</strong></td>
<td>Refines a query after a query has been executed.</td>
<td><code>Dim busComp as SiebelBusComp</code> <code>busComp.RefineQuery</code></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><code>Dim busComp as SiebelBusComp</code> <code>busComp.SetFieldValue(FieldName as String, FieldValue as String)</code></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><code>Dim busComp as SiebelBusComp</code> <code>busComp.SetFormattedFieldValue(FieldName as String, FieldValue as String)</code></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><code>Dim busComp as SiebelBusComp</code> <code>busComp.SetMultipleFieldValues(propSet as SiebelPropertySet)</code></td>
</tr>
<tr>
<td><strong>SetNamedSearch Method</strong></td>
<td>Sets a named search specification on the business component.</td>
<td><code>Dim busComp as SiebelBusComp</code> <code>busComp.SetNamedSearch(searchName as String, searchSpec as String)</code></td>
</tr>
<tr>
<td><strong>SetSearchExpr Method</strong></td>
<td>Sets the search expression for the business component.</td>
<td><code>Dim busComp as SiebelBusComp</code> <code>busComp.SetSearchExpr(searchSpec as String)</code></td>
</tr>
<tr>
<td><strong>SetSearchSpec Method</strong></td>
<td>Sets the search specification for the specified field.</td>
<td><code>Dim busComp as SiebelBusComp</code> <code>busComp.SetSearchSpec(FieldName as String, searchSpec as String)</code></td>
</tr>
<tr>
<td><strong>SetSortSpec Method</strong></td>
<td>Sets the sort specification for a query.</td>
<td><code>Dim busComp as SiebelBusComp</code> <code>busComp.SetSortSpec(sortSpec as String)</code></td>
</tr>
<tr>
<td><strong>SetUserProperty Method</strong></td>
<td>Sets the value of the specified User Property.</td>
<td><code>Dim busComp as SiebelBusComp</code> <code>busComp.SetUserProperty(propertyName as String, newValue as String)</code></td>
</tr>
</tbody>
</table>
Table 34. Business Component 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
busComp.UndoRecord`                                                            |
| WriteRecord Method      | Commits to the database any changes made to the current record. | `Dim busComp as SiebelBusComp
busComp.WriteRecord`                                                            |

Table 35 lists a summary of the Business Object methods’ syntax.

Table 35. 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)`                 |
| GetLastErrCode Method   | Returns the last Siebel error number.    | `Dim busobject as SiebelBusObject
Dim iErr as Integer
iErr = busobject.GetLastErrCode`                                           |
| GetLastErrText Method   | Returns the last error text message.     | `Dim busobject as SiebelBusObject
Dim sValue as String
sValue= busobject.GetLastErrText`                                           |
| Name Method             | Returns the name of the business object.  | `Dim busObject as SiebelBusObject
Dim sName as String
sName = busObject.Name`                                                    |
## Business Service Methods for Mobile/Dedicated Web Client Automation Server

Table 36 lists a summary of the Business Service methods’ syntax.

**Table 36. 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` |
| GetNextProperty       | Method once the name of the first property has been retrieved, retrieves the | `Dim oService as SiebelService  
Dim sName as String  
sName = oService.GetNextProperty` |
| GetProperty           | Method retrieves the value stored in the specified property.                | `Dim oService as SiebelService  
Dim sValue as String  
sValue = oService.GetProperty(propName as String)` |
| InvokeMethod          | Method calls a specialized method or a user-created method on the business  | `Dim oService as SiebelService  
oService.InvokeMethod(methodName as String, InputArguments as SiebelPropertySet, OutputArguments as SiebelPropertySet)` |
| Name Method           | Method returns the name of the business service.                           | `Dim oService as SiebelService  
Dim sName as String  
sName = oService.Name` |
| PropertyExists        | Method returns a Boolean value indicating whether the property specified in  | `Dim oService as SiebelService  
Dim bool as Boolean  
bool = oService.PropertyExists(propName as String)` |
| RemoveProperty        | Method removes a property from a business service.                         | `Dim oService as SiebelService  
Dim bool as Boolean  
bool = 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 Mobile/Dedicated Web Client Automation Server**

Table 37 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  
|                   |                                                                                                                                             | 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 childPropSet as SiebelPropertySet  
|                   |                                                                                                                                             | set childPropSet =  
|                   |                                                                                                                                             | 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 Siebel error number.                                                                                                         | Dim oPropSet as SiebelPropertySet  
|                   |                                                                                                                                             | Dim iErr as Integer  
|                   |                                                                                                                                             | iErr = oPropSet.GetLastErrCode                                                                   |
| GetLastErrText Method | Returns the last error text message.                                                                                                           | Dim oPropSet as SiebelPropertySet  
|                   |                                                                                                                                             | Dim sValue as String  
|                   |                                                                                                                                             | sValue = 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 sPropVal as String  
|                   |                                                                                                                                             | sPropVal = oPropSet.GetProperty(propName as String)                                               |
| GetPropertyCount Method | Returns the number of properties contained within the property set.                                                                           | Dim oPropSet as SiebelPropertySet  
|                   |                                                                                                                                             | Dim lCount as Long  
<p>|                   |                                                                                                                                             | lCount = oPropSet.GetPropertyCount                                                               |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetType Method</td>
<td>Retrieves the data value stored in the type attribute of a property set.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim sTypeVal as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sTypeVal = oPropSet.GetType</td>
</tr>
<tr>
<td>GetValue Method</td>
<td>Retrieves the data value stored in the value attribute of a property set.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim sValVal as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sValVal = oPropSet.GetValue</td>
</tr>
<tr>
<td>InsertChildAt</td>
<td>Inserts a child property set into a parent property set at a specific location.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet.InsertChildAt(childObject as SiebelPropertySet, index as Long)</td>
</tr>
<tr>
<td>PropertyExists</td>
<td>Returns a Boolean value indicating whether the property specified in the argument exists.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td>Method</td>
<td></td>
<td>Dim bool as Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bool = oPropSet.PropertyExists(propName as String)</td>
</tr>
<tr>
<td>RemoveChild Method</td>
<td>Removes a child property set as a specified index from a parent property set.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet.RemoveChild(index as Long)</td>
</tr>
<tr>
<td>RemoveProperty</td>
<td>Removes the property specified in its argument from a property set.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td>Method</td>
<td></td>
<td>oPropSet.RemoveProperty(propName as String)</td>
</tr>
<tr>
<td>Reset Method</td>
<td>Removes every property and child property set from a property set.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet.Reset</td>
</tr>
<tr>
<td>SetProperty Method</td>
<td>Assigns a value to the property of a property set specified in its argument.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet.SetProperty(propName as String, propValue as String)</td>
</tr>
<tr>
<td>setType Method</td>
<td>Assigns a data value to a type member of a property set.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet.SetType(value as String)</td>
</tr>
<tr>
<td>setValue Method</td>
<td>Assigns a data value to a value member of a property set.</td>
<td>Dim oPropSet as SiebelPropertySet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet.SetValue(value as String)</td>
</tr>
</tbody>
</table>
This quick reference has the following topics:

- “SiebelHTMLApplication Methods for Siebel Web Client Automation Server”
- “SiebelService Methods for Siebel Web Client Automation Server” on page 360
- “PropertySet Methods for Siebel Web Client Automation Server” on page 360

### SiebelHTMLApplication Methods for Siebel Web Client Automation Server

Table 38 lists a summary of the Siebel HTMLApplication methods’ syntax.

<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)` |
| Name Method          | Returns the name of the current application as defined in the repository.   | `Dim siebelApp As SiebelHTMLApplication
Dim name as String
name = siebelApp.Name` |
| NewPropertySet Method | Constructs and returns a new property set object.                           | `Dim siebelApp As SiebelHTMLApplication
Dim propSet as SiebelPropertySet
Set propSet = siebelAppNewPropSet` |
**SiebelService Methods for Siebel Web Client Automation Server**

Table 39 lists a summary of the SiebelService methods’ syntax.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| `GetLastErrCode`  | Returns the last error code.                                                | `Dim svc As SiebelService  
Dim iErr as Long  
iErr = svc.GetLastErrCode`                                           |
| `GetLastErrText`  | Returns the last error text message.                                        | `Dim svc As SiebelService  
Dim sText as String  
sText = svc.GetLastErrText`                                              |
| `InvokeMethod`    | Calls a specialized method or a user-created method on the business service.| `Dim svc As SiebelService  
svc.InvokeMethod(MethodName as String, inputPropSet as SiebelPropertySet, outputPropSet as SiebelPropertySet)` |
| `Name`            | Returns the name of the business service.                                   | `Dim svc As SiebelService  
Dim name as String  
name = svc.Name`                                                         |

**PropertySet Methods for Siebel Web Client Automation Server**

Table 40 lists a summary of the PropertySet methods’ syntax.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| `AddChild`   | Adds subsidiary property sets to a property set.                            | `Dim oPropSet as SiebelPropertySet  
oPropSet.AddChild(childObject as SiebelPropertySet)` |
| `Copy`       | Returns a copy of a property set.                                           | `Dim oPropSet1 as SiebelPropertySet  
Dim oPropSet2 as SiebelPropertySet  
Set oPropSet2 = oPropSet1.Copy`                                           |
| `GetChild`   | Returns a specified child property set of a property set.                   | `Dim oPropSet as SiebelPropertySet  
Dim oChildPropSet as SiebelPropertySet  
Set oChildPropSet = oPropSet.GetChild(index as Long)`               |
### PropertySet Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| 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.GetLastErrCode |
| 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 |
| GetType Method       | Returns the value stored in a type in a property set.                       | Dim oPropSet as SiebelPropertySet  
Dim sType as String  
sType = 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) |
### Table 40. PropertySet Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoveProperty Method</td>
<td>Removes the property specified in its argument from a property set.</td>
<td>Dim oPropSet as SiebelPropertySet oPropSet.RemoveProperty(propName as String)</td>
</tr>
<tr>
<td>Reset Method</td>
<td>Removes every property and child property set from a property set.</td>
<td>Dim oPropSet as SiebelPropertySet oPropSet.Reset</td>
</tr>
<tr>
<td>SetProperty Method</td>
<td>Assigns a value to the property of a property set specified in its argument.</td>
<td>Dim oPropSet as SiebelPropertySet oPropSet.SetProperty(propName as String, propValue as String)</td>
</tr>
<tr>
<td>SetType Method</td>
<td>Assigns a data value to a type member of a property set.</td>
<td>Dim oPropSet as SiebelPropertySet oPropSet.SetType(value as String)</td>
</tr>
<tr>
<td>SetValue Method</td>
<td>Assigns a data value to a value member of a property set.</td>
<td>Dim oPropSet as SiebelPropertySet oPropSet.SetValue(value as String)</td>
</tr>
</tbody>
</table>
This quick reference has the following topics:

- “Data Bean Methods for Java Data Bean”
- “Business Component Methods for Java Data Bean” on page 365
- “Business Object Methods for Java Data Bean” on page 368
- “Business Service Methods for Java Data Bean” on page 369
- “Property Set Methods for Java Data Bean” on page 370
- “SiebelException Methods for Java Data Bean” on page 371

### Data Bean Methods for Java Data Bean

Table 41 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><code>boolean attach(String sessionID) throws SiebelException</code></td>
</tr>
<tr>
<td>CurrencyCode Method</td>
<td>Returns the three-letter operating currency code.</td>
<td><code>String currencyCode()</code></td>
</tr>
<tr>
<td>Detach Method</td>
<td>Returns a string containing the Siebel session ID.</td>
<td><code>String detach() throws SiebelException</code></td>
</tr>
<tr>
<td>GetBusObject Method</td>
<td>Instantiates and returns a new instance of the business object specified in the argument.</td>
<td><code>SiebelBusObject getBusObject(String boName) throws SiebelException</code></td>
</tr>
<tr>
<td>GetProfileAttr Method</td>
<td>Returns the value of an attribute in a user profile.</td>
<td><code>String getProfileAttr(String attrName) throws SiebelException</code></td>
</tr>
<tr>
<td>GetService Method</td>
<td>Returns a specified service. If the service is not already running, it is constructed.</td>
<td><code>SiebelService getService(String serviceName) throws SiebelException</code></td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>Calls the named specialized method.</td>
<td><code>String invokeMethod(String name, String[] args) throws SiebelException</code></td>
</tr>
</tbody>
</table>
Table 41. SiebelDataBean Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login Method</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>LoginId Method</td>
<td>Returns the login ID of the user who started the Siebel application.</td>
<td>String loginId()</td>
</tr>
<tr>
<td>LoginName Method</td>
<td>Returns the login name of the user who started the Siebel application.</td>
<td>String loginName()</td>
</tr>
<tr>
<td>Logoff Method</td>
<td>Disconnects the client from the server.</td>
<td>boolean logoff() throws SiebelException</td>
</tr>
<tr>
<td>NewPropertySet Method</td>
<td>Constructs and returns a new property set object.</td>
<td>SiebelPropertySet newPropertySet()</td>
</tr>
<tr>
<td>PositionId Method</td>
<td>Returns the position ID that describes the user’s current position.</td>
<td>String positionId()</td>
</tr>
<tr>
<td>PositionName Method</td>
<td>Returns the position name of the user’s current position.</td>
<td>String positionName()</td>
</tr>
<tr>
<td>setPositionId Method</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>setPositionName Method</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>setProfileAttr Method</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>Trace Method</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>
Business Component Methods for Java Data Bean

Table 42 lists a summary of the Siebel BusComp methods’ syntax.

