SIEBEL 7
eBusiness

SIEBEL DEVELOPER'S REFERENCE

VERSION 7.5.3, REV. A

12-FRWMRR

NOVEMBER 2003

Siebel Systems, Inc., 2207 Bridgepointe Parkway, San Mateo, CA 94404

Contents

Chapter 2. Business Component Classes

Chapter 3. Applet Classes

Chapter 4. User Properties

Chapter 5. Tags

Chapter 6. Siebel Templates for Employee Applications

Chapter 7. Siebel Templates for Customer Applications

Chapter 8. Cascading Style Sheets

Chapter 9. Operators, Expressions, and Conditions

Chapter 10. Online Help Development

Index

Introduction

This guide contains reference information for Siebel Tools providing information on user interface elements, tags, templates, cascading style sheets, operators, expressions, conditions, user properties, and online help development.

The topics in this section will be useful primarily to people whose titles or job descriptions match one of the following:

Database Administrators	Persons who administer the database system, including data loading, system monitoring, backup and recovery, space allocation and sizing, and user account management.
Siebel Application Administrators	Persons responsible for planning, setting up, and maintaining Siebel applications.
Siebel Application Developers	Persons who plan, implement, and configure Siebel applications, possibly adding new functionality.
Siebel System Administrators	Persons responsible for the whole system, including installing, maintaining, and upgrading Siebel applications.

Product Modules and Options

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

Revision History

Siebel Developer's Reference

Version 7.5.3, Rev A

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

Topic	Revision
"CSSBCAccountSIS Class"	Added this topic.
"CSSBCContactSIS Class"	Added this topic.
"CSSBCFINNA Class"	Added this topic.
"CSSBCFINOppty Class"	Added this topic.
"CSSBCFINSActivity Class"	Added this topic.
"CSSBCForecast Class"	Added this topic.
"CSSBCForecastBase Class"	Added this topic.
"CSSBCForecastItem Class"	Added this topic.
"CSSBCForecastItemDetail Class"	Added this topic.
"CSSBCPharmaSpecializedAct Class"	Added this topic.
"CSSBCProposal Class"	Added this topic.
"CSSSWEFrameContactOrgChart Class"	Added this topic.
"CSSSWEFrameFINApplication and CSSSWEFrameListFINApplication Classes"	Added this topic.
"CSSSWEFrameListDocGen Class"	Added this topic.
"CSSSWEFrameUserRegistration Class"	Added this topic.
"Affiliated Account Id Field"	Added this topic.
"ApplicationContextType"	Added this topic.
"Application Name"	Added this topic.
"Associate: Completion Timeout (Client)"	Added this topic.

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

Topic	Revision
"Associate: Completion Timeout (Server)"	Added this topic.
"Associate: Sleep Time Between Attempts"	Added this topic.
"AutoAssignSearch"	Added this topic.
"BC eAuto Sales Step"	Added this topic.
"BC eAuto Sales Step Admin"	Added this topic.
"BC Opportunity"	Added this topic.
"BC Position"	Added this topic.
"BO eAuto Sales Step Admin"	Added this topic.
"Calc Actual OnWriteRecord"	Added this topic.
"Contact-Activity BC Name"	Added this topic.
"Contact-Opportunity BC Name"	Added this topic.
"Contact Relationship Type"	Added this topic.
"DataSourceBuscompName"	Added this topic.
"Default Display Field"	Added this topic.
"DisableNewRecord"	Added this topic.
"DocumentContextType"	Added this topic.
"eAuto Enable Create Sales Step"	Added this topic.
"eAuto Status Field Name"	Added this topic.
"eAuto Status Field Value"	Added this topic.
"Enable Create Opportunity"	Added this topic.
"FINS Query Mode Disabled Method n"	Added this topic.
"Forecast Analysis BC"	Added this topic.
"Forecast Rollup"	Added this topic.
"Maintain Master Account"	Added this topic.

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

Topic	Revision
“Master Account Field”	Added this topic.
“Named Search: Forecast Series Date Range”	Added this topic.
“NoDelete Field”	Added this topic.
“Non-SalesRep View Mode SearchSpec”	Added this topic.
“Opportunity Name”	Added this topic.
“Parent Account Field”	Added this topic.
“ParentBC Account Id Field”	Added this topic.
“Parent Id Field”	Added this topic.
“Political Analysis Field”	Added this topic.
“Position Join Fields”	Added this topic.
“Revenue Aggregation Field n”	Added this topic.
“Revenue Associate List”	Added this topic.
“Revenue Field Map: fieldname”	Added this topic.
“Set Primary Sales Rep As Owner”	Added this topic.
“Set User As Contact”	Added this topic.
“Show Required n”	Added this topic.
“Skip Existing Forecast Series Date”	Added this topic.
“TypeRetailNew”	Added this topic.
“TypeRetailUsed”	Added this topic.
“Update Parent BC”	Added this topic.
“Update Planned Field On Set: StartDate, StartTime”	Added this topic.
“Update Status To Synchronized”	Added this topic.
“Update Status To Synchronized Types”	Added this topic.

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

Topic	Revision
"Validate Parent Account"	Added this topic.
"WorkFlow Behaviour"	Added this topic.
"Functions in Calculation Expressions"	Updated this topic to clarify syntax.

Version 7.5.3

Table 2. Changes Made in Version 7.5.3

Topic	Revision
Chapter 2, "Business Component Classes"	Added this chapter.
Chapter 3, "Applet Classes"	Added this chapter.
"All Mode Sort"	Added information about the Normal setting.
"AutoPopulateResponsibility"	Added this topic.
"Contact MVG PreