

SIEBEL[®] eBUSINESS APPLICATIONS

PRODUCT CONFIGURATOR INTEGRATION OBJECT API REFERENCE

SIEBEL 2000
VERSION 6.0

10PA1-CI01-06000

MARCH 2000

Siebel Systems, Inc., 1855 South Grant St., San Mateo, CA 94402
Copyright © 2000 Siebel Systems, Inc.
All rights reserved. Published 1997–2000
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.

Target Account Selling Methodologies, Copyright © 1996 Target Marketing International, Inc. All rights reserved.

The full text search capabilities of Siebel Enterprise Applications include technology used under license from Fulcrum Technologies, Inc. and are the copyright of Fulcrum Technologies, Inc. and/or its licensors.

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

Windows® is a registered trademark of Microsoft Corporation.

All other product names, marks, logos, and symbols may be trademarks or registered trademarks of their respective owners.

U.S. GOVERNMENT RESTRICTED RIGHTS. Programs, Ancillary Programs and Documentation, delivered subject to the Department of Defense Federal Acquisition Regulation Supplement, are “commercial computer software” as set forth in DFARS 227.7202, Commercial Computer Software and Commercial Computer Software Documentation, and as such, any use, duplication and disclosure of the Programs, Ancillary Programs and Documentation shall be subject to the restrictions contained in the applicable Siebel license agreement. All other use, duplication and disclosure of the Programs, Ancillary Programs and Documentation by the U.S. Government shall be subject to the applicable Siebel license agreement and the restrictions contained in subsection (c) of FAR 52.227-19, Commercial Computer Software - Restricted Rights (June 1987), or FAR 52.227-14, Rights in General Data Alternative III (June 1987), as applicable. Contractor/licensor is Siebel Systems, Inc., 1855 South Grant Street, San Mateo, CA 94402.

Proprietary Information

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

Contents

Introduction

Who Should Use This Guide	Intro-2
Assumptions	Intro-2
Additional Documentation	Intro-3
Contacting Siebel Technical Support	Intro-4
Siebel Welcomes Your Comments	Intro-5

Chapter 1. Overview

About This Chapter	1-2
Introduction to Product Configuration	1-2
Overview of the Information Exchange Process	1-5

Chapter 2. Getting Started

About This Chapter	2-2
Setting Initialization Parameters	2-4

Chapter 3. API Reference

About This Chapter	3-2
Obtaining a Reference to the Product Configurator Interface	3-2
Product Configurator Program Flow	3-3
Siebel Product Configurator Interface Methods	3-4
ConfigurationDone	3-5
GetModelName	3-6
GetSiebelContextInfo	3-7

GetSiebelInitialLineItem	3-9
GetSiebelInitialLineItemCount	3-11
IsVerifyMode	3-13
SetSiebelLineItem	3-14
Sample Program	3-18

Introduction

Who Should Use This Guide	Intro-2
Assumptions	Intro-2
Additional Documentation	Intro-3
Contacting Siebel Technical Support	Intro-4
Siebel Welcomes Your Comments	Intro-5

Who Should Use This Guide

This guide describes how to use the Siebel Product Configurator Interface methods to create a Productd Configurator Interface for exchanging information with a Siebel application.

The audience for this guide consists of:

Project Manager or Product Marketing Administrators Persons involved in managing sales force automation projects.

Siebel Application Developers Persons who plan, implement, and configure Siebel applications, possibly adding new functionality. A developer is typically someone from the Information Services department.

Configurators Persons responsible for planning, implementing, and configuring Siebel applications. A configurator is typically a consultant or someone from the Information Systems department.

Assumptions

This manual is a reference for the OLE2 Automation Integration between the Siebel application and a third-party Product Configurator application (or “Product Configurator”). It makes the following assumptions:

- You are familiar with the Siebel Enterprise applications.
- You know how to develop applications using Microsoft Visual Basic or other OLE-enabled development languages.
- You know how to use your third-party Product Configurator.

NOTE: The integration to Siebel is not confined to using Microsoft Visual Basic. You can use any language that supports OLE2 technology. Because of its simplicity and popularity, Microsoft Visual Basic is used in the examples in this manual.

Additional Documentation

The following documentation also provides information on the topics addressed in this guide.

Siebel Data Model Reference

Siebel Server Administration Guide

Siebel Tools Guide

Siebel VB Language Reference

For copies of these documents, please use Siebel Books Online, accessible via the Worldwide Services tab on the Siebel Systems Web site (www.siebel.com). Through Siebel Books Online, you can order additional Siebel documentation and copies of the *Siebel Bookshelf for Enterprise Applications* CD-ROM.

Another source of information is the *Siebel Online Help*.

Contacting Siebel Technical Support

Do you know how to access Siebel Technical Support? It is crucial that you understand the requirements for getting support. This will ensure the best experience possible. If you have questions, please don't hesitate to contact us.

To ensure that you maximize your knowledge of Siebel products and your return on investment:

- You must attend Siebel training to become a “designated contact.”
- Your trained designated contacts provide technical support to your users. Siebel Technical Support provides support to your “designated contacts” only.

To provide efficient and timely support, and to empower you in the process:

- Siebel Technical Support is primarily Web-based, accessed via Siebel SupportWeb (<http://supportweb.siebel.com>). Please submit new service requests to us through SupportWeb, where you can also search the knowledge base for solutions.
- Designated contacts receive read/write access to SupportWeb. All other project team members at your company receive a read-only account to ensure they can reap the benefits of the support knowledge base.

To register for Siebel training, please access <http://www.siebel.com/education/> and choose Siebel Customer Technical Education. Questions on the above can be directed to Siebel Technical Support at:

eBusiness Customers: support@siebel.com

Workgroup Customers: swasupport@siebel.com or:

Americas: eBusiness: 800.214.0400 or 650.295.5724 Workgroup: 800.354.1571

London: +44.1784.494.949 Tokyo: +81.3.5469.3811 (main number)

Munich: +49.89.95718.400 Singapore: +65.320.8533 (main number)

Please submit technical issues and updates to Siebel SupportWeb (<http://supportweb.siebel.com>). If you do not have a SupportWeb account, please email us at the relevant email address above. Thank you and we hope you enjoy working with Siebel Technical Support!

Siebel Welcomes Your Comments

To help us with future versions, we want to know about any corrections or clarifications that you would find useful. Please include in your message:

- The title and version of this guide
- Your name, your company name, job title or functional area, phone number, and e-mail address

Write to us via regular mail or email at:

Siebel Systems, Inc.
Technical Publications Department
1855 South Grant Street
San Mateo, CA 94402-2667

doc@siebel.com

We appreciate your feedback.

Introduction

Siebel Welcomes Your Comments

About This Chapter1-2

Introduction to Product Configuration1-2

Overview of the Information Exchange Process1-5

About This Chapter

This chapter provides an introduction to the Siebel Product Configurator Integration Object API. In addition, it contains an overview of the information exchange process that you will use to move data between your Siebel application and a 3rd-party configuration engine.

Introduction to Product Configuration

A Product Configurator is software that enables users to model the rules and relationships between products that they sell in a way that facilitates the validation of sales quotes. Product Configurators allow Product Marketing and Engineering departments to define the rules, requirements, and accessories for complex products and product suites. When a sales representative is creating a quotation for these complex products, the Product Configurator assists the representative by informing him or her of when items are not compatible and when certain selections are not available. The Product Configurator can also help the representative by ensuring that all of the required components are selected and present on the quote.

Each implementation of the Product Configurator is custom-developed. The user-interface component is usually developed by the Configurator vendor or the customer, typically in Visual Basic, PowerBuilder, or C++. Because of this custom implementation, Siebel created the Product Configurator Integration Object, a standard API and integration object that makes the methods of the object available through industry-standard technology. The object uses OLE2 to expose methods to external programs such as the custom VB applications.

Figure 1-1 summarizes the information flow among the various components in the product configuration process.

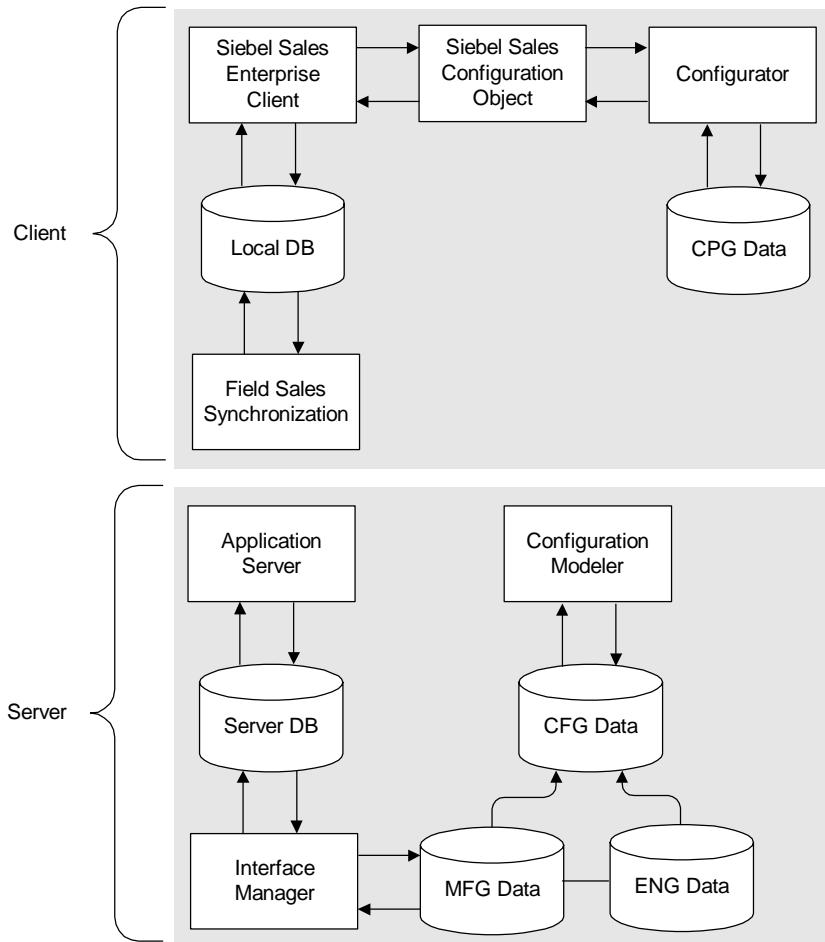


Figure 1-1. System Architecture—Client

The Siebel Product Configurator Integration Object is a software object that allows bidirectional exchange of quote information from Siebel quotes to a Product Configurator and back. The result is a Siebel quote that has been verified by the Product Configurator.

Users can click a button in the Siebel Quote Configuration view to cause the Siebel Product Configurator Integration Object to launch a third-party Product Configurator software program. The Siebel Product Configurator Integration Object then allows your configuration program to access information such as the quote number, the account name, the opportunity, and the model name to the third-party Configurator. The third-party Configurator displays its user interface, and the user makes the desired product selections, or *configuration*. When the configuration is complete, the user clicks a button indicating completion, and the Siebel Product Configurator Integration Object manages the receipt of product selection information back into the Siebel quote.

Overview of the Information Exchange Process

The information exchange process is described in the following steps. This procedure assumes that a quote has been created and that it contains a solution with line items. A *solution* is the name of a group of products configured together on the quote.

The Information Exchange Process

- 1 The user clicks either the Configure button or the Verify button on the Siebel Quote Configuration view.

The Quote Configuration view appears.

The screenshot shows the Siebel Sales - Quote Configuration window. The interface includes a menu bar (File, Edit, View, Screens, Go, Query, Reports, Help), a toolbar with various icons, and a main content area. The main content area is divided into several sections:

- Quote Section:** Contains fields for Quote # (1-3V5D), Account (Mach Systems), Last Name, Price List, Revision (1), Site (Northern California), First Name, and Discount.
- Solution Set Section:** Contains a table with columns: Solution, Probabl., Qty, Model Product. The data row shows: Quote 03/05/00, 1, High-End Server.
- Line Items Section:** Contains a table with columns: Line, Qty Req, Product, Current Discount, Next Discount, Upsell, Net Price, Write-In Prod.

The status bar at the bottom indicates "Item: 10 of 18".

NOTE: For complete information on the Siebel Quote Configuration view, see the online documentation.

- 2 The Siebel application launches the specified Configurator application and passes it certain pertinent information on the command line.
- 3 The Siebel application passes control to the Configurator application, blocking mouse and keyboard events until it receives the signal from the Configurator application that the configuration has completed.
- 4 The Configurator application creates a Configurator Interface Object and begins to query the Siebel application for both contextual information and detailed product information regarding each line item.
- 5 Using this information, the Configurator application can update its user interface and interact with the user. The diagram in [Figure 1-2](#) is a simplified illustration of this process.

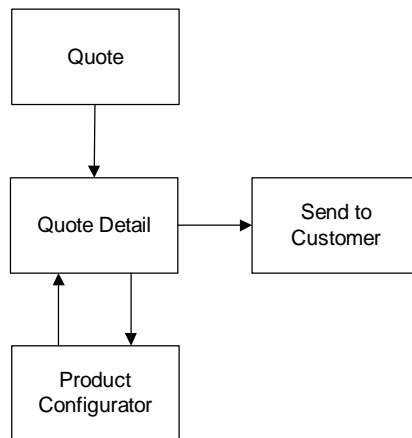


Figure 1-2. Process Flow

- 6 When the user clicks the OK button to indicate that his or her interaction with the Product Configurator is complete, the Product Configurator sends the information back to Siebel.

- 7** The Siebel application updates the quote line items, resulting in a quote that contains a line item for each product selected in the Product Configurator application.

The OLE2 Automation Interface that performs the integration between the Siebel application and the Configurator application is called “Siebel Product Configurator Interfaces,” and it is found in the file named `SPRODCFG.TLB`. This file is installed with the Siebel application and is in the same directory as the `SIEBEL.EXE` file, usually `C:\SIEBEL\BIN`. This interface is registered automatically each time the Siebel application is executed when you have the OLE Automation parameter set to `TRUE` in your `SIEBEL.CFG` file (`EnableOLEAutomation = TRUE`).

Overview

Overview of the Information Exchange Process

Getting Started

2

About This Chapter	2-2
Setting Initialization Parameters	2-4

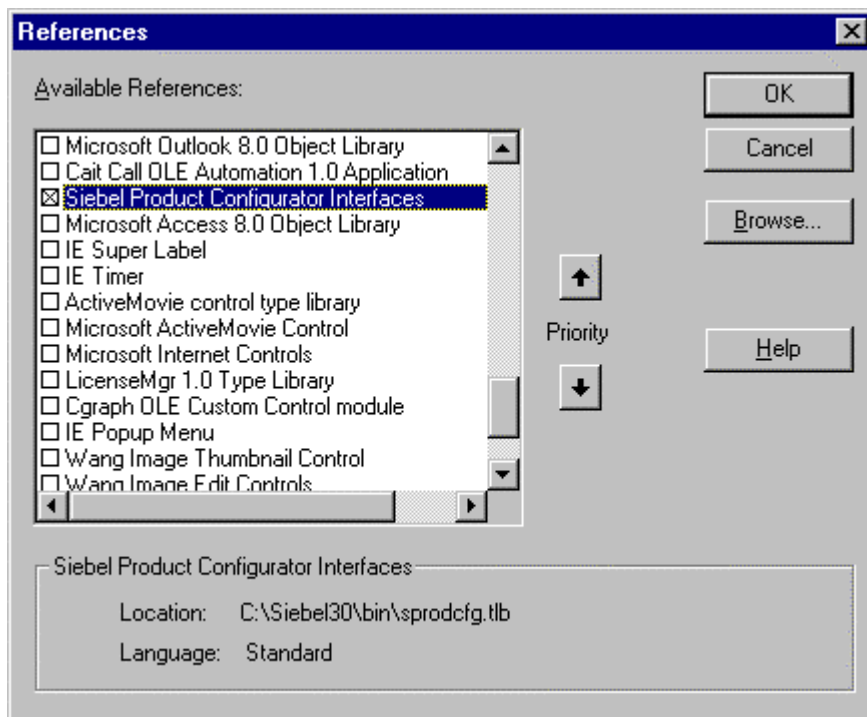
About This Chapter

This chapter describes how to enable the Siebel Product Configurator Interfaces and how to set initialization parameters.

To enable the Siebel Product Configurator Interfaces

- 1 In Visual Basic, choose References from the menu (in Visual Basic version 4.0, the menu name is Tools; in version 5.0, the menu name is Projects).

The References dialog box appears with a list of available references.

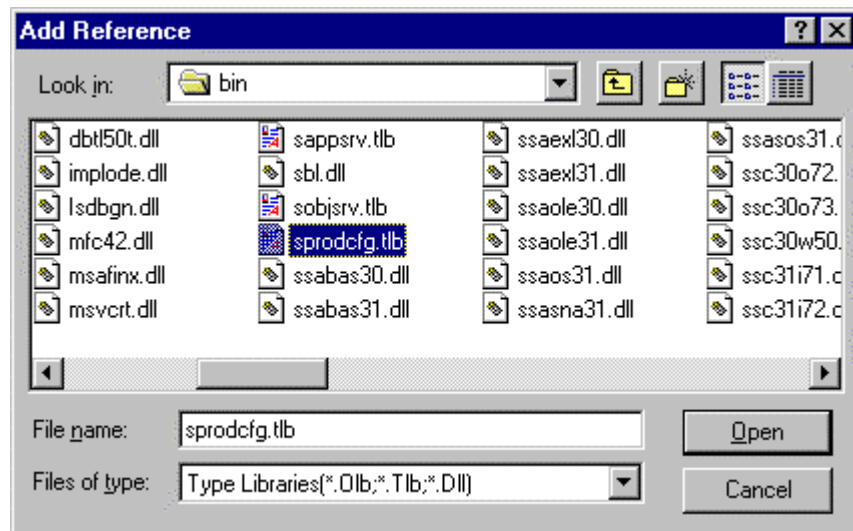


2 Choose Siebel Product Configurator Interfaces.

If Siebel Product Configurator Interfaces is not listed in the dialog box or is shown as “MISSING,” click the Browse button to locate and select SPRODCFG.TLB from the Siebel installation directory.

3 Click OK. You are finished enabling the Interfaces.**4** If you want to view arguments for the Interfaces, select Object Browser from the View menu.

The Object Browser appears. It shows the available methods and their arguments for the interface.

**5** To view a method, select it from the methods list, and click Show.

Setting Initialization Parameters

The Siebel configuration file, SIEBEL.CFG, supports optional parameters that allow you to determine the behavior of the interfaces. This file is installed with the Siebel application and is located in the same directory as the SIEBEL.EXE file, usually C:\SIEBEL\BIN.

The form of the parameters in the ProductConfigurator section of SIEBEL.CFG appears below.

NOTE: All parameters are optional.

```
[ProductConfigurator]
ConfiguratorDLLName = filename
ConfiguratorExeDir = directoryname
ConfiguratorExeFileName = filename
ConfiguratorModelDir = directoryname
ConfiguratorModelExportFileName = filename
```

NOTE: If a file name or directory name has spaces in it, surround the name with quotation marks (" ").

The parameters are described as follows.

ConfiguratorDLLName = *filename*

Currently, this parameter is supported for the Calico integration only. Specify the name of the DLL file that will handle the configuration integration.

For example, if you are integrating to Calico, you will need to specify the name of the Calico integration DLL, SSPCALXX.DLL, as a parameter to the ConfiguratorDLLName parameter in the configuration file.

NOTE: The “xx” refers to the Siebel version number. For example, the DLL will be named SSPCAL31 for version 3.1, SSPCAL32 for version 3.2, and so on. Use the version that corresponds to the version of Siebel you are running.

ConfiguratorExeDir = *directoryname*

Specify the name of the base directory containing the Configurator executable, for example, C:\CFGGRATOR\. This parameter is supported for OLE2 integrations. If ConfiguratorExeDir is not specified, the executable is found using the PATH environment variable.

ConfiguratorExeFileName = *filename*

Specify the file name of the Configurator executable. This is used in conjunction with ConfiguratorExeDir to locate the executable. If multiple executables are required, specify the file name for the default configuration here. The product-specific executable file name is entered in the Configuration File field in the Marketing Administration/Product List view. This parameter is supported for OLE2 integrations.

ConfiguratorModelDir = *directoryname*

Many Product Configurators store model information in a separate file. The format of the file is vendor-specific. Specify the name of the base directory for the model file that is used for storing configuration models. The actual file name should be specified in Siebel in the Configuration File field in the Marketing Administration/Product List view.

Many Product Configurators use their own source of data to manage and configure products. If data sources are used that are external to Siebel, you must ensure that these external data sources are synchronized with the Siebel data sources. In particular, the Product Configurator integration will validate that products returned from the Product Configurator exist in the Siebel Product Catalog. Also, your Product Configurator will likely validate that products sent from Siebel to the Product Configurator exist in the external data source. It is your system administrator's responsibility to ensure that this synchronization occurs.

ConfiguratorModelExportFileName = *filename*

This parameter is supported for the Calico-specific integration only. Specify the name of a file that will be used to export the model file information. The information is exported in ASCII format so that it can be imported easily to Siebel.

Getting Started

Setting Initialization Parameters

About This Chapter	3-2
Obtaining a Reference to the Product Configurator Interface	3-2
Product Configurator Program Flow	3-3
Siebel Product Configurator Interface Methods	3-4
ConfigurationDone	3-5
GetModelName	3-6
GetSiebelContextInfo	3-7
GetSiebelInitialLineItem	3-9
GetSiebelInitialLineItemCount	3-11
IsVerifyMode	3-13
SetSiebelLineItem	3-14
Sample Program	3-18

About This Chapter

This chapter is a reference to the Siebel Product Configurator Integration Object methods. The first section describes how to obtain a reference to the Siebel Product Configurator Interface; the second section includes a flow diagram that shows an example of using the methods in a Configurator application. The third section includes descriptions of each Siebel Product Configurator Interface method, and the final section gives a sample program that illustrates the process.

Obtaining a Reference to the Product Configurator Interface

The Siebel Product Configurator Interface contains several methods that are used to handle the exchange of data between the Siebel application and the Product Configurator application. To obtain a reference to the Product Configurator Interface, use the following statements:

```
Dim siebelPrdObj As SiebelProductConfigurator
Set siebelPrdObj = GetObject("", "SiebelProductConfigurator")
If (siebelPrdObj is Nothing) then
    MsgBox ("Unable to connect to the Product
           Configurator Interface")
End If
```

NOTE: Every `GetObject()` call must be matched with a call that sets the object to `Nothing`. This will allow proper destruction of the object. For example:

```
Set siebelPrdObj = Nothing
```

Once a valid reference is obtained, all methods described in the reference section can be invoked. After users have completed their interactions with the user interface, the application should call `ConfigurationDone()` to allow the Siebel application to begin processing and updating the quote line items.

Product Configurator Program Flow

The diagram in [Figure 3-1](#) shows one possible example of using the Product Configurator methods.

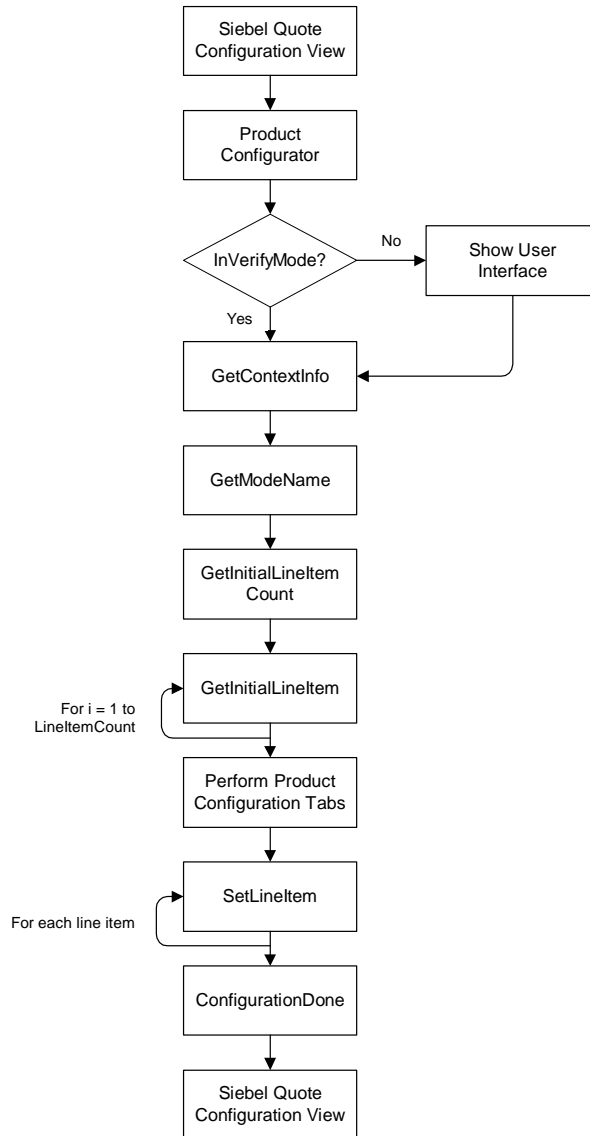


Figure 3-1. Product Configurator Program Flow

Siebel Product Configurator Interface Methods

This section lists the Siebel Product Configurator Interface methods in alphabetical order. The discussion for each method includes the syntax, a description, an example, and error handling information (if applicable).

Briefly, the Siebel Product Configurator Interface methods are as follows:

- **ConfigurationDone** signals to the Siebel application that the Product Configurator application has completed the configuration.
- **GetModelName** allows the Product Configurator application to get the name of the model file that should be used for configuration.
- **GetSiebelContextInfo** allows the Product Configurator application to get contextual information related to the Siebel quote.
- **GetSiebelInitialLineItem** allows the Product Configurator application to get information from the Siebel application for each line item belonging to the current solution. A *solution* is the name of a group of products configured together on a quote.
- **GetSiebelInitialLineItemCount** allows the Product Configurator application to get the number of quote line items belonging to the current solution.
- **IsVerifyMode** allows the Product Configurator application to determine if the Verify button was clicked by the user of the Siebel application to invoke the current configuration.
- **SetSiebelLineItem** allows the Product Configurator application to pass the line item information of the configured product back to the Siebel application.

ConfigurationDone

Signals to the Siebel application that the Product Configurator application has completed the configuration. Until this method is called, the user cannot interact with the Siebel application.

Syntax ConfigurationDone(*bSave*)

Argument	Description
<i>bSave</i>	TRUE or FALSE

Usage A value of TRUE for the *bSave* argument indicates that Siebel should commit the changes. A value of FALSE indicates that Siebel will ignore the changes.

NOTE: Once the ConfigurationDone() method has been called, it is illegal to call any other methods on the Siebel Product Configurator Interface Object. It is recommended that the Product Configurator application immediately free the object (Set siebelPrdObj = Nothing).

Example

```
Private Sub OKButton_Click()  
    If siebelPrdObj Is Nothing Then  
        If Not SiebelPrdObj  
            Then SiebelPrdObj.ConfigurationDone (True)  
        End If  
        Set siebelPrdObj = Nothing  
    End Sub
```

GetModelName

Allows the Product Configurator application to get the name of the model file that should be used for configuration.

Syntax `GetModelName(modelName, errCode)`

Argument	Description
<i>modelName</i>	Name of the model field

Usage The model file name is also passed as an argument to the application via the command line.

All products that can be selected must be synchronized and put into the Siebel products database before starting this configuration process.

Error Handling If an error is encountered with this method, the *errCode* variable will be set with a nonzero integer.

GetSiebelContextInfo

Allows the Product Configurator application to get contextual information related to the Siebel quote.

Syntax

GetSiebelContextInfo(*bGetFirst*, *attribName*, *attribValue*, *errCode*)

Argument	Description
<i>bGetFirst</i>	TRUE or FALSE
<i>attribName</i>	Attribute name
<i>attribValue</i>	Attribute value
<i>errCode</i>	Error code

Usage

Four attributes can be retrieved: Account, Name, Opportunity, Quote Number.

When the argument *bGetFirst* is set to TRUE, the first pair of values (Attribute name, Attribute value for “Account”) is fetched. When *bGetFirst* is set to FALSE, all remaining pairs of values are fetched (Attribute name, Attribute value for “Name”; Attribute name, Attribute value for “Opportunity”; and last, Attribute name, Attribute value for “Quote Number”).

You can retrieve additional information by using the Siebel BusObject Interface. This interface allows you to access all Siebel data. Refer to the *Siebel Tools Reference Manual* for details regarding this interface.

If this function returns a value of TRUE, it indicates that the values were fetched successfully. A return value of FALSE can indicate either that no more records were available to be fetched or that an error occurred. Check the *errCode* argument to determine (a) if no additional records were fetched or (b) if any error occurred.

The arguments *attribName* and *attribValue* are Variants, and they must be initialized properly (set with an initial value) before calling these methods. *Do not initialize the Variants with empty strings.*

Error Handling

If an error is encountered with this method, the *errCode* variable will be set with a nonzero integer.

Example

```
Private Sub LoadContextInfoButton_Click()
Dim i As Integer
Dim errCode As Integer
Dim bGetFirst, bStatus As Boolean
Dim attribName, attribValue As Variant

    attribName = "dummyName"
    attribValue = "dummyValue"

    bGetFirst = True
    bStatus = True
    i = 0

    Do While (bStatus)
        If (i <> 0) Then
            bGetFirst = False
        End If
        i = i + 1

        bStatus = siebelPrdObj.GetSiebelContextInfo(bGetFirst, _
                                                    attribName, attribValue, _
                                                    errCode)

        If (bStatus = True) Then
            If (i = 1) Then Text1.Text = attribName + " ** " + _
                attribValue
            If (i = 2) Then Text2.Text = attribName + " ** " + _
                attribValue
            If (i = 3) Then Text3.Text = attribName + " ** " + _
                attribValue
            If (i = 4) Then Text4.Text = attribName + " ** " + _
                attribValue

            Debug.Print "LoadContext()-" & i & " "; _
                attribName & " ** " & attribValue
        Else
            Debug.Print "GetSiebelContextInfo() failed " & i & errCode
        End If
    Loop
End Sub
```

GetSiebelInitialLineItem

Allows the Product Configurator application to get information from the Siebel application for each line item belonging to the current solution.

Syntax

GetSiebelInitialLineItem(*bGetFirst*, *prodName*, *miscInfo*, *quantity*, *rowID*, *errCode*)

Argument	Description
<i>bGetFirst</i>	TRUE or FALSE
<i>prodName</i>	Product name
<i>miscInfo</i>	Miscellaneous information about the line item
<i>quantity</i>	Number of products being quoted on this line item
<i>rowID</i>	Row ID
<i>errCode</i>	Error code

Usage

When the argument *bGetFirst* is set to TRUE, the first set of values (that is, the first row) is fetched. When *bGetFirst* is set to FALSE, the next set of values (or subsequent rows) is fetched.

If this function returns a value of TRUE, it indicates that the values were fetched successfully. A return value of FALSE can indicate either that no more records were available to be fetched or that an error occurred. Check the *errCode* argument to determine (a) if no additional records were fetched or (b) if any error occurred.

The arguments *prodName*, *miscInfo*, *quantity*, and *rowID* are all Variants, and they must be initialized properly (set with an initial value) before calling these methods. *Do not initialize the Variants with empty strings.* The *rowID* field uniquely identifies the different quote line items within Siebel and must be passed back without modifications during the SetSiebelLineItem() call.

Use the *miscInfo* variable to store information specific to the configuration that is used to reinstantiate the Configurator. Siebel will not use this information; it is stored for the Configurator's use only.

Error Handling

If an error is encountered with this method, the *errCode* variable will be set with a nonzero integer.

Example

```
Private Sub LoadQuotesButton_Click()

    Dim bGetFirst, bStatus As Boolean
    Dim i, numItems As Integer

    Dim quoteLineItemArray(3) As String
    Dim productName, productMiscInfo, productQuantity, _
    productRowID As Variant
    Dim errCode As Integer

    productName = "dummyP"
    productMiscInfo = "dummyMiscInfo"
    productQuantity = "dummyQ"
    productRowID = "dummyRowID"

    numItems = siebelPrdObj.GetSiebelInitialLineItemCount
    QuoteItemCount.Text = numItems

    g_maxDataRow = g_maxDataRow + numItems

    bGetFirst = True
    For i = 1 To numItems
        If (i <> 1) Then
            bGetFirst = False
        End If
        bStatus = siebelPrdObj.GetSiebelInitialLineItem _
            (bGetFirst, _
            productName, productMiscInfo, _
            productQuantity, productRowID, _
            errCode)

        If (bStatus = True) Then
            Debug.Print "Load()- " & productName & " _
                " & productMiscInfo & " _
                " & productQuantity & " _
                " & productRowID

            quoteLineItemArray(0) = productName
            quoteLineItemArray(1) = productQuantity
            quoteLineItemArray(2) = productMiscInfo
            quoteLineItemArray(3) = productRowID

            AddToFromSiebelQuotes quoteLineItemArray, 3
        End If
    Next i
End Sub
```

```
        Else
            MsgBox "GetInitialItems failed " & " " & productName _
                & errCode
        End If
    Next i
End Sub
```

GetSiebelInitialLineItemCount

Allows the Product Configurator application to get the number of quote line items belonging to the current solution.

Syntax GetSiebelInitialLineItemCount()

Usage This method returns the number of lines (or rows) associated with the current solution. You can use this method to facilitate the loop in which you retrieve your line item data.

Example Private Sub LoadQuotesButton_Click()

```
Dim bGetFirst, bStatus As Boolean
Dim i, numItems As Integer

Dim quoteLineItemArray(3) As String
Dim productName, productMiscInfo, productQuantity, _
    productRowID As Variant
Dim errCode As Integer

    productName = "dummyP"
    productMiscInfo = "dummyMiscInfo"
    productQuantity = "dummyQ"
    productRowID = "dummyRowID"

    numItems = siebelPrdObj.GetSiebelInitialLineItemCount
    QuoteItemCount.Text = numItems

    g_maxDataRow = g_maxDataRow + numItems

    bGetFirst = True
```

```
For i = 1 To numItems
  If (i <> 1) Then
    bGetFirst = False
  End If
  bStatus = siebelPrdObj.GetSiebelInitialLineItem
                                (bGetFirst, _
                                productName, productMiscInfo, _
                                productQuantity, productRowID, _
                                errCode)

  If (bStatus = True) Then
    Debug.Print "Load()- " & productName & " _
                " & productMiscInfo & " _
                " & productQuantity & " _
                " & productRowID

    quoteLineItemArray(0) = productName
    quoteLineItemArray(1) = productQuantity
    quoteLineItemArray(2) = productMiscInfo
    quoteLineItemArray(3) = productRowID

    AddToFromSiebelQuotes quoteLineItemArray, 3
  Else
    MsgBox "GetInitialItems failed " & " " & productName _
          & errCode
  End If
Next i
End Sub
```

IsVerifyMode

Allows the Product Configurator application to determine if the Verify button was clicked by the user of the Siebel application to invoke the current configuration.

Syntax IsVerifyMode()

Usage If this function returns a value of TRUE, it indicates the Verify button was clicked. A return value of FALSE indicates the Configure button was clicked.

The Siebel Solution applet allows for interactive Product Configurator development or validation of a configuration without showing the Product Configurator's user interface. Use this method when you want to determine if the user wants to simply validate the configuration or develop one with the Product Configurator's user interface.

NOTE: This information is passed as an argument to the application via the command line as /Verify.

Example

```
Dim bVerifyMode as Boolean
bVerifyMode = SiebelPrdObj.IsVerifyMode
```

SetSiebelLineItem

Allows the Product Configurator application to pass line item information of the configured product back to the Siebel application.

Syntax

SetSiebelLineItem(*bValid*, *lineStatus*, *productName*, *miscInfo*, *quantity*, *rowID*, *errText*, *errCode*)

Argument	Description
<i>bValid</i>	TRUE or FALSE
<i>lineStatus</i>	Line status value as described in “LineStatus Constants” later in this chapter.
<i>productName</i>	Product name
<i>miscInfo</i>	Miscellaneous information about the line item
<i>quantity</i>	Number of products being quoted on this line item
<i>rowID</i>	Row ID
<i>errText</i>	Error text displayed for the user
<i>errCode</i>	Error code

Usage

Use the *miscInfo* argument to store the information required to reinstantiate the Product Configurator with the product selection. For example, when integrating to Calico, set this column to the value of Object ID and Parent Object ID. Users can manually add new products to the solution in the Siebel Quote Configuration view. These products will have a NULL value for the *miscInfo* field. The Configurator application needs to handle these additional products.