Table 42. 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 isInserBefore) 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>
<tr>
<td>DeactivateFields Method</td>
<td>Deactivates every currently activated field.</td>
<td>boolean deactivateFields()</td>
</tr>
</tbody>
</table>
Table 42. SiebelBusComp Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeleteRecord Method</td>
<td>Removes the current record from the business component.</td>
<td>boolean deleteRecord() throws SiebelException</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>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>
<tr>
<td>GetSearchSpec Method</td>
<td>Returns the current search specification for the field specified in the argument.</td>
<td>String getSearchSpec(String fieldName) throws SiebelException</td>
</tr>
<tr>
<td>GetUserProperty Method</td>
<td>Returns the value for the specified property.</td>
<td>String getUserProperty(String property) throws SiebelException</td>
</tr>
</tbody>
</table>
Table 42. SiebelBusComp Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetViewMode Method</td>
<td>Returns the visibility mode for the business component.</td>
<td>int getViewMode()</td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>Calls the specialized method named in the argument.</td>
<td>String invokeMethod(String methodName, String[] methodArgs) throws SiebelException</td>
</tr>
<tr>
<td>LastRecord Method</td>
<td>Moves to the last record in the business component.</td>
<td>boolean lastRecord() throws SiebelException</td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the business component.</td>
<td>String name()</td>
</tr>
<tr>
<td>NewRecord Method</td>
<td>Adds a new record to the business component.</td>
<td>boolean newRecord(boolean isInsertBefore) throws SiebelException</td>
</tr>
<tr>
<td>NextRecord Method</td>
<td>Moves to the next record in the business component.</td>
<td>boolean nextRecord() throws SiebelException</td>
</tr>
<tr>
<td>ParentBusComp Method</td>
<td>Returns the parent business component.</td>
<td>SiebelBusComp parentBusComp() throws SiebelException</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>boolean pick() throws SiebelException</td>
</tr>
<tr>
<td>PreviousRecord Method</td>
<td>Moves to the previous record in the business component.</td>
<td>boolean previousRecord() throws SiebelException</td>
</tr>
<tr>
<td>RefineQuery Method</td>
<td>Refines a query after a query has been executed.</td>
<td>boolean refineQuery() throws SiebelException</td>
</tr>
<tr>
<td>Release Method</td>
<td>Enables the release of the business component and its resources on the Siebel Server.</td>
<td>void release()</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>boolean SetFieldValue(String fieldName, String fieldValue) throws SiebelException</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>boolean SetFormattedFieldValue(String fieldName, String fieldValue) throws SiebelException</td>
</tr>
</tbody>
</table>
Table 42. SiebelBusComp Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetMultipleFieldValues Method</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>SetNamedSearch Method</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>SetSearchExpr Method</td>
<td>Sets an entire search expression on the business component.</td>
<td>boolean setSearchExpr(String searchExpr) throws SiebelException</td>
</tr>
<tr>
<td>SetSearchSpec Method</td>
<td>Sets the search specification for the specified field.</td>
<td>boolean setSearchSpec(String fieldName, String searchSpec) throws SiebelException</td>
</tr>
<tr>
<td>SetSortSpec Method</td>
<td>Sets the sort specification for a query.</td>
<td>boolean setSortSpec(String sortSpec) throws SiebelException</td>
</tr>
<tr>
<td>SetUserProperty Method</td>
<td>Sets the value of the specified User Property.</td>
<td>boolean setUserProperty(String propName, String propVal)</td>
</tr>
<tr>
<td>SetViewMode Method</td>
<td>Sets the visibility type for the business component.</td>
<td>boolean setViewMode(int mode) throws SiebelException</td>
</tr>
<tr>
<td>UndoRecord Method</td>
<td>Reverses any uncommitted changes made to the record.</td>
<td>boolean undoRecord() throws SiebelException</td>
</tr>
<tr>
<td>WriteRecord Method</td>
<td>Commits to the database any changes made to the current record.</td>
<td>boolean writeRecord() throws SiebelException</td>
</tr>
</tbody>
</table>

Table 43 lists a summary of the Siebel BusObject methods’ syntax.

Table 43. 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>SiebelBusComp getBusComp(String busCompName) throws SiebelException</td>
</tr>
</tbody>
</table>
Table 43. SiebelBusObject Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name Method</td>
<td>Returns the name of the business object.</td>
<td>String name()</td>
</tr>
<tr>
<td>Release Method</td>
<td>Enables the release of the business object and its resources on the Siebel Server.</td>
<td>void release()</td>
</tr>
</tbody>
</table>

Table 44. 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>String getFirstProperty()</td>
</tr>
<tr>
<td>GetNextProperty Method</td>
<td>Once the name of the first property has been retrieved, retrieves the name of the next property of a business service.</td>
<td>String getNextProperty()</td>
</tr>
<tr>
<td>GetProperty Method</td>
<td>Retrieves the value stored in the specified property.</td>
<td>String getProperty(String propName) throws SiebelException</td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>Calls a specialized method or a user-created method on the business service.</td>
<td>boolean invokeMethod(String methodName, SiebelPropertySet inputPropertySet, SiebelPropertySet outputPropertySet) throws SiebelException</td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the business service.</td>
<td>String Name()</td>
</tr>
<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>
</tbody>
</table>
Table 45 lists a summary of the SiebelPropertySet methods’ syntax.

<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>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 45. 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>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 getChildCount()</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>SetProperty 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 GetPropertyCount()</td>
</tr>
<tr>
<td>GetType Method</td>
<td>Returns the value stored in the Type attribute of a PropertySet.</td>
<td>String getType()</td>
</tr>
<tr>
<td>GetValue Method</td>
<td>Returns the value stored in the Value attribute of a PropertySet.</td>
<td>String 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>boolean insertChildAt(SiebelPropertySet propertySet, int index)</td>
</tr>
</tbody>
</table>
Table 45.  SiebelPropertySet 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 propertyName)</td>
</tr>
<tr>
<td>RemoveChild Method</td>
<td>Removes a child property set as a specified index from a parent property set.</td>
<td>boolean removeChild(int index)</td>
</tr>
<tr>
<td>RemoveProperty Method</td>
<td>Removes the property specified in its argument from a property set.</td>
<td>boolean removeProperty(String propertyName)</td>
</tr>
<tr>
<td>Reset Method</td>
<td>Removes every property and child property set from a property set.</td>
<td>boolean reset()</td>
</tr>
<tr>
<td>SetProperty Method</td>
<td>Assigns a value to the property of a property set specified in its argument.</td>
<td>boolean setProperty(String propertyName, String propertyValue)</td>
</tr>
<tr>
<td>SetType Method</td>
<td>Assigns a data value to a type member of a property set.</td>
<td>boolean setType(String type)</td>
</tr>
<tr>
<td>SetValue Method</td>
<td>Assigns a data value to a value member of a property set.</td>
<td>boolean setValue(String value)</td>
</tr>
</tbody>
</table>

SiebelException Methods for Java Data Bean

Table 46 lists a summary of the SiebelException methods’ syntax.

Table 46.  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: C:siebsrvr\CLASSES.
Applet Methods for Siebel VB

Table 47 lists a summary of the Applet methods’ syntax.

Table 47. Applet Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusComp Method</td>
<td>Function that returns the business component that is</td>
<td>Dim oApplet as Applet</td>
</tr>
<tr>
<td></td>
<td>associated with the applet.</td>
<td>Dim oBusComp as BusComp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set oBusComp = oApplet.BusComp</td>
</tr>
<tr>
<td>BusObject Method</td>
<td>Function that returns the business object for the</td>
<td>Dim oApplet as Applet</td>
</tr>
<tr>
<td></td>
<td>business component of the applet.</td>
<td>Dim oBusObject as BusObject</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set oBusObject = oApplet.BusObject</td>
</tr>
<tr>
<td>InvokeMethod</td>
<td>Invokes the specialized or custom method specified</td>
<td>Dim oApplet as Applet</td>
</tr>
<tr>
<td>Method</td>
<td>by its argument.</td>
<td>oApplet.InvokeMethod methodName as String, methodArgs as String or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StringArray</td>
</tr>
<tr>
<td>Name Method</td>
<td>Function that returns the name of the applet.</td>
<td>Dim oApplet as Applet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim sApplet as String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sApplet = oApplet.Name</td>
</tr>
</tbody>
</table>
Table 48 lists a summary of the WebApplet Events.

### Table 48. 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, &amp;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.InvokeMethod.</td>
<td>WebApplet_PreInvokeMethod(MethodName as String)</td>
</tr>
<tr>
<td>WebApplet_Load Event</td>
<td>Called when the applet loses focus.</td>
<td>WebApplet_Load</td>
</tr>
</tbody>
</table>
Table 48. 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</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</td>
</tr>
</tbody>
</table>

Application Methods for Siebel VB

Table 49 lists a summary of the Application methods’ syntax.

Table 49. 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 of the active view.</td>
<td>Dim oApplication as Application&lt;br&gt;Dim oBusObject as BusObject&lt;br&gt;Set oBusObject = oApplication.ActiveBusObject</td>
</tr>
<tr>
<td>ActiveViewName Method</td>
<td>Function that returns the name of the active view.</td>
<td>Dim oApplication as Application&lt;br&gt;Dim sView as String&lt;br&gt;sView = oApplication.ActiveViewName</td>
</tr>
<tr>
<td>CurrencyCode Method</td>
<td>Returns the three-letter operating currency code.</td>
<td>Dim oApplication as Application&lt;br&gt;Dim sCur as String&lt;br&gt;sCur = oApplication.CurrencyCode</td>
</tr>
<tr>
<td>GetBusObject Method</td>
<td>Instantiates and returns a new instance of the argument-specified business object.</td>
<td>Dim oApplication as Application&lt;br&gt;Dim oBusObject as BusObject&lt;br&gt;set oBusObject = oApplication.GetBusObject busobject as String</td>
</tr>
</tbody>
</table>
Table 49. Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| GetProfileAttr    | Returns the value of an attribute in a user profile.    | Dim oApplication as Application
                     | Dim sAttr as String
                     | SAttr = oApplication.GetProfileAttr(name as String)                   |
| GetService        | Instantiates and returns a new instance of the          | Dim oApplication as Application
                     | argument-specified service.                                           | Dim oService as Service
                     | set oService = oApplication.GetService(serviceName as String)          |
| GetSharedGlobal   | Gets the shared user-defined global variables.          | Dim oApplication as Application
                     | Dim sName as String
                     | sName = oApplication.GetSharedGlobal(varName as String)               |
| GotoView          | Activates the named view and its business object.       | Dim oApplication as Application
                     | oApplication.GotoView(viewName as String, [BusinessObject]BusinessObject) |
| InvokeMethod      | Calls the named specialized method.                     | Dim oApplication as Application
                     | Dim sReturn as String
                     | sReturn = oApplication.InvokeMethod(methodName as String, methodArgs as String or StringArray) |
| LoginId           | Function that returns the login ID of the user who      | Dim oApplication as Application
                     | started the Siebel application.                                      | Dim sID as String
                     | iID = oApplication.LoginId                                            |
| LoginName         | Function that returns the login name of the user who    | Dim oApplication as Application
                     | started the Siebel application.                                      | Dim sUser as String
                     | sUser = oApplication>LoginName                                        |
| NewPropertySet    | Constructs and returns a new property set object.       | Dim oApplication as Application
                     | Dim oPropSet as PropertySet
                     | oPropSet = oApplication.NewPropertySet()                              |
| PositionId        | Function that returns the position ID that describes    | Dim oApplication as Application
                     | the user’s current position.                                          | Dim sRow as String
                     | sRow = oApplication.PositionId                                        |
| PositionName      | Function that returns the position name of the user’s  | Dim oApplication as Application
                     | current position.                                                     | Dim sPosition as String
                     | sPosition = oApplication.PositionName                                  |
| RaiseError        | Raises a scripting error message to the browser. The    | Dim oApplication as Application
                     | error code is a canonical number.                                     | oApplication.RaiseError(keyValue as String, param1 as String, ...) |
Table 49. Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<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 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 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 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 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 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 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 oApplication.TraceOff</td>
</tr>
<tr>
<td>TraceOn Method</td>
<td>Turns tracing on.</td>
<td>Dim oApplication as Application oApplication.TraceOn filename as String, type as String, selection as String</td>
</tr>
</tbody>
</table>

Table 50 lists a summary of the Application Events.

Table 50. 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</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)</td>
</tr>
</tbody>
</table>
Table 50. Application Events Summary

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application_PreNavigate</td>
<td>Called before the client has navigated from one view to the next.</td>
<td><code>Application_PreNavigate (DestViewName As String, DestBusObjName As String)</code></td>
</tr>
<tr>
<td>Application_Start</td>
<td>Called when the client starts.</td>
<td><code>Application_Start(commandLine as String)</code></td>
</tr>
</tbody>
</table>

Table 51 lists a summary of the Business Component methods’ syntax.

Table 51. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| ActivateField Method | Allows queries to retrieve data for the specified field. | `Dim oBusComp as BusComp
oBusComp.ActivateField fieldName as String` |
| ActivateMultipleFields Method | Allows queries to retrieve data for the fields specified in the property set. | `Dim oBusComp as BusComp
oBusComp.ActivateMultipleFields oPropSet as PropertySet` |
| Associate Method     | Creates a new many-to-many relationship for the parent object through an association business component. | `Dim oBusComp as BusComp
oBusComp.Associate whereIndicator as Integer` |
| BusObject Method     | Function that returns the business object that contains the business component. | `Dim oBusComp as BusComp
Dim oBusObject as BusObject
Set oBusObject = oBusComp.BusObject` |
| ClearToQuery Method  | Clears the current query and sort specifications on the business component. | `Dim oBusComp as BusComp
oBusComp.ClearToQuery` |
| DeactivateFields Method | Deactivates every currently activated field. | `Dim oBusComp as BusComp
oBusComp.DeactivateFields` |
| DeleteRecord Method  | Removes the current record from the business component. | `Dim oBusComp as BusComp
oBusComp.DeleteRecord` |
| ExecuteQuery Method  | Retrieves a set of BusComp records.               | `Dim oBusComp as BusComp
oBusComp.ExecuteQuery cursorMode as Integer` |
### Table 51. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| ExecuteQuery2 Method    | Retrieves a set of BusComp records.                                         | Dim oBusComp as BusComp  
                             oBusComp.ExecuteQuery2 cursorMode as Integer, ignoreMaxCursorSize as Integer |
| FirstRecord Method      | Moves to the first record in the business component.                       | Dim oBusComp as BusComp  
                             Dim iIsRecord as Integer  
                             iIsRecord = oBusComp.FirstRecord |
| FirstSelected Method    | Moves the focus to the first record of the multiple selection in the business component. | Dim oBusComp as BusComp  
                             Dim iIsMultipleSelection as Integer  
                             iIsMultipleSelection = oBusComp.FirstSelected |
| GetAssocBusComp Method  | Function that returns the association business component.                   | Dim oBusComp as BusComp  
                             Dim AssocBusComp as BusComp  
                             Set AssocBusComp = oBusComp.GetAssocBusComp |
| GetFieldValue Method    | Function that returns a value for the argument-specified field.             | Dim oBusComp as BusComp  
                             Dim sValue as String  
                             sValue = oBusComp.GetFieldValue(FieldName as String) |
| GetFormattedFieldValue 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 oFields 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) |
### Table 51. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| **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) |
| **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 |
| **InvokeMethod Method**| Calls the specialized method or user-created method specified in the argument. | Dim oBusComp as BusComp  
Dim Return  
Return = oBusComp.InvokeMethod(methodName as String, methodArgs as String or StringArray) |
| **LastRecord Method**  | Moves to the last record in the business component.  | Dim oBusComp as BusComp  
Dim iReturn as Integer  
iReturn = oBusComp.LastRecord |
| **Name Method**         | Function that returns the name of the business component. | Dim oBusComp as BusComp  
Dim sName as String  
sName = oBusComp.Name |
| **NewRecord Method**    | Adds a new record to the business component.          | Dim oBusComp as BusComp  
Dim oBusComp.NewRecord(whereIndicator as Integer) |
| **NextRecord Method**   | Moves to the next record in the business component.  | Dim oBusComp as BusComp  
Dim iReturn as Integer  
iReturn = oBusComp.NextRecord |
| **NextSelected Method** | Moves to the next record of the current multiple selection. | Dim oBusComp as BusComp  
Dim iReturn as Integer  
iReturn = oBusComp.NextSelected |
| **ParentBusComp Method**| Function that returns the parent business component.  | Dim oBusComp as BusComp  
Dim parentBusComp as BusComp  
Set parentBusComp = oBusComp.ParentBusComp |
### Table 51. Business Component Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| Pick Method             | Places the currently selected record in a picklist business component into the appropriate fields of the parent business component. | Dim oBusComp as BusComp  
oBusComp.Pick                             |
| PreviousRecord Method   | Moves to the previous record in the business component.                      | Dim oBusComp as BusComp  
Dim iReturn as Integer  
iReturn = oBusComp.PreviousRecord |
| RefineQuery Method      | Refines a query after a query has been executed.                             | Dim oBusComp as BusComp  
oBusComp.RefineQuery                      |
| SetFieldVal 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 |
| SetFormattedVal 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 |
Table 52 lists a summary of the Business Component Events.

Table 52. Business Component Events Summary

<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>
</tbody>
</table>
### Table 52. Business Component Events Summary

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<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>
<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 53 lists a summary of the Business Object methods’ syntax.

Table 53. 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>Function that returns the specified business component.</td>
<td>Dim oBusObject as BusObject&lt;br&gt;Dim oBusComp as BusComp&lt;br&gt;set oBusComp = oBusObject.GetBusComp(BusCompName as String)</td>
</tr>
<tr>
<td>Name Method</td>
<td>Function that returns the name of the business object.</td>
<td>Dim oBusObject as BusObject&lt;br&gt;Dim sName as String&lt;br&gt;sName = oBusObject.Name</td>
</tr>
</tbody>
</table>

Business Service Methods for Siebel VB

Table 54 lists a summary of the Business Service methods’ syntax.

Table 54. 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 Service&lt;br&gt;Dim sName as String&lt;br&gt;sName = oService.GetFirstProperty()</td>
</tr>
<tr>
<td>GetNextProperty Method</td>
<td>Once 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 Service&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 Service&lt;br&gt;Dim sValue as String&lt;br&gt;sValue = oService.GetProperty(propName as String)</td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>Calls a specialized method or a user-created method on the business service.</td>
<td>Dim oService as Service&lt;br&gt;Dim Return&lt;br&gt;Return = oService.InvokeMethod(methodName as String, InputArguments as PropertySet, OutputArguments as PropertySet)</td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the business service.</td>
<td>Dim oService as Service&lt;br&gt;Dim sName as String&lt;br&gt;sName = oService.Name</td>
</tr>
</tbody>
</table>
Table 54. Business Service 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>Dim oService as Service oService.PropertyExists(propName as String)</td>
</tr>
<tr>
<td>RemoveProperty Method</td>
<td>Removes a property from a business service.</td>
<td>Dim oService as Service oService.RemoveProperty propName as String</td>
</tr>
<tr>
<td>SetProperty Method</td>
<td>Assigns a value to a property of a business service.</td>
<td>Dim oService as Service oService.SetProperty propName as String, propValue as String</td>
</tr>
</tbody>
</table>

Table 55 lists a summary of the Business Service Events.

Table 55. 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 56 lists a summary of the Property Set methods’ syntax.

### Table 56. 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. | Dim oPropSet as PropertySet  
oPropSet.AddChild(childObject as PropertySet) |
| Copy Method       | Returns a copy of a property set.                | Dim oPropSet1 as PropertySet  
Dim oPropSet2 as PropertySet  
set oPropSet2 = oPropSet1.Copy() |
| GetChild Method   | Returns a specified child property set of a property set. | Dim oPropSet as PropertySet  
Dim childPropSet as SiebelPropertySet  
set childPropSet = oPropSet.GetChild(index as Long) |
| GetChildCount Method | Returns the number of child property sets attached to a parent property set. | Dim oPropSet as PropertySet  
Dim iCount as Integer  
iCount = oPropSet.GetChildCount() |
| GetFirstProperty Method | Returns the name of the first property in a property set. | Dim oPropSet as PropertySet  
Dim sPropName as String  
sPropName = oPropSet.GetFirstProperty() |
| GetNextProperty Method | Returns the name of the next property in a property set. | Dim oPropSet as PropertySet  
Dim sPropName as String  
sPropName = oPropSet.GetNextProperty() |
| GetProperty Method | Returns the value of a property when given the property name. | Dim oPropSet as PropertySet  
Dim sPropVal as String  
sPropVal = oPropSet.GetProperty(propName as String) |
| GetPropertyCount Method | Returns the number of properties attached to a property set. | Dim oPropSet as PropertySet  
Dim count as Long  
count = oPropSet.GetPropertyCount |
| GetType Method    | Returns the value stored in a type in a property set. | Dim oPropSet as PropertySet  
Dim sTypeVal as String  
sTypeVal = oPropSet.GetType |
| GetValue Method   | Returns a value stored as part of a property set. | Dim oPropSet as PropertySet  
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 PropertySet  
oPropSet.InsertChildAt childObject as SiebelPropertySet, index as Integer |
### Property Set 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><code>Dim oPropSet as PropertySet</code>&lt;br&gt;<code>oPropSet.PropertyExists(propName as String)</code></td>
</tr>
<tr>
<td>GetPropertyCount Method</td>
<td>Returns the number of properties attached to a property set.</td>
<td><code>Dim oPropSet as PropertySet</code>&lt;br&gt;<code>Dim count as Long</code>&lt;br&gt;<code>count=oPropSet.GetPropertyCount</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>Dim oPropSet as PropertySet</code>&lt;br&gt;<code>oPropSet.RemoveChild(index as Integer)</code></td>
</tr>
<tr>
<td>RemoveProperty Method</td>
<td>Removes the property specified in its argument from a property set.</td>
<td><code>Dim oPropSet as PropertySet</code>&lt;br&gt;<code>oPropSet.RemoveProperty(propName as String)</code></td>
</tr>
<tr>
<td>Reset Method</td>
<td>Removes every property and child property set from a property set.</td>
<td><code>Dim oPropSet as PropertySet</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>Dim oPropSet as PropertySet</code>&lt;br&gt;<code>oPropSet.SetProperty(propName as String, propName as String)</code></td>
</tr>
<tr>
<td>SetType Method</td>
<td>Assigns a data value to a type member of a property set.</td>
<td><code>Dim oPropSet as PropertySet</code>&lt;br&gt;<code>oPropSet.SetType(value as String)</code></td>
</tr>
<tr>
<td>SetValue Method</td>
<td>Assigns a data value to a value member of a property set.</td>
<td><code>Dim oPropSet as PropertySet</code>&lt;br&gt;<code>oPropSet.SetValue(value as String)</code></td>
</tr>
</tbody>
</table>
## Miscellaneous Methods for Siebel VB

Table 57 lists a summary of the Miscellaneous methods’ syntax.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>TheApplication</td>
<td>Global method that returns the unique object of type Application.</td>
<td>TheApplication</td>
</tr>
</tbody>
</table>
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 69 and Table 70 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, read *Siebel Developer’s Reference*.

### Browser Script Events and Methods

The following is a list of the Events and Methods available in Browser Script:

- “Applet Methods for Browser Script” on page 390
- “Application Methods for Browser Script” on page 391
- “Business Component Methods for Browser Script” on page 393
- “Business Object Methods for Browser Script” on page 394
- “Business Service Methods for Browser Script” on page 395
- “PropertySet Methods for Browser Script” on page 396
- “Control Methods for Browser Script” on page 398

**See Also**

“Supported DOM Events for High Interactivity Mode” on page 399
“Supported DOM Events for Standard Interactivity Mode” on page 400
Applet Methods for Browser Script

Table 58 lists a summary of the Applet methods’ syntax.