To help the Siebel application uniquely identify the rows to modify in the quote line items, the *rowID* must be set to that which was originally obtained during the GetSiebelInitialLineItem() call. However, the *rowID* should be left blank or empty for product name items not obtained from the GetSiebelInitialLineItem() call (that is, new product name items).

Error Handling

If an error is encountered with this method, the *errCode* variable will be set with a nonzero integer.

Example

```
Private Sub SaveQuotesButton_Click( )

Dim bStatus, bValid As Boolean
Dim lineStatus, numItems, r As Integer

Dim errText, productName, productMiscInfo, _
    productQuantity, productRowID As String
Dim errCode As Integer

    bValid = True
    errText = ""
    lineStatus = LineStatusConstants.CIO_MODEL_LINE_UNKNOWN

    numItems = g_maxDataRow

    For r = 1 To numItems
        FromSblQuotes.Row = r
        FromSblQuotes.Col = 0

        If (FromSblQuotes.Text <> "Deleted") Then
            FromSblQuotes.Col = 1
            productName = FromSblQuotes.Text

            FromSblQuotes.Col = 2
            productQuantity = FromSblQuotes.Text

            FromSblQuotes.Col = 3
            productMiscInfo = FromSblQuotes.Text

            FromSblQuotes.Col = 4
            productRowID = FromSblQuotes.Text

            Debug.Print "Save()- " & bValid & _
                " " & productName & " " & productMiscInfo & _
                " " & productQuantity & " " & productRowID & _
                " " & lineStatus & " " & errText & " _
                " " & errCode
```

```
        bStatus = siebelPrdObj.SetSiebelLineItem(bValid, _  
                                                lineStatus, productName, _  
                                                productMiscInfo, _  
                                                productQuantity, productRowID, _  
                                                errText, errCode)  
    End If  
Next r  
SaveQuotesButton.Enabled = False  
End Sub
```

LineStatus Constants

Use the LineStatus constants to flag line items as deleted, modified, new, or not changed when they are passed back to Siebel in the SetSiebelLineItem method. Or, use the UNKNOWN LineStatus constant to cause Siebel to make its own determination of the line status values.

NOTE: If you decide to set line status using the known values, you must set it for all items. For example, if you call SetSiebelLineItem for ten items, you cannot set a line status of DELETED, MODIFIED, NEW, or NOCHANGE for four of the items and then set a line status of UNKNOWN for the other six items. If the first line you send back to Siebel in the SetSiebelLineItem() method has the value of CIO_MODEL_LINE_UNKNOWN, Siebel will make a comparison of the entire set of returned rows with the set originally sent to the Product Configurator. For each row, Siebel determines if it should add, modify, or delete a corresponding quote line item using the information passed to it in the SetSiebelLineItem() status method.

The LineStatus constants are as follows:

CIO_MODEL_LINE_DELETED	Tells the Siebel application to delete the corresponding quote line item, or you do not want the item sent back to Siebel.
CIO_MODEL_LINE_MODIFIED	Tells the Siebel application to update the corresponding quote line item.
CIO_MODEL_LINE_NEW	Tells the Siebel application to create a new quote line item.
CIO_MODEL_LINE_NOCHANGE	Tells the Siebel application not to update the corresponding quote line item.
CIO_MODEL_LINE_UNKNOWN	Tells the Siebel application to compute the value for each quote line item.

Siebel uses the following logic when making this comparison:

- Siebel assumes a line item has been modified when the product name, quantity, miscellaneous information, or an empty ID is passed back to Siebel with modifications. Siebel will update the corresponding quote line item accordingly.
- Siebel assumes a line item is new when there is an empty row ID being passed to it during the SetSiebelLineItem() call. Siebel will create a new quote line item.
- Siebel assumes a line item has not changed when product name, quantity, miscellaneous information, or row ID are all passed back to Siebel without modifications.