Table 58. Applet Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveMode Method</td>
<td>Returns a string containing the name of the current Web Template mode.</td>
<td><code>var oApplet; var mode = oApplet.ActiveMode();</code></td>
</tr>
<tr>
<td>BusComp Method</td>
<td>Returns the business component that is associated with the applet.</td>
<td><code>var oApplet; var busComp = oApplet.BusComp();</code></td>
</tr>
<tr>
<td>BusObject Method</td>
<td>Returns the business object for the business component for the applet.</td>
<td><code>var oApplet; var oBusObject = oApplet.BusObject();</code></td>
</tr>
<tr>
<td>FindActiveXControl Method</td>
<td>Returns the ActiveX control whose name is specified in the argument.</td>
<td><code>var oApplet; var oControl; oControl = oApplet.FindActiveXControl(controlName);</code></td>
</tr>
<tr>
<td>FindControl Method</td>
<td>Returns the control whose name is specified in the argument.</td>
<td><code>var oApplet; var oControl; oControl = oApplet.FindControl(controlName);</code></td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>Calls an argument-specified specialized method.</td>
<td><code>var oApplet; var outPs = theApplication().NewPropertySet(); outPs = oApplet.InvokeMethod(MethodName, inputPropSet);</code></td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the applet.</td>
<td><code>var oApplet; var name = oApplet.Name();</code></td>
</tr>
</tbody>
</table>

Table 59 lists a summary of the Applet Events.

Table 59. Applet Events Summary

<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><code>Applet_ChangeFieldValue(field, value)</code></td>
</tr>
<tr>
<td>Applet_ChangeRecord Event</td>
<td>Called when the user moves to a different row or view.</td>
<td><code>Applet_ChangeRecord()</code></td>
</tr>
</tbody>
</table>
Table 59. Applet Events Summary

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<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>

Table 60 lists a summary of the Application methods’ syntax.

Table 60. Application Methods Syntax Summary

<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>
<tr>
<td>ActiveViewName Method</td>
<td>Returns the name of the active view.</td>
<td>var viewName; viewName = theApplication().ActiveViewName();</td>
</tr>
<tr>
<td>FindApplet Method</td>
<td>Returns the applet object identified in the argument.</td>
<td>var applet; applet = theApplication().FindApplet(appletName);</td>
</tr>
<tr>
<td>GetProfileAttr Method</td>
<td>Returns the value of an attribute in a user profile.</td>
<td>var sAttr; sAttr = theApplication().GetProfileAttr(name);</td>
</tr>
</tbody>
</table>

Application Methods for Browser Script

Table 60 lists a summary of the Application methods’ syntax.
Browser Scripting  ■ Application Methods for Browser Script

Table 60. Application Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetService Method</td>
<td>Instantiates and returns a new instance of the service specified in the argument.</td>
<td>var svc; svc = theApplication().getService(serviceName);</td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>Calls the named specialized method.</td>
<td>var outPs; outPs = theApplication().InvokeMethod(methodName, inputPropSet);</td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns name of the application.</td>
<td>var appName; appName = theApplication().Name();</td>
</tr>
<tr>
<td>NewPropertySet Method</td>
<td>Constructs and returns a new property set object.</td>
<td>var PropSet; PropSet = theApplication().NewPropertySet();</td>
</tr>
<tr>
<td>SetProfileAttr Method</td>
<td>Used in personalization to assign values to attributes in a user profile.</td>
<td>theApplication().SetProfileAttr(name, value);</td>
</tr>
<tr>
<td>SWEAlert Method</td>
<td>Displays a modal dialog box containing a message to the user.</td>
<td>theApplication().SWEAlert(message);</td>
</tr>
</tbody>
</table>

Table 61 lists a summary of the Application Events syntax.

Table 61. 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 method is invoked.</td>
<td>Application_InvokeMethod(name, inputPropSet)</td>
</tr>
<tr>
<td>Application_PreInvokeMethod Event</td>
<td>Called before a specialized method is invoked.</td>
<td>Application_PreInvokeMethod(name, inputPropSet)</td>
</tr>
</tbody>
</table>
## Business Component Methods for Browser Script

Table 62 lists a summary of the Business Component methods’ syntax.

### Table 62. 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 that contains the business component.</td>
<td><code>var busComp; var busObject; busObject = busComp.BusObject();</code></td>
</tr>
<tr>
<td>GetFieldValue Method</td>
<td>Returns a value for the field specified in the argument.</td>
<td><code>var busComp; var value; value = busComp.GetFieldValue(fieldName);</code></td>
</tr>
<tr>
<td>GetFormattedFieldValue Method</td>
<td>Returns a formatted value for the field specified in the argument.</td>
<td><code>var busComp; var sValue; sValue = busComp.GetFormattedFieldValue(fieldName);</code></td>
</tr>
<tr>
<td>GetSearchExpr Method</td>
<td>Returns the current search expression.</td>
<td><code>var busComp; var sExpr; sExpr = busComp.GetSearchExpr();</code></td>
</tr>
<tr>
<td>GetSearchSpec Method</td>
<td>Returns the current search specification for the field specified in the argument.</td>
<td><code>var busComp; var sSpec; sSpec = busComp.GetSearchSpec(fieldName);</code></td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the business component.</td>
<td><code>var busComp; var sName; 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 component.</td>
<td><code>var busComp; busComp.SetFieldValue(fieldName, fieldValue);</code></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; 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; busComp.WriteRecord();</code></td>
</tr>
</tbody>
</table>
Table 63 lists a summary of the Business Component Events syntax.

Table 63. Business Component Events Syntax Summary

<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 pushed down into the business component from the user interface. This Browser Script event is not invoked if the 'Immediate Post Changes' property of the Business Component field is set to TRUE.</td>
<td>BusComp_PreSetFieldValue(fieldName, value);</td>
</tr>
</tbody>
</table>

Business Object Methods for Browser Script

Table 64 lists a summary of the Business Object methods’ syntax.

Table 64. 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>var busObject; var Comp; 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 65 lists a summary of the Business Service methods’ syntax.

## Table 65. 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>var svc;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var sName = svc.GetFirstProperty();</td>
</tr>
<tr>
<td>GetNextProperty Method</td>
<td>Once the name of the first property has been retrieved, retrieves the name of the next property of a business service.</td>
<td>var svc;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var sName = svc.GetNextProperty();</td>
</tr>
<tr>
<td>GetProperty Method</td>
<td>Retrieves the value stored in the specified property.</td>
<td>var svc;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var value;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value = svc.GetProperty(name);</td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>Calls a specialized method or a user-created method on the business service.</td>
<td>var svc;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var oPropSet = theApplication().NewPropertySet();</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet = svc.InvokeMethod(methodName, inputPropSet);</td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the business service.</td>
<td>var svc;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var name;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>name = svc.Name();</td>
</tr>
<tr>
<td>PropertyExists Method</td>
<td>Returns a Boolean value indicating whether the property specified in the argument exists.</td>
<td>var svc;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var bool;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bool = svc.PropertyExists(name);</td>
</tr>
<tr>
<td>RemoveProperty Method</td>
<td>Removes a property from a business service.</td>
<td>var svc;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>svc.RemoveProperty(name);</td>
</tr>
<tr>
<td>SetProperty Method</td>
<td>Assigns a value to a property of a business service.</td>
<td>var svc;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>svc.SetProperty(name, value);</td>
</tr>
</tbody>
</table>
Table 66 lists a summary of the Business Service Events syntax.

Table 66. Business Service Events Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Service_InvokeMethod Event</strong></td>
<td>Called when a business service is accessed.</td>
<td><code>Service_InvokeMethod(methodName, input, output);</code></td>
</tr>
<tr>
<td><strong>Service_PreCanInvokeMethod Event</strong></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><code>Service_PreCanInvokeMethod(methodName);</code></td>
</tr>
<tr>
<td><strong>Service_PreInvokeMethod Event</strong></td>
<td>Called before a specialized method is invoked on a business service.</td>
<td><code>Service_PreInvokeMethod(methodName, inputPropSet, outputPropSet);</code></td>
</tr>
</tbody>
</table>

**PropertySet Methods for Browser Script**

Table 67 lists a summary of the PropertySet methods’ syntax.

Table 67. PropertySet Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddChild Method</strong></td>
<td>Adds subsidiary property sets to a property set.</td>
<td><code>var oPropSet; var iIndex; iIndex = oPropSet.AddChild(childObject);</code></td>
</tr>
<tr>
<td><strong>Copy Method</strong></td>
<td>Returns a copy of a property set.</td>
<td><code>var oPropSet1; var oPropSet2; oPropSet2 = oPropSet1.Copy();</code></td>
</tr>
<tr>
<td><strong>GetChild Method</strong></td>
<td>Returns a specified child property set of a property set.</td>
<td><code>var oPropSet; var oChildPropSet; oChildPropSet = oPropSet.GetChild(index);</code></td>
</tr>
<tr>
<td><strong>GetChildCount Method</strong></td>
<td>Returns the number of child property sets attached to a parent property set.</td>
<td><code>var oPropSet; var iCount; iCount = oPropSet.GetChildCount();</code></td>
</tr>
<tr>
<td><strong>GetFirstProperty Method</strong></td>
<td>Returns the name of the first property in a property set.</td>
<td><code>var oPropSet; var sPropName; sPropName = oPropSet.GetFirstProperty();</code></td>
</tr>
</tbody>
</table>
### Table 67. PropertySet Methods Syntax Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetNextProperty Method</td>
<td>Returns the name of the next property in a property set.</td>
<td><code>var oPropSet; var sPropName; 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; var sValue; 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; var iCount; 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; var type; 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; var sValue; 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; 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; var bool; 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; 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; 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; 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; oPropSet SetProperty(propName, propValue);</code></td>
</tr>
</tbody>
</table>
### Control Methods for Browser Script

Table 68 lists a summary of the Control methods’ syntax.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applet Method</td>
<td>Returns the parent applet for the control.</td>
<td>var oControl; var oApplet; oApplet = oControl.Applet();</td>
</tr>
<tr>
<td>BusComp Method</td>
<td>Returns the corresponding business component for the control.</td>
<td>var oControl; var busComp; busComp = oControl.Buscomp();</td>
</tr>
<tr>
<td>GetProperty Method</td>
<td>Returns the value of the property of a control.</td>
<td>var oControl; var propVal; propVal = oControl.GetProperty(propName);</td>
</tr>
<tr>
<td>GetValue Method</td>
<td>Returns the value of a control.</td>
<td>var oControl; var sValue; sValue = oControl.GetValue();</td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the control.</td>
<td>var oControl; var sName; sName = oControl.Name();</td>
</tr>
<tr>
<td>SetProperty Method</td>
<td>Sets the visual properties of a control.</td>
<td>var oControl; oControl.SetProperty(propName, propValue);</td>
</tr>
<tr>
<td>SetValue Method</td>
<td>Sets the contents of the control to the indicated value.</td>
<td>var oControl; oControl.SetValue(value);</td>
</tr>
</tbody>
</table>
## Supported DOM Events for High Interactivity Mode

Table 69 lists the supported DOM Events for high interactivity mode.

Table 69. 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</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnBlur</td>
<td></td>
</tr>
<tr>
<td>CheckBox</td>
<td>Native</td>
<td>OnFocus</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnBlur</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rendered as Input Type=CHECKBOX.</td>
<td></td>
</tr>
<tr>
<td>Link</td>
<td>Native</td>
<td>OnFocus</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnBlur</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rendered through paired anchor tags or as INPUT TYPE = TEXT in edit mode.</td>
<td></td>
</tr>
<tr>
<td>List Column</td>
<td>Native</td>
<td>This control does not expose any scriptable events.</td>
<td></td>
</tr>
<tr>
<td>Mailto</td>
<td>Native</td>
<td>OnFocus</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnBlur</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rendered as anchor tags with HREF=mailto or as INPUT TYPE=TEXT in Edit mode.</td>
<td></td>
</tr>
<tr>
<td>MiniButton</td>
<td>Native</td>
<td>OnFocus</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnBlur</td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td>Native</td>
<td>OnFocus</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnBlur</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rendered as Input Type = password.</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td>Native</td>
<td>OnFocus</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnBlur</td>
<td></td>
</tr>
<tr>
<td></td>
<td></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>
<td></td>
</tr>
<tr>
<td>TextArea</td>
<td>Native</td>
<td>OnFocus</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnBlur</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rendered as TEXTAREA.</td>
<td></td>
</tr>
<tr>
<td>Tree</td>
<td>Native</td>
<td>Tree applets and controls do not expose any scriptable events.</td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td>Native</td>
<td>OnFocus</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnBlur</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rendered through paired anchor tags with an HREF = underlying field value or as INPUT TYPE = TEXT in edit mode.</td>
<td></td>
</tr>
</tbody>
</table>
**Supported DOM Events for Standard Interactivity Mode**

Table 70 lists the 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)</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>CheckBox</td>
<td>Native</td>
<td>OnBlur (Base/Edit)</td>
<td>In Base mode, a CheckBox appears as a Y or N text value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnFocus (Edit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnChange (Edit)</td>
<td>In Edit mode, a CheckBox is rendered as Input Type=CHECKBOX.</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>Link</td>
<td>Native</td>
<td>OnFocus (Base/Edit)</td>
<td>Rendered through paired anchor tags or as INPUT TYPE = TEXT in Edit mode.</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>List Column</td>
<td>Native</td>
<td>List Columns currently do not expose any scriptable events.</td>
<td></td>
</tr>
</tbody>
</table>
Table 70. 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) OnBlur (Base/Edit) OnMouseOut (Base/Edit) OnMouseOver (Base/Edit)</td>
<td></td>
</tr>
<tr>
<td>MiniButton</td>
<td>Native</td>
<td>OnFocus (Base/Edit) OnBlur (Base/Edit) OnMouseOut (Base/Edit) OnMouseOver (Base/Edit) 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) 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) 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) OnMouseOver (Edit)</td>
<td></td>
</tr>
</tbody>
</table>
### Table 70. 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>Tree</td>
<td>Native</td>
<td>At this time, tree applets and controls do not expose any scriptable events.</td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td>Native</td>
<td>OnChange (Edit) OnFocus (Base/Edit) OnBlur (Base/Edit) OnMouseOut (Base/Edit) OnMouseOver (Base/Edit)</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>
This quick reference has the following topics:

- "Applet Methods for eScript"
- "Application Methods for eScript" on page 405
- "Business Component Methods for eScript" on page 407
- "Business Object Methods for eScript" on page 413
- "Business Service Methods for eScript" on page 414
- "PropertySet Methods for eScript" on page 415
- "Miscellaneous Methods for eScript" on page 417

## Applet Methods for eScript

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>
<tbody>
<tr>
<td>BusComp Method</td>
<td>Returns the business component that is associated with the applet.</td>
<td>var applet;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp = applet.BusComp();</td>
</tr>
<tr>
<td>BusObject Method</td>
<td>Returns the business object for the business component for the applet.</td>
<td>var applet;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var busObject;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>busObject = applet.BusObject();</td>
</tr>
<tr>
<td>InvokeMethod</td>
<td>Calls an argument-specified specialized method.</td>
<td>var applet;</td>
</tr>
<tr>
<td>Method</td>
<td></td>
<td>applet.InvokeMethod(methodName, methodArg1, methodArg2, ...)</td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the applet.</td>
<td>var applet;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var sApplet;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sApplet = applet.Name();</td>
</tr>
</tbody>
</table>
Table 72 lists a summary of the WebApplet Events.

Table 72. 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(Method Name);</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(Method Name, &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>
<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>
### Application Methods for eScript

Table 73 lists a summary of the Application methods’ syntax.

#### Table 73. 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><code>var busObject; busObject = TheApplication().ActiveBusObject();</code></td>
</tr>
<tr>
<td>ActiveViewName Method</td>
<td>Returns the name of the active view.</td>
<td><code>var sView; sView = TheApplication().ActiveViewName();</code></td>
</tr>
<tr>
<td>CurrencyCode Method</td>
<td>Returns the three-letter operating currency code.</td>
<td><code>var sCur; sCur = TheApplication().CurrencyCode();</code></td>
</tr>
<tr>
<td>GetBusObject Method</td>
<td>Instantiates and returns a new instance of the business object specified in the argument.</td>
<td><code>var myBusObject; myBusObject = TheApplication().GetBusObject(BusObjectName);</code></td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the application.</td>
<td><code>var name; name = TheApplication().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 Service; Service = TheApplication().GetService(serviceName);</code></td>
</tr>
<tr>
<td>GetSharedGlobal Method</td>
<td>Gets the shared user-defined global variables.</td>
<td><code>var sName; sName = TheApplication().GetSharedGlobal(varName);</code></td>
</tr>
<tr>
<td>GotoView Method</td>
<td>Activates the named view and its business object.</td>
<td><code>TheApplication().GotoView(viewName,[BusinessObject]);</code></td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>Calls the named specialized method.</td>
<td><code>TheApplication().InvokeMethod(methodName, methodArg1, methodArg2,..., methodArgn);</code></td>
</tr>
<tr>
<td>LoginId Method</td>
<td>Returns the login ID of the user who started the Siebel application.</td>
<td><code>var sID; sID = TheApplication().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>var sUser; sUser = TheApplication().LoginName();</code></td>
</tr>
<tr>
<td>NewPropertySet Method</td>
<td>Constructs and returns a new property set object.</td>
<td><code>var oPropSet; oPropSet = TheApplication().NewPropertySet();</code></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>PositionId Method</td>
<td>Returns the position ID that describes the user’s current position.</td>
<td><code>var sRow; sRow = TheApplication().PositionId();</code></td>
</tr>
<tr>
<td>PositionName Method</td>
<td>Returns the position name of the user’s current position.</td>
<td><code>var sPosition; sPosition = TheApplication().PositionName();</code></td>
</tr>
<tr>
<td>RaiseError Method</td>
<td>Raises a scripting error message to the browser. The error code is a canonical number.</td>
<td><code>var keyVal; var arg1 ...; TheApplication().RaiseError(keyVal, arg1, ...);</code></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><code>var message; TheApplication().RaiseErrorText(message);</code></td>
</tr>
<tr>
<td>SetPositionId Method</td>
<td>Sets the active position to the position ID specified in the argument.</td>
<td><code>var success; success = TheApplication().SetPositionId(posId);</code></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 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 74 lists a summary of the Application Events syntax.

Table 74. 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>Application_Close();</td>
</tr>
<tr>
<td>Application_InvokeMethod</td>
<td>Called after a specialized method is invoked.</td>
<td>Application_InvokeMethod(methodName);</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_PreInvokeMethod</td>
<td>Called before a specialized method is invoked.</td>
<td>Application_PreInvokeMethod(methodName);</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, DestBusObjName)</td>
</tr>
<tr>
<td>Application_Start Event</td>
<td>Called when the client starts.</td>
<td>Application_Start(commandLine);</td>
</tr>
</tbody>
</table>

Business Component Methods for eScript

Table 75 lists a summary of the Business Component methods’ syntax.

Table 75. 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>var myBusComp; myBusComp.ActivateField(fieldName);</td>
</tr>
<tr>
<td>ActivateMultipleFields</td>
<td>Allows queries to retrieve data for the fields specified in the property set.</td>
<td>var myBusComp; myBusComp.ActivateMultipleFields(oPropSet);</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>var myBusComp; myBusComp.Associate(whereIndicator);</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>BusObject Method</td>
<td>Returns the business object that contains the business component.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var busObject;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>busObject = myBusComp.BusObject();</td>
</tr>
<tr>
<td>ClearToQuery Method</td>
<td>Clears the current query and sort specifications on the business component.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.ClearToQuery();</td>
</tr>
<tr>
<td>DeactivateFields Method</td>
<td>Deactivates every currently activated field.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.DeactivateFields();</td>
</tr>
<tr>
<td>DeleteRecord Method</td>
<td>Removes the current record from the business component.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.DeleteRecord();</td>
</tr>
<tr>
<td>ExecuteQuery Method</td>
<td>Retrieves a set of BusComp records.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myComp.ExecuteQuery(cursorMode);</td>
</tr>
<tr>
<td>ExecuteQuery2 Method</td>
<td>Retrieves a set of BusComp records.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.ExecuteQuery2(cursorMode, ignoreMaxCursorSize);</td>
</tr>
<tr>
<td>FirstRecord Method</td>
<td>Moves to the first record in the business component.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var bIsRecord;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bIsRecord = myBusComp.FirstRecord();</td>
</tr>
<tr>
<td>FirstSelected Method</td>
<td>Moves to the first record of the multiple selection in the business component.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var bIsMultipleSelection;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bIsMultipleSelection = myBusComp.FirstSelected();</td>
</tr>
<tr>
<td>GetAssocBusComp Method</td>
<td>Returns the association business component.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var AssocBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AssocBusComp = myBusComp.GetAssocBusComp();</td>
</tr>
<tr>
<td>GetFieldValue Method</td>
<td>Returns a value for the field specified in the argument.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var sValue;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sValue = myBusComp.GetFieldValue(FieldName);</td>
</tr>
<tr>
<td>GetFormattedFieldValue Method</td>
<td>Returns a formatted value for the field specified in the argument.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var sValue;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sValue = myBusComp.GetFormattedFieldValue(FieldName);</td>
</tr>
<tr>
<td>GetMultipleFieldValues Method</td>
<td>Returns a value for the fields specified in the property set.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.GetMultipleFieldValues(fields, oValues);</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>GetMVGBusComp Method</td>
<td>Returns the MVG business component associated with the field specified in the argument.</td>
<td>var myBusComp; var MvgBusComp; MvgBusComp = myBusComp.GetMVGBusComp(FieldName);</td>
</tr>
<tr>
<td>GetNamedSearch Method</td>
<td>Returns the named search specification specified in the argument.</td>
<td>var myBusComp; var sValue; sValue = myBusComp.GetNamedSearch(SearchName);</td>
</tr>
<tr>
<td>GetPicklistBusComp Method</td>
<td>Returns the pick business component associated with the field specified in the argument.</td>
<td>var myBusComp; var pickBusComp; pickBusComp = myBusComp.GetPicklistBusComp(FieldName);</td>
</tr>
<tr>
<td>GetSearchExpr Method</td>
<td>Returns the current search expression.</td>
<td>var myBusComp; var sExpr; sExpr = myBusComp.GetSearchExpr();</td>
</tr>
<tr>
<td>GetSearchSpec Method</td>
<td>Returns the current search specification for the field specified in the argument.</td>
<td>var myBusComp; var sSpec; sSpec = myBusComp.GetSearchSpec(FieldName);</td>
</tr>
<tr>
<td>GetUserProperty Method</td>
<td>Returns the value for a property name specified in the argument.</td>
<td>var myBusComp; var sValue; sValue = myBusComp.GetUserProperty(propertyName);</td>
</tr>
<tr>
<td>GetViewMode Method</td>
<td>Returns the visibility mode for the business component.</td>
<td>var myBusComp; var iMode; iMode = myBusComp.GetViewMode();</td>
</tr>
<tr>
<td>InvokeMethod Method</td>
<td>Calls the specialized method named in the argument.</td>
<td>var myBusComp; var sReturn; sReturn = myBusComp.InvokeMethod(methodName, methodArg1, methodArg2,..., methodArgn);</td>
</tr>
<tr>
<td>LastRecord Method</td>
<td>Moves to the last record in the business component.</td>
<td>var myBusComp; var iReturn; iReturn = myBusComp.LastRecord();</td>
</tr>
<tr>
<td>Name Method</td>
<td>Returns the name of the business component.</td>
<td>var myBusComp; var sName; sName = myBusComp.Name();</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>NewRecord Method</td>
<td>Adds a new record to the business component.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.NewRecord( whereIndicator );</td>
</tr>
<tr>
<td>NextRecord Method</td>
<td>Moves to the next record in the business component.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var bFound;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bFound = myBusComp.NextRecord();</td>
</tr>
<tr>
<td>NextSelected Method</td>
<td>Moves to the next record of the current multiple selection.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var iReturn;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>iReturn = myBusComp.NextSelected();</td>
</tr>
<tr>
<td>ParentBusComp Method</td>
<td>Returns the parent business component.</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var parentBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>parentBusComp =</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.ParentBusComp();</td>
</tr>
<tr>
<td>Pick Method</td>
<td>Places the currently selected record in a picklist business component into</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the appropriate fields of the parent business component.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.Pick();</td>
</tr>
<tr>
<td>PreviousRecord Method</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>RefineQuery Method</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>SetFieldValue Method</td>
<td>Assigns a new value to the named field for the current row of the</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>business component.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.SetFieldValue( FieldName, FieldValue );</td>
</tr>
<tr>
<td>SetFormattedFieldValue Method</td>
<td>Accepts the field value in the current local format and assigns the new</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value to the named field for the current row of the business</td>
</tr>
<tr>
<td></td>
<td></td>
<td>component.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.SetFormattedFieldValue( FieldName, FieldValue );</td>
</tr>
<tr>
<td>SetMultipleFieldValues Method</td>
<td>Assigns a new value to the fields specified in the property set for the</td>
<td>var myBusComp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>current row of the business component.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>myBusComp.SetMultipleFieldValues( oPropSet );</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>SetNamedSearch Method</td>
<td>Sets a named search specification on the business component.</td>
<td>var myBusComp; myBusComp.SetNamedSearch(searchName, searchSpec);</td>
</tr>
<tr>
<td>SetSearchExpr Method</td>
<td>Sets the search specification for the business component.</td>
<td>var myBusComp; myBusComp.SetSearchExpr(searchSpec);</td>
</tr>
<tr>
<td>SetSearchSpec Method</td>
<td>Sets the search specification for the specified field.</td>
<td>var myBusComp; myBusComp.SetSearchSpec(FieldName, searchSpec);</td>
</tr>
<tr>
<td>SetSortSpec Method</td>
<td>Sets the sort specification for a query.</td>
<td>var myBusComp; myBusComp.SetSortSpec(sortSpec);</td>
</tr>
<tr>
<td>SetUserProperty Method</td>
<td>Sets the value of the specified User Property.</td>
<td>var myBusComp; myBusComp.SetUserProperty(propertyName, newValue);</td>
</tr>
<tr>
<td>SetViewMode Method</td>
<td>Sets the visibility type for the business component.</td>
<td>var myBusComp; myBusComp.SetViewMode(viewMode);</td>
</tr>
<tr>
<td>UndoRecord Method</td>
<td>Reverses any uncommitted changes made to the record.</td>
<td>var myBusComp; myBusComp.UndoRecord();</td>
</tr>
<tr>
<td>WriteRecord Method</td>
<td>Commits to the database any changes made to the current record.</td>
<td>var myBusComp; myBusComp.WriteRecord();</td>
</tr>
</tbody>
</table>

Table 76 lists a summary of the Business Component Events syntax.