Sample Program

The following sample code is a complete Visual Basic program that creates a Product Configurator Interface.

```
VERSION 5.00
Begin VB.Form Form1
    Caption           = "OLE - Configurator"
    ClientHeight      = 5928
    ClientLeft        = 768
    ClientTop         = 1188
    ClientWidth       = 8724
    LinkTopic         = "Form1"
    PaletteMode       = 1 'UseZOrder
    ScaleHeight       = 5928
    ScaleWidth        = 8724
    Begin VB.TextBox CommandLineEdit
        Enabled        = 0 'False
        Height          = 288
        Left            = 2400
        TabIndex        = 23
        Top             = 120
        Width           = 6132
    End
    Begin VB.TextBox Text4
        BeginProperty Font
            Name          = "Small Fonts"
            Size          = 6.6
            Charset       = 0
            Weight        = 400
            Underline     = 0 'False
            Italic        = 0 'False
            Strikethrough  = 0 'False
        EndProperty
        Height          = 250
        Left            = 4080
        TabIndex        = 19
        Top             = 1080
        Width           = 3372
    End
End
```

```
Begin VB.TextBox Text3
  BeginProperty Font
    Name           = "Small Fonts"
    Size           = 6.6
    Charset        = 0
    Weight         = 400
    Underline      = 0   'False
    Italic         = 0   'False
    Strikethrough  = 0   'False
  EndProperty
  Height          = 250
  Left            = 360
  TabIndex       = 18
  Top            = 1080
  Width          = 3372
End
Begin VB.TextBox Text2
  BeginProperty Font
    Name           = "Small Fonts"
    Size           = 6.6
    Charset        = 0
    Weight         = 400
    Underline      = 0   'False
    Italic         = 0   'False
    Strikethrough  = 0   'False
  EndProperty
  Height          = 250
  Left            = 4080
  TabIndex       = 17
  Top            = 720
  Width          = 3372
End
Begin VB.TextBox Text1
  BeginProperty Font
    Name           = "Small Fonts"
    Size           = 6.6
    Charset        = 0
    Weight         = 400
    Underline      = 0   'False
    Italic         = 0   'False
    Strikethrough  = 0   'False
  EndProperty
```

```
        Height          = 250
        Left            = 360
        TabIndex        = 16
        Top             = 720
        Width           = 3372
    End
Begin VB.CommandButton DeleteRowButton
    Caption           = "Delete Row"
    Enabled           = 0   'False
    Height            = 300
    Left              = 3360
    TabIndex          = 15
    Top               = 5040
    Width             = 1452
End
Begin VB.TextBox LineNumberEdit
    Enabled           = 0   'False
    BeginProperty Font
        Name            = "Small Fonts"
        Size            = 6.6
        Charset         = 0
        Weight          = 400
        Underline       = 0   'False
        Italic          = 0   'False
        Strikethrough   = 0   'False
    EndProperty
    Height            = 250
    Left              = 360
    TabIndex          = 13
    Top               = 4680
    Width             = 492
End
Begin VB.CommandButton UpdateRowButton
    Caption           = "Update Row"
    Enabled           = 0   'False
    Height            = 300
    Left              = 5760
    TabIndex          = 12
    Top               = 5040
    Width             = 1452
End
```

```
Begin VB.CommandButton AddRowButton
    Caption           = "Add Row"
    Enabled           = 0 'False
    Height            = 300
    Left              = 840
    TabIndex          = 11
    Top               = 5040
    Width             = 1452
End
Begin VB.TextBox productRowIDEdit
    Enabled           = 0 'False
    BeginProperty Font
        Name           = "Small Fonts"
        Size           = 6.6
        Charset        = 0
        Weight         = 400
        Underline      = 0 'False
        Italic         = 0 'False
        Strikethrough  = 0 'False
    EndProperty
    Height            = 250
    Left              = 6960
    TabIndex          = 10
    Top               = 4680
    Width             = 1452
End
Begin VB.TextBox productMiscInfoEdit
    BeginProperty Font
        Name           = "Small Fonts"
        Size           = 6.6
        Charset        = 0
        Weight         = 400
        Underline      = 0 'False
        Italic         = 0 'False
        Strikethrough  = 0 'False
    EndProperty
    Height            = 250
    Left              = 4320
    TabIndex          = 9
    Top               = 4680
    Width             = 2532
End
```

```
Begin VB.TextBox productQuantityEdit
  BeginProperty Font
    Name           = "Small Fonts"
    Size           = 6.6
    Charset       = 0
    Weight        = 400
    Underline     = 0 'False
    Italic        = 0 'False
    Strikethrough = 0 'False
  EndProperty
  Height         = 250
  Left           = 2760
  TabIndex      = 8
  Top           = 4680
  Width         = 1452
End
Begin VB.TextBox productNameEdit
  BeginProperty Font
    Name           = "Small Fonts"
    Size           = 6.6
    Charset       = 0
    Weight        = 400
    Underline     = 0 'False
    Italic        = 0 'False
    Strikethrough = 0 'False
  EndProperty
  Height         = 250
  Left           = 960
  TabIndex      = 7
  Top           = 4680
  Width         = 1692
End
Begin VB.CommandButton SaveQuotesButton
  Caption        = "Save Quotes To Siebel"
  Enabled       = 0 'False
  Height        = 300
  Left          = 6480
  TabIndex     = 6
  Top          = 1560
  Width        = 2052
End
```

```
Begin VB.CommandButton LoadQuotesButton
    Caption           = "Load Quotes From Siebel"
    Height            = 300
    Left               = 3960
    TabIndex          = 5
    Top                = 1560
    Width              = 2052
End
Begin VB.TextBox QuoteItemCount
    Enabled            = 0    'False
    BeginProperty Font
        Name            = "Small Fonts"
        Size             = 6.6
        Charset          = 0
        Weight           = 400
        Underline        = 0    'False
        Italic           = 0    'False
        Strikethrough    = 0    'False
    EndProperty
    Height             = 250
    Left                = 2520
    TabIndex           = 3
    Text                = "9999999999999"
    Top                 = 1560
    Width               = 972
End
Begin VB.CommandButton CancelButton
    Caption            = "Cancel"
    Height              = 300
    Left                = 4440
    TabIndex           = 2
    Top                 = 5520
    Width               = 1452
End
Begin VB.CommandButton OKButton
    Caption             = "OK"
    Height              = 300
    Left                = 2400
    TabIndex           = 1
    Top                 = 5520
    Width               = 1452
End
```

```
Begin VB.Frame Frame1
    Height           = 972
    Left             = 240
    TabIndex        = 14
    Top              = 4440
    Width           = 8292
End
Begin VB.Frame Frame2
    Height           = 1092
    Left             = 240
    TabIndex        = 20
    Top              = 360
    Width           = 8292
Begin VB.CommandButton LoadContextInfoButton
    Caption          = "Load Context Information"
BeginProperty Font
    Name             = "Small Fonts"
    Size             = 6.6
    Charset          = 0
    Weight           = 400
    Underline        = 0 'False
    Italic           = 0 'False
    Strikethrough    = 0 'False
EndProperty
    Height           = 612
    Left             = 7320
    TabIndex        = 21
    Top              = 360
    Width           = 900
End
End
Begin VB.PictureBox FromSblQuotes
    BackColor        = &H00C0DCC0&
    Height           = 2412
    Left             = 360
    ScaleHeight      = 2364
    ScaleWidth       = 8124
    TabIndex        = 0
    Top              = 1920
    Width           = 8172
End
```

```
Begin VB.Label Label2
  Caption      = "Command Line Options"
  BeginProperty Font
    Name        = "MS Sans Serif"
    Size        = 7.8
    Charset     = 0
    Weight      = 700
    Underline   = 0 'False
    Italic      = 0 'False
    Strikethrough = 0 'False
  EndProperty
  Height       = 252
  Left         = 360
  TabIndex    = 22
  Top         = 120
  Width       = 2052
End
Begin VB.Label Label1
  Alignment    = 2 'Center
  AutoSize    = -1 'True
  Caption     = "Number of Initial Quote Items"
  Enabled     = 0 'False
  BeginProperty Font
    Name        = "Small Fonts"
    Size        = 6.6
    Charset     = 0
    Weight      = 700
    Underline   = 0 'False
    Italic      = 0 'False
    Strikethrough = 0 'False
  EndProperty