### Table 76. Business Component Events Syntax Summary

<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>
</tbody>
</table>
### Business Component Events Syntax Summary

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<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>
<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(FieldNa__me, &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(FieldNa__me, 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>
</tbody>
</table>
Table 76. Business Component Events Syntax Summary

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<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 77 lists a summary of the Business Object methods’ syntax.

Table 77. 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. | var myBusObject;  
|                  |                      | var myBusComp;  
|                  |                      | myBusComp = myBusObject.GetBusComp(BusCompName);                      |
| Name Method     | Returns the name of the business object. | var myBusObject as BusObject;  
|                  |                      | var sName;  
|                  |                      | sName = myBusObject.Name();                                           |
### Table 78. Business Service Methods Syntax Summary

<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 oService; var sName; sName = oService.GetFirstProperty();</code></td>
</tr>
<tr>
<td>GetNextProperty</td>
<td>Once 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</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</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</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>
<tr>
<td>RemoveProperty</td>
<td>Removes a property from a business service.</td>
<td><code>var oService; oService.RemoveProperty(propName);</code></td>
</tr>
<tr>
<td>SetProperty</td>
<td>Assigns a value to a property of a business service.</td>
<td><code>var oService; oService.SetProperty(propName, propValue);</code></td>
</tr>
</tbody>
</table>
Table 79 lists a summary of the Business Service Events syntax.

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 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>

PropertySet Methods for eScript

Table 80 lists a summary of the PropertySet methods’ syntax.

Table 80. PropertySet 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>var oPropSet;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var iIndex;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>iIndex = oPropSet.AddChild(childObject);</td>
</tr>
<tr>
<td>Copy Method</td>
<td>Returns a copy of a property set.</td>
<td>var oPropSet1;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var oPropSet2;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet2 = oPropSet1.Copy();</td>
</tr>
<tr>
<td>GetChild Method</td>
<td>Returns a specified child property set of a property set.</td>
<td>var oPropSet;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var sPropVal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sPropVal = 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;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var iCount;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>iCount = oPropSet.GetChildCount();</td>
</tr>
</tbody>
</table>

Siebel Object Interfaces Reference Version 7.7, Rev. C  415
Table 80. PropertySet 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>var oPropSet;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var sPropName;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sPropName = 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;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var sPropName;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sPropName = 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;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var sPropVal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sPropVal = 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;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>count = 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;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var sTypeVal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sTypeVal = 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;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>var sValVal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sValVal = 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;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet.InsertChildAt(childObject, index);</td>
</tr>
<tr>
<td>PropertyExists Method</td>
<td>Returns a Boolean value indicating whether the property specified in the argument exists.</td>
<td>Dim oService as SiebelService</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dim propExists as Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>propExists = oService.PropertyExists(propName as String)</td>
</tr>
<tr>
<td>RemoveChild Method</td>
<td>Removes a child property set as a specified index from a parent property set.</td>
<td>var oPropSet;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet.RemoveChild(index);</td>
</tr>
<tr>
<td>RemoveProperty Method</td>
<td>Removes the property specified in its argument from a property set.</td>
<td>var oPropSet;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet.RemoveProperty(propName);</td>
</tr>
<tr>
<td>Reset Method</td>
<td>Removes every property and child property set from a property set.</td>
<td>var oPropSet;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet.Reset();</td>
</tr>
<tr>
<td>SetProperty Method</td>
<td>Assigns a value to the property of a property set specified in its argument.</td>
<td>var oPropSet;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oPropSet SetProperty(propName, propValue);</td>
</tr>
</tbody>
</table>
Table 81 lists a summary of the Miscellaneous Method syntax.

<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>var oPropSet; oPropSet.SetType(value);</td>
</tr>
<tr>
<td>SetValue Method</td>
<td>Assigns a data value to a value member of a property set.</td>
<td>var oPropSet; oPropSet.SetValue(value);</td>
</tr>
</tbody>
</table>

**Miscellaneous Methods for eScript**

Table 81 lists a summary of the Miscellaneous Method syntax.

<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>
Use the following procedure to invoke a custom method with a MiniButton.

**Invoking Custom Methods**

Be sure to set up Tools for the appropriate Target Browser Group.

**To invoke a custom method with a MiniButton**

1. Choose an applet (for example, Account List Applet) and create a control with the following properties.
   
   Name = ButtonTest  
   Caption = Test  
   HTML Type = MiniButton  
   Method Invoked = MyTest

2. Right click the Applet and choose Edit Web Layout.
   The Web layout editor appears.

3. Change the template mode on the Web Controls toolbar to 3: Edit List.
   A window opens with the available controls, including the one you just created.

4. Drag and drop the ButtonTest control onto an available location. When you release the mouse button, the button appears.

5. Click Save and then choose File > Close.

6. To add a server script to the applet that enables the button, right-click the applet and choose Edit Server Scripts. Add the following script to the WebApplet_PreCanInvokeMethod() function.

```
function WebApplet_PreCanInvokeMethod(MethodName, &CanInvoke) {
    if (MethodName == "MyTest") {
        CanInvoke = 'TRUE';
        return( CancelOperation );
    }
    return (ContinueOperation);
}
```

7. Add the following browser script to the applet you are using (for example, the Account List Applet).

```
function Applet_PreInvokeMethod(name, inputPropSet) {
    switch (name) {
```
Invoking Custom Methods with MiniButtons  ■  Invoking Custom Methods

```javascript
    case "MyTest":
        alert( "Siebel 7 browser script!" );
        return("Cancel Operation");
        break;
    }
    return ("ContinueOperation");
}
```

8 Run any application that has access to accounts, and go to the Accounts screen. The new button should appear.

9 Click Test.

    The Browser Script should display an alert box indicating "Siebel 7 Browser Script!"
Index

A
ActivateField business component method, about 180
ActivateMultipleFields business component method, about 182
ActiveApplet application method, about 120
ActiveBusObject application method, about 122
ActiveMode applet method, about 97
ActiveViewName application method, about returning name of active view 124
ActiveX control, about using Login method 145
AddChild property set method, about 296, 297
allocations, about using TraceOn application method to track 169
applet
ActiveApplet, about returning reference to currently focused applet 120
object interface events, table of 94
applet events
Applet_ChangeFieldValue, about 104
Applet_ChangeRecord event, about 105
Applet_InvokeMethod, about 106
Applet_Load, about 107
Applet_PreInvokeMethod, about 109
WebApplet_InvokeMethod, about 110
WebApplet_Load applet event 111
WebApplet_Load, about 108
WebApplet_PreCanInvokeMethod, about 112
WebApplet_PreInvokeMethod, about 113
WebApplet_ShowControl 115
WebApplet_ShowListColumn, about 116
applet methods
ActiveMode, about 97
BusComp, about 98
BusObject, about 98
Find control, about 100
FindActiveXControl, about 99
InvokeMethod, about 101
Name, about 102
syntax summary (Browser Script), table of 390
syntax summary (eScript), table of 403
Applet_ChangeFieldValue event, about 104
Applet_ChangeRecord event, about 105
Applet_InvokeMethod event, about 106
Applet_Load, about 107
Applet_PreInvokeMethod event, about 109
applets
applet methods syntax summary (Browser Script), table of 390
applet methods syntax summary (eScript), table of 403
applet methods syntax summary (Siebel VB), table of 373
Browser or Server script, adding to applet 47
events, about and list of 81
FindApplet, about returning applet identified by argument 131
object type, described 47
parent applet object, about returning for control 287
Webapplet events summary (Siebel VB), table of 374
WebApplet events summary, table of (eScript), table of 404
WebApplet events syntax summary (Browser Script), table of 390
application events
Application_Close event, about 173
Application_InvokeMethod, about 173
Application_Navigate, about 174
Application_PreNavigate, about 176
Application_Start, about 177
PreInvokeMethod, about 174
syntax summary, table of (eScript) 407
application events, about and list of 82
application extensions, running with 22
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
GetLastErrorCode, about 134
GetLastErrText, about 135
GetProfileAttr, about 135
GetService, about 136
GetSharedGlobal, about 138
GotoView, about 140
InvokeMethod, about 142
LoadObjects, about 143
LoadUserAttributes, about using to load user profile 144
Login. about 145
LoginID, about 147
LoginName, about 148
Logoff, about 149
LookupMessage, about 150
Name, about 151
NewPropertySet, about 152
PositionID, about 154
PositionName, about 154
RaiseError, about 156
RaiseErrorText, about 157
SetPositionID, about 159
SetPositionName, about 160
SetProfileAttr, about 160
SetSharedGlobal, about 162
syntax summary (COM data control), table 323
syntax summary (COM data server), table 335
syntax summary (eScript), table of 405
syntax summary, table of (eScript) 405
Trace, about 166
TraceOff, about 168
TraceOn, about 169
application object type
described 45
unique object type, about using to return 312
application, table of object interface events 95
Application_Close event, about 173
Application_InvokeMethod application event, about 173
Application_Navigate application event, about 174
Application_PreNavigate application event, about 176
Application_Start application event, about 177
applications
application events syntax summary (eScript), table of 407
application methods summary (Siebel VB), table of 375
application methods syntax summary (COM data control), table 323
application methods syntax summary (COM data server), table 335
application methods syntax summary (eScript), table of 405
application methods syntax summary (mobile/dedicated Web client), table 347
events summary (Siebel VB), table of 377
methods syntax summary (Browser Script), table of 391
association business component
Associate, about creating many-to-many relationship 183
BusComp_Associate, about calling after record added to create association 250
GetAssocBusComp, returning association business component 197
Attach application method, about 125
B
Basic program, verifying syntax 22
breakpoint, about and setting when debugging 29
Browser Script
See also Siebel Script Editor about 20
about Browser Script 389
applet methods syntax summary, table 390
application methods syntax summary, table 391
business component methods syntax summary, table 393
business object methods syntax summary, table 394
business service events syntax summary, table 396
business service methods syntax summary, table 395
Control methods syntax summary, table 398
PropertySet methods syntax summary, table 396
WebApplet events syntax summary, table 390
Browser, adding to applet 47
BusComp
applet method, about 98
control method, about 288
ExecuteQuery, about return record using method 191
ExecuteQuery2, about returning records using method 193
object interface events, table of 95
Index

BusComp_Associate business component event, about 250
BusComp_ChangeRecord business component event, about 251
BusComp_CopyRecord business component event, about 252
BusComp_DeleteRecord business component event, about 253
BusComp_InvokeMethod business component event, about 254
BusComp_NewRecord business component event, about 254
BusComp_PreAssociate business component event, about 255
BusComp_PreCopyRecord business component event, about 255
BusComp_PreDeleteRecord business component event, about 256
BusComp_PreGetFieldValue business component event, about 257
BusComp_PreInvokeMethod business component event, about 258
BusComp_PreNewRecord business component event, about 259
BusComp_PreQuery business component event, about 259
BusComp_PreSetFieldValue business component event, about 260
BusComp_PreWriteRecord business component event, about 262
BusComp_Query business component event, about 263
BusComp_SetFieldValue business component event, about 265
BusComp_WriteRecord business component event, about 265

business component methods
ActivateField, about 180
ActivateMultipleFields, about 182
Associate, about 183
BusObject, about 185
ClearToQuery, about 186
DeactivateFields, about 188
DeleteRecord, about 190
ExecuteQuery, about 191
ExecuteQuery2, about 193
FirstRecord, about 193
GetAssocBusComp, about 197
GetFieldValue, about 199
GetFormattedFieldValue, about 201
GetLastErrCode, about 202
GetLastErrText, about 203
GetMultipleFieldValues, about 204
GetMVGBusComp, about 204
GetNamedSearch, about 206
GetPicklistBusComp, about 206
GetSearchExpr, about 208
GetSearchSpec, about 209
GetUserProperty, about 210
GetViewMode, about 211
InvokeMethod, about 212
LastRecord, about 218
Name, about 218
NewRecord, about 219
NextRecord, about 220
NextSelected, about 221
ParentBusComp, about 222
Pick, about 223
PreviousRecord, about 224
RefineQuery, about 225
Release, about 226
SetFieldValue, about 228
SetFormattedFieldValue, about 230
SetMultipleFieldValues, about 232
SetNamedSearch, about 233
SetSearchExpr, about 235
SetSearchSpec, about 237

BusComp NewRecod, about 254
BusComp_PreAssociate, about 255
BusComp_PreDeleteRecord, about 256
BusComp_PreGetFieldValue, about 257
BusComp_PreInvokeMethod, about 258
BusComp_PreNewRecord, about 259
BusComp_PreQuery, about 259
BusComp_PreSetFieldValue, about 260
BusComp_PreWriteRecord, about 262
BusComp_Query, about 263
BusComp_SetFieldValue, about 265
BusComp_WriteRecord, about 265

syntax summary, table of (eScript) 411

business component
applet, associated with 98
BusComp method, about returning for the control 288
GetBusComp, about returning for a business component 266
methods syntax summary, table of (eScript) 407
name property, returning 218

business component events
BusCom_PreCopyRecord, about 255
BusComp_Associate, about 250
BusComp_ChangeRecord, about 251
BusComp_CopyRecord, about 252
BusComp_DeleteRecord, about 253
BusComp_InvokeMethod, about 254
Index ■ B

SetSortSpec, about 241
SetUserProperty, about 243
SetViewMode, about 245
syntax summary (COM data control),
table 326
syntax summary (COM data server),
table 338
UndoRecord, about 248
WriteRecord, about 248

business components
about  66
BusComp object, logical flow of
instantiating  67
business component events summary (Siebel
VB), table of 382
business component events syntax summary
(eScript), table of 411
business component methods syntax summary (COM data control),
table 326
business component methods syntax summary (COM data server),
table 338
business component methods syntax summary (eScript), table of 407
business component methods syntax summary (Siebel VB), table of 378
business rules, adding to 21
database, committing records to 66
definitions, adding and inserting 66
definitions, scenarios 66
SiebelBusComp methods syntax summary (Java), table of 365

business object methods
GetBusComp, about 266
GetLastErrCode, about 267
GetLastErrText, about 268
Name, about 269
Release, about 269
syntax summary (COM data control),
table 330
syntax summary (COM data server),
table 342
table of 91

business objects
active applet, about returning for business component 122
business object methods syntax summary
(COM data control), table 330
business object methods syntax summary
(COM data server), table 342
business object methods syntax summary
(eScript), table of 413
business object methods syntax summary
(Siebel VB), table of 384
BusObject, about returning business object
for applet 98
BusObject, about returning business object
that contains business component 185
methods syntax summary (Browser Script),
table of 394
methods syntax summary (mobile/dedicated
Web client), table 354
Name, about using to return name of business
object 269
object type, described 46

business rules
business component, adding to 21
described 20

business service
object interface events, table of 96
object interface methods, table of 92

business service events
Service_InvokeMethod, about 282
Service_PreCanInvokeMethod, about 284
Service_PreInvokeMethod, about 284
syntax summary, table of (eScript) 415

business service methods
GetFirstProperty, about 271
GetLastErrCode, about 272
GetLastErrText, about 273
GetNextProperty, about 274
GetProperty, about 275
InvokeMethod, about 276
Name, about 277
PropertyExists, about 278
Release, about 278
RemoveProperty, about 280
SetProperty, about 281
syntax summary (COM data control),
table 330, 331
syntax summary (COM data server),
table 343
syntax summary, table of (eScript) 414

business services
business service events syntax summary
(eScript), table of 415
business service events syntax summary
(Siebel VB), table of 385
business service methods syntax summary
(COM data control), table 330, 331
business service methods syntax summary (COM data server), table 343
business service methods syntax summary (eScript), table of 414
business service methods syntax summary (Siebel VB), table of 384
events syntax summary (Browser Script), table of 396
methods syntax summary (Browser Script), table of 395
methods syntax summary (mobile/dedicated Web client), table of 386
retrieving property names 274
setProperty, about assigning values to members of 280
SiebelService methods syntax summary (Java), table of 369

business services object type, described 46
BusObject
  applet method, about 98
  business component method, about 185

C
C++
  Siebel COM Server, building in 315
  Siebel COM Server, testing program 321
Calls window, about and accessing 30
ChangeFieldValue, about 104
ChangeRecord event, about 105
ClearToQuery business component method, about 186
coding, caution, about and using Siebel Tools 19
COM data control
  application methods syntax summary (table) 323
  business component methods syntax summary (table) 326
  business object methods syntax summary (table) 330
  business service methods syntax summary (table) 331
  installation, about 40
  property set methods syntax summary (table) 331
COM data control, load balancing with 84
COM data server
  application methods syntax summary (table) 335
  business component methods syntax summary (table) 338
  business object methods syntax summary (table) 342
business service methods syntax summary (table) 343
installation, about 40
interface method, about COM error handling 85
LoadObjects method, about using to start object and return reference 143
property set methods syntax summary (table) 344
COM error handling, about and methods 85
COM interfaces
  Siebel COM client in C++, building 315
  Siebel COM client in C++, testing program 321
comparison operators, using in search expressions 238
connect string
  about, syntax, and example 82
  leveraging load balancing with 84
  Siebel Server, substitutions when logging into (table) 83
constants, table of 96
count
  FindControl, about argument specified in 100
  GetValue, about returning value of count 289
object interface methods, table of 92
SetValue, about using to set the contents of the control 293
count methods
  Applet method, about returning parent applet object 287
  BusComp, about 288
  GetProperty, about 288
  GetValue, about returning control value 289
  Name, about returning object name 290
  SetProperty, about 290, 292
  SetValue, about using to set contents of the control 293
  syntax summary, table of (Browser Script), table of 398
counts
  GetProperty, assigning values to properties 288
  SetProperty, assigning values to properties 290, 292
Copy property set method, about 297
copying records, using NewRecord method 220
CurrencyCode application method, about 127
custom method, invoking with a MiniButton 419
D

data bean, table of SiebelDataBean methods syntax summary (Java), table of 363
data value
  SetProperty, about using to assign value to 308
  SetType, about using to assign data value of type to property set 309
database, about using WriteRecord to commit to database 248
DeactivateFields business component method, about 188
deallocations, using TraceOn application method to track 169
debug tracings methods, table of 74
debugger. See Siebel Debugger
DeleteRecord business component method, about 190
Detach application method, about 128

Document Object Model (DOM). See Browser Script

E

EnableExceptions application method, about 129
error code
  application methods, about using GetLastErrCode to return last error code 134
  business component methods, about using GetLastErrCode to return most recent 202
  business object methods, about using GetLastErrCode to return most recent 267
  business service methods, about using GetLastErrCode to return most recent 272
  GetErrorInfo, about using with Java Data Bean to display numeric code 311
error handling
  See also individual Siebel object interface entries
  COM error handling, about and examples 85
  error message tracking 86
  native COM error handling, enabling and disabling 129
error messages
  function_name Is An Unknown Function, about and correcting 32
  GetErrorMessage, about using with Java Data Bean to display message 312
  GetLastErrText, about returning last text error message 135

error text messages
  business component methods, about using GetLastErrText 203
  business object methods, about using GetLastErrText 268
  business service methods, about using GetLastErrText 273
event method syntax 74
events, object interface events, table of 94
ExecuteQuery business component method, about 191
ExecuteQuery2 business component method, about 193
exposed object types, table of 48
external applications
  logging in 145

F

field value, method of retuning in the current local format 201
FindActiveXControl applet method, about 99
FindApplet application method, about 131
FindControl applet method, about 100
FirstRecord business component method, about 193
FirstSelected business component method 196

G

GetAssocBusComp business component method, about 197
GetBusComp business object method, about 266
GetChild property set method, about 298
GetChildCount property set method, about 300
GetErrorCode method, about 311
GetErrorMessage method, about using to display error messages 312
GetFieldId business component method, about 199
GetFirstProperty
  business service methods, about 271
property set methods, about 300
GetFormattedFieldValue business component method, about 201
GetLastErrCode
  application methods, about 134
  business component methods, about 202
  business object methods, about 267
  business service methods, about 272
GetLastErrorText
  application methods, about 135
  business object methods, about 268
  business service methods, about 273
  note, about availability to interfaces 33
GetLastErrorText business component method, about 203
GetMultipleFieldValues business component method, about 204
GetMVGBusComp business component method, about 204
GetNamedSearch business component method, about 206
GetNextProperty
  business service methods, about 274
  property set methods, about 301
GetPicklistBusComp business component method, about 206
GetProfileAttr application method, about 135
GetProperty
  business service methods, about 275
  control methods, about 288
  property set methods, about 302
GetPropertyCount property set method, about 303
GetSearchExpr business component method, about 208
GetSearchSpec business component method, about 209
GetService application method, about 136
GetSharedGlobal application method, about 138
GetType property set method 303
GetUserProperty business component method, about 210
GetValue
  control methods, about 289
  property set methods, about 304
GetViewMode business component method, about 211
global state, properties and functions 71
global variables
  about and VB example 73
  GetSharedGlobal application method, about 138
GotoView application method, about 140

H
  high interactivity mode, about running Browser scripts 389
I
  InsertChildAt property set method, about 305
  installation procedures, object interfaces 40
  inter-application variable methods, table of 74
  interface methods, table grouped by object interface type 87
InvokeMethod
  applet methods, about 101
  Applet_InvokeMethod, about 106
  application methods, about 142
  business component methods, about 212
  business service methods, about 276
  WeApplet_InvokeMethod, about 110

J
  java Bean. See individual Siebel Java entries
Java Cryptography Extension (JCE), enabling 64
Java Data Bean
  GetErrorCode, about using to display numeric error codes 311
  GetErrorMessage, about using to display error messages 312
Java Data Beans, load balancing with 84
JavaScript. See Siebel eScript
JCE (Java Cryptography Extension), enabling 64

L
  LastRecord business component method, about 218
  load balancing 84
Load event
  Applet_Load, about triggering after applet is loaded 107
  WebApplet_Load event, about triggering just after applet is loaded 111
LoadObjects application method, about 143
LoadUserAttributes application method, about 144
local variables, described and VB example 72
locating objects method, about and list of methods 65
logical operators in search expressions 238
Login method application method, about 145
LoginId application method, about 147
LoginName application method, about 148
Logoff application method, about 149
LookupMessage application method, about 150

M
methods
custom method, invoking with a MiniButton 419
table grouped by interface type 87
Microsoft Foundation Class (MFC) library.
See Siebel COM Data Server
Microsoft Visual Basic
Siebel COM Data Control Interface, setting up to access 57
Siebel COM Data Server, setting up to access 55
Siebel Mobile/Dedicated Web Client Automation Server, setting up to access 53
Siebel Web Client Automation Server, setting up to access 52
MiniButton, using to invoke custom method 419
mobile Web client automation server, about installation 40
mobile/dedicated Web client
application methods syntax summary, table of 347
business component methods syntax summary, table of 350
business object methods syntax summary, table of 354
business service methods syntax summary, table of 355
property set methods syntax summary, table of 356
module variables, about and VB example 72
MVG business component, returning 204