  Height       = 156
  Left         = 288
  TabIndex    = 4
  Top         = 1560
  Width       = 2196
End
Attribute VB_Name = "Form1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
```

```
Option Explicit
Dim g_maxDataRow As Integer
Dim siebelPrdObj As SiebelProductConfigurator
Dim errCode As Integer
Private Sub AddRowButton_Click()

    If (productNameEdit.Text = "") Or (productQuantityEdit.Text = _
        "") Then
        MsgBox ("Both the product Name and quantity should be entered")
    Else
        FromSblQuotes.Row = LineNumberEdit.Text

        FromSblQuotes.Col = 0
        FromSblQuotes.Text = LineNumberEdit.Text

        FromSblQuotes.Col = 1
        FromSblQuotes.Text = productNameEdit.Text

        FromSblQuotes.Col = 2
        FromSblQuotes.Text = productQuantityEdit.Text

        FromSblQuotes.Col = 3
        FromSblQuotes.Text = productMiscInfoEdit.Text

        AddRowButton.Enabled = False
        SaveQuotesButton.Enabled = True
        g_maxDataRow = g_maxDataRow + 1
    End If
End Sub
Private Sub CancelButton_Click()
    If siebelPrdObj Is Nothing Then

    Else
        siebelPrdObj.ConfigurationDone (False)
        Set siebelPrdObj = Nothing
    End If

    Unload Form1
End Sub
```

```
Private Sub DeleteRowButton_Click()  
  
    FromSblQuotes.Col = 0  
    FromSblQuotes.Text = "Deleted" ' mark records that should _  
                                   not be sent  
  
    DeleteRowButton.Enabled = False  
    SaveQuotesButton.Enabled = True  
End Sub  
Private Sub Form_Unload(Cancel As Integer)  
  
    If siebelPrdObj Is Nothing Then  
  
    Else  
        siebelPrdObj.ConfigurationDone (True)  
        Set siebelPrdObj = Nothing  
    End If  
  
End Sub  
Private Sub FromSblQuotes_Click()  
Dim curRow As Integer  
  
    curRow = FromSblQuotes.Row  
  
    If (curRow <= g_maxDataRow) Then  
        FromSblQuotes.Row = curRow  
  
        LineNumberEdit.Text = curRow  
  
        FromSblQuotes.Col = 1  
        productNameEdit.Text = FromSblQuotes.Text  
  
        FromSblQuotes.Col = 2  
        productQuantityEdit.Text = FromSblQuotes.Text  
  
        FromSblQuotes.Col = 3  
        productMiscInfoEdit.Text = FromSblQuotes.Text  
  
        FromSblQuotes.Col = 4  
        productRowIDEdit.Text = FromSblQuotes.Text  
  
        AddRowButton.Enabled = False  
        DeleteRowButton.Enabled = True  
        UpdateRowButton.Enabled = True
```

```
ElseIf (curRow = g_maxDataRow + 1) Then
    LineNumberEdit.Text = curRow
    productNameEdit.Text = ""
    productQuantityEdit.Text = ""
    productMiscInfoEdit.Text = ""
    productRowIDEdit.Text = ""

    AddRowButton.Enabled = True
    UpdateRowButton.Enabled = False
End If
End Sub
Private Sub LoadContextInfoButton_Click()
    Dim i As Integer
    Dim bGetFirst, bStatus As Boolean
    Dim attribName, attribValue As Variant

    attribName = "dummyName"
    attribValue = "dummyValue"

    bGetFirst = True
    bStatus = True
    i = 0

    Do While (bStatus)
        If (i <> 0) Then
            bGetFirst = False
        End If
        i = i + 1

        bStatus = siebelPrdObj.GetSiebelContextInfo(bGetFirst, _
                                                    attribName, attribValue, _
                                                    errCode)

        If (bStatus = True) Then
            If (i = 1) Then Text1.Text = attribName + " ** " + _
                attribValue
            If (i = 2) Then Text2.Text = attribName + " ** " + _
                attribValue
            If (i = 3) Then Text3.Text = attribName + " ** " + _
                attribValue
            If (i = 4) Then Text4.Text = attribName + " ** " + _
                attribValue
```

```
        Debug.Print "LoadContext()-" & i & " "; _
            attribName & " ** " & attribValue
    Else
        Debug.Print "GetSiebelContextInfo() failed " & i
    End If
Loop
End Sub
Private Sub LoadQuotesButton_Click()

    Dim bGetFirst, bStatus As Boolean
    Dim i, numItems As Integer

    Dim quoteLineItemArray(3) As String
    Dim productName, productMiscInfo, productQuantity, _
        productRowID As Variant

    productName = "dummyP"
    productMiscInfo = "dummyMiscInfo"
    productQuantity = "dummyQ"
    productRowID = "dummyRowID"

    numItems = siebelPrdObj.GetSiebelInitialLineItemCount
    QuoteItemCount.Text = numItems

    g_maxDataRow = g_maxDataRow + numItems

    bGetFirst = True
    For i = 1 To numItems
        If (i <> 1) Then
            bGetFirst = False
        End If
        bStatus = siebelPrdObj.GetSiebelInitialLineItem(bGetFirst, _
            productName, productMiscInfo, _
            productQuantity, productRowID, _
            errCode)

        If (bStatus = True) Then
            Debug.Print "Load()- " & productName & " _
                " & productMiscInfo & " _
                " & productQuantity & " _
                " & productRowID
```

```
        quoteLineItemArray(0) = productName
        quoteLineItemArray(1) = productQuantity
        quoteLineItemArray(2) = productMiscInfo
        quoteLineItemArray(3) = productRowID

        AddToFromSiebelQuotes quoteLineItemArray, 3
    Else
        MsgBox "GetInitialItems failed " & " " & productName
    End If
Next i
End Sub
Private Sub OKButton_Click()

    If siebelPrdObj Is Nothing Then

    Else
        If (SaveQuotesButton.Enabled = True) Then
            SaveQuotesButton_Click
        End If
        siebelPrdObj.ConfigurationDone (True)
        Set siebelPrdObj = Nothing
    End If

    Unload Form1

End Sub
Private Sub SaveQuotesButton_Click()

Dim bStatus, bValid As Boolean
Dim lineStatus, numItems, r As Integer

Dim errText, productName, productMiscInfo, _
    productQuantity, productRowID As String

    bValid = True
    errText = ""
    lineStatus = LineStatusConstants.CIO_MODEL_LINE_UNKNOWNN

    numItems = g_maxDataRow

    For r = 1 To numItems
        FromSblQuotes.Row = r
        FromSblQuotes.Col = 0
```

```
If (FromSblQuotes.Text <> "Deleted") Then
    FromSblQuotes.Col = 1
    productName = FromSblQuotes.Text

    FromSblQuotes.Col = 2
    productQuantity = FromSblQuotes.Text

    FromSblQuotes.Col = 3
    productMiscInfo = FromSblQuotes.Text

    FromSblQuotes.Col = 4
    productRowID = FromSblQuotes.Text

    Debug.Print "Save()- " & bValid & _
        " " & productName & " " & productMiscInfo & _
        " " & productQuantity & " " & productRowID & _
        " " & lineStatus & " " & errText & " " & errCode

    bStatus = siebelPrdObj.SetSiebelLineItem(bValid, _
        lineStatus, _
        productName, productMiscInfo, _
        productQuantity, productRowID, _
        errText, errCode)

    End If
Next r
SaveQuotesButton.Enabled = False
End Sub
Private Sub Form_Load()

    Set siebelPrdObj = Nothing

    Set siebelPrdObj = GetObject("", "SiebelProductConfigurator")

    If (siebelPrdObj Is Nothing) Then
        MsgBox "Unable to connect to SiebelProductConfigurator"
        Return
    Else
        CommandLineEdit.Text = Command
    End If
End Sub
Public Function AddToFromSiebelQuotes(name() _
As String, numItems As Integer)
```

```
Dim i As Integer
Static currRow As Integer

    If (currRow = 0) Then
        FromSblQuotes.Row = 0
        FromSblQuotes.ColWidth(0) = 500

        FromSblQuotes.Col = 1
        FromSblQuotes.ColWidth(1) = 2000
        FromSblQuotes.Text = "Product Name"

        FromSblQuotes.Col = 2
        FromSblQuotes.ColWidth(2) = 850
        FromSblQuotes.Text = "Quantity"

        FromSblQuotes.Col = 3
        FromSblQuotes.ColWidth(3) = 2500
        FromSblQuotes.Text = "Misc Info"

        FromSblQuotes.Col = 4
        FromSblQuotes.ColWidth(4) = 2000
        FromSblQuotes.Text = "RowID"
    End If

currRow = currRow + 1

FromSblQuotes.Row = currRow

FromSblQuotes.Col = 0
FromSblQuotes.Text = currRow    ' Display row number

For i = 0 To numItems
    FromSblQuotes.Col = i + 1
    FromSblQuotes.Text = name(i)
Next i

End Function
Private Sub UpdateRowButton_Click()

    If (productNameEdit.Text = "") Or (productQuantityEdit.Text = _
        "") Then
        MsgBox ("Both the productName and quantity should be entered")
    End If
End Sub
```

```
Else
    FromSblQuotes.Row = LineNumberEdit.Text

    FromSblQuotes.Col = 0
    FromSblQuotes.Text = LineNumberEdit.Text

    FromSblQuotes.Col = 1
    FromSblQuotes.Text = productNameEdit.Text

    FromSblQuotes.Col = 2
    FromSblQuotes.Text = productQuantityEdit.Text

    FromSblQuotes.Col = 3
    FromSblQuotes.Text = productMiscInfoEdit.Text

    UpdateRowButton.Enabled = False

    SaveQuotesButton.Enabled = True
End If
End Sub
```