O
object interface events
applet, table of 94
application, table of 95
BusComp, table of 95
business service, table of 96
object interface methods tables
applet, table of 87
application, table of 88
business component, table of 90
business object, table of 91
business service, table of 92
control, table of 92
miscellaneous methods and events, table of 93
property set, table of 93
object interfaces
component of Siebel programming environment described 20
object types
applet object type, described 47
application, described 45
business component, described 46
business object, described 46
business service, described 46
property set, described 47
Script flag, about 24
Siebel Object Interface, object types, table of 48
object, about using Name method to return object name 290
operating currency code, returning 127

P
ParentBusComp business component method, about 222
Pick business component method
GetPicklistBusComp, returns component 206
Pick method, about 223
PositionId application method, about 154
PositionName application method, about 154
PreCanInvokeMethod, about
WebApplet_PreCanInvokeMethod 12
preferences
directing preferences, accessing and settings 27
run-time preferences, accessing and settings 28
Siebel Script Editor, accessing and window features 25
PreInvokeMethod
Applet_PreInvokeMethod, about 109
Application_PreInvokeMethod, about 174
WebApplet_PreInvokeMethod, about 113
PreviousRecord business component method, about 224
programming
custom extension routines, about extending data validation 20
environment, component of 19
languages, about 19
user interface components, about customizing behavior 21
properties of controls
GetProperty, about assigning 288
SetProperty, about assigning visual properties 290, 292
property set methods
AddChild, about adding subsidiary property set to 296
Copy, about returning copy of set 297
GetChild, about returning child property of property set 298
GetChildCount, about returning child property sets attached to 300
GetFirstProperty, about returning name of first property 300
GetNextProperty, about returning next property 301
GetProperty, about returning property value when given name 302
GetPropertyCount, about returning number of properties attached to 303
GetValue, about retrieving data value 304
InsertChildAt, about inserting child property set into parent property 305
object interface methods, table of 93
RemoveChild, about removing child property set from parent property set 306
RemoveProperty, about removing a property from property set 307
SetProperty, about assigning a data value to property 308
SetType, about assigning data value of type 309
syntax summary (COM data control), table 331
syntax summary (COM data server), table 344
syntax summary table (eScript) 415
property set object type, described 47
property sets
business service methods syntax summary (COM data control), table 331
business service methods syntax summary (COM data server), table 344
Copy, about returning copy of 297
GetChild, about retrieving child property set 298
GetFirstProperty, about retrieving property names 300
GetNextProperty, about retrieving property names 301
GetProperty, about retrieving property values 302
GetPropertyCount, about retrieving values of type members 303
GetType, about retrieving values of type members 303
GetValue, about retrieving value values 304
InsertChildAt, about adding subsidiary 305
methods syntax summary (mobile/dedicated Web client), table 356
property set methods syntax summary (eScript), table of 415
property set methods syntax summary (Siebel VB), table of 386
RemoveChild, about removing child property set 306
RemoveProperty, about removing properties of 307
Reset, about removing properties and child properties 307
SetProperty, about assigning values to members of 308
SetType, about assigning values to type members 309
SetValue, about assigning values to value member 310
SiebelPropertySet methods syntax summary (Java), table of 370
tree-structured data structures, for 296
PropertyExists
business service method, about 278
property set method, about returning Boolean value 305
PropertySet
methods syntax summary (Browser Script), table of 396
methods syntax summary (Siebel Web client), table of 360
queries
  ClearToQuery, about using to clear query 186
  RefineQuery, about using to define after execution 225
  SetSortSpec, about using to set sort specification 241
quotation marks, about using in search expressions 238
R
  RaiseError application method, about 156
  RaiseErrorText application method, about 157
records
  LastRecord, about using to move to 218
  NewRecord, about adding a new record (row) 219
  NextSelected, about using to move focus to next record 221
  Pick, about placing record in a picklist 223
  PreviousRecord, about moving to previous record 224
  UndoRecord, about using to reverse uncommitted changes 248
  WriteRecord, about committing database changes 248
  RefineQuery business component method, about 225
Release
  business component method, about 226
  business object method, about 269
  business service method, about 278
RemoveChild property set method, about 306
RemoveProperty
  business service method, about 280
  property set method, about 307
Reset property set method, about removing properties and child property sets 307
Run-time Engine, invoking 32
S
scripts
  syntax, checking 29
search expression
  GetSearchExpr, about using to return current search expression 208
  SetSearchExpr, about setting on entire search expression 235
search specification
  Field name argument, about returning for field specified in 209
  searchName, returns named search specification 206
  SetNamedSearch, about setting a named search specification on the business component 233
  SetSearchSpec, about setting for a particular field 237
  SetSearchSpec, about setting for particular field 237
Server Script, components 19
  server, about Logoff method 149
Service_InvokeMethod business service event, about 282
Service_PreCanInvokeMethod business service event, about 284
Service_PreInvokeMethod business service event, about 284
SetFieldVal business component method, about 228
SetFormattedFieldValue business component method, about 230
SetMultipleFieldValues business component method, about 232
SetNamedSearch business component method, about 233
SetPositionID application method, about 159
SetProfileAttr application method, about 160
SetProperty
  business service method, about assigning 281
  control property, about returning value of 288
  control, about setting visual properties 290, 292
  property set method, about assigning data value to 308
SetSearchExpr business component method, about 235
SetSearchSpec business component method, about 237
SetSearchSpec Method business component method 237
SetSharedGlobal application method, about 162
SetSortSpec business component method, about 241
SetType property set method, about 309
SetUserProperty business component...
method, about 243
SetValue control, about using to set contents of 293
property set, about assigning data value to 310
SetViewMode business component method 245
SetViewMode business component method, about 245
ShowModalDialog business component method 163
ShowModalDialog Method 163
Siebel business components, about events and list of 79
Siebel COM Data Control
about and diagram 41
instantiating 57
Siebel COM Data Server
about and diagram 42
building in C++ 315
C++, testing program 321
instantiating 55
Siebel COM interfaces
accessing 51
COM Data Control interfaces, about and diagram 41
COM Data Server, about and diagram 42
COM error handling 85
Siebel Mobile/Dedicated Web Client Automation Server, about and diagram 43
Siebel Web Client Automation Server, about and diagram 42
Siebel Compiler
compiler/interpreter described 20
invoking 32
order considerations and error message 32
Siebel constants table 96
Siebel Debugger
about using 25
accessing 26
breakpoint, about and setting when debugging 29
Debug toolbar, diagram and description of 26
debugging preferences, accessing and settings 27
run-time preferences, accessing and settings 28
script syntax, checking 29
subroutines and function calls, displaying 30
variable window, about 30
Siebel Dedicated Automation Server
installation, about 40
Siebel eScript
See also Siebel Script Editor about 19
applet methods, syntax summary (table) 403
application events syntax summary, table of 407
application methods syntax summary, table of 405
business component events syntax summary, table of 411
business component methods syntax summary, table of 405
business object methods syntax summary, table of 415
business service events syntax summary, table of 415
business service methods syntax summary, table of 414
objects, destroying and example 37
property set methods syntax summary, table of 415
script syntax, checking 29
Siebel VB, differences between 35
Switch construct, making effective use of 37
syntax conventions 50
syntax, verifying 22
theApplication, method syntax summary, table of 417
this object reference, about using and example 36
variables, declaring 36
WebApplet event summary, table of 404
with shortcut, about and example 36
Siebel eScript language, about 19
Siebel extension events
applet events, about and list of 81
applications events, about and list of 82
events occur, determining when 79
method syntax 74
program flow, process affected by script 75
Siebel business component events, about and list of 79
Siebel interface objects
Siebel methods and events, about accessing from scripts 44
usage evaluation matrix, table 44
Siebel Java Bean
codepage support (table) 62
data Bean, about installation 40
JDB and Siebel Server, encrypting communication between 63
SiebelBusComp methods syntax summary, table of 365
Siebel DataBean methods syntax summary, table of 363
Siebel Exceptions methods syntax summary, table of 371
Siebel PropertySet methods syntax summary, table of 370
Siebel Service methods syntax summary, table of 369

Siebel Java interfaces
multiple threads, using with 44
object, about using to access 43

Siebel Mobile/Dedicated Web Client Automation Server
about and diagram 43
accessing 53

Siebel object interface
See also error handling
interface installations, about 40
Java Data Bean 59
Siebel COM Data Control, instantiating 57
Siebel COM Data Server, instantiating 55
Siebel COM interfaces, accessing method 41
Siebel Java interfaces 43

Siebel Object Interface method
devices examples 50
syntax 48

Siebel object interface, events
See also individual Siebel object interface entries
applet events, about and list of 81
application events, about and list of 82
events occur, determining when 79
method syntax 74
program flow, process affected by script 75
Siebel business component events, about and list of 79

Siebel object interface, getting started
See also individual Siebel object interface entries
connect string, about, syntax, and example 82
connect string, substitutions when logging into a Siebel Server (table) 83
Siebel COM Data Control, accessing and screen example 57
Siebel COM interfaces, accessing 51
Siebel mobile/dedicated Web client automation server, accessing 53
Siebel Web Client Automation Server, accessing 51

Siebel object interface, methods
See also individual Siebel object interface entries
business components, accessing 66
global state properties and functions 71
list of 64
locating objects, about and list of methods 65
navigation methods 70
user interaction, about and methods 71

Siebel Object Model. See Browser Script
Siebel programming
constants, table of 96
custom extension routines, about extending data validation 20
environment, components of 19
user interface components, about customizing behavior 21

Siebel script
debug tracing methods, table of 74
global variables, about and VB example 73
inter-application communication methods, list of 74
local variables, about and VB example 72
module variables, about and VB example 72

Siebel Script Editor
about 20
accessing and screen example 23
custom programs, tips about creating 22
menu options, list of 23, 24
preferences, accessing and window features 25
Scripted flag, about 24

Siebel Server
applet, adding to 47
JDB and Siebel Server, encrypting between 63

Siebel session ID, about returning string containing Id 128

Siebel VB
See also Siebel Script Editor
about 19
applet methods syntax summary, table of 373
application events summary, table of 377
application methods syntax summary, table of 375
business component methods syntax summary, table of 378
business components events summary, table of 382
business object methods syntax summary, table of 384
business service events syntax summary, table of 385
business service methods syntax summary, table of 384
components of 19
getting started 32
picklist, picking a value from 207
property set methods syntax summary, table of 386
script syntax, checking 29
Siebel eScript, differences between 35
syntax conventions 49
theApplication method, syntax summary 388
Webapplet events, summary (table) 374
Siebel VB language, about 19
Siebel VB, getting started
date variables, about working with 34
Me object reference, about using and example 33
naming conventions, about using standardized 32
objects, destroying and example 35
run-time errors, about trapping 33
Select Case, making effective use of 33
variables, declaring 32
With shortcut, using and example 34
Siebel Web client
PropertySet methods syntax summary, table of 360
Siebel Service methods syntax summary, table of 360
SiebelHTMLApplication methods syntax summary, table of 359
Siebel Web Client Automation Server
about and diagram 42
accessing 51
installation, about 40
SiebelBusComp methods syntax summary (Java), table of 365
SiebelDataBean methods syntax summary (Java), table of 363
SiebelException methods
syntax summary (Java), table of 371
SiebelHTMLApplication methods syntax summary, table of 359
SiebelPropertySet methods syntax summary (Java), table of 370
SiebelService methods
syntax summary (Java), table of 369
syntax summary (Siebel Web client), table of 360
sorting specification, setting 241
special characters, using in search expressions 238
specialized methods, calling 212
subsidiary property sets, about using AddChild to add to a property set 296
syntax checking 28

T
theApplication method
object type, about using to return 312
syntax summary (eScript), table of 417
syntax summary (Siebel VB) 388
Trace application method, about 166
TraceOff application method
about 168
debug tracing, about 74
TraceOn application method
about 169
debug tracing, about 74
tree-structured data structures, creating using property sets 296

U
UndoRecord business component method, about 248
user interaction, object interface methods 71
user interface control object type 47
user property value
getUserProperty, about using to return value 210
SetUserProperty, about using to set the value of named business user property 243
user-created methods, calling 212

V
value, about returning value of control 289
variables, about 30
visibility mode, about returning current visibility mode 211
visibility type
SetViewMode, about setting for business component 245

W
Web Client Automation Server, enabling 42
WebApplet events
summary, table of (eScript) 404
syntax summary, table of (Browser Script) 390
WebApplet_InvokeMethod event, about 108
WebApplet_Load, about 111
WebApplet_PreInvokeMethod event, about 112, 113
WebApplet_ShowControl event, about 115
WebApplet_ShowListColumn, about 116
WriteRecord business component method,
Index

about 248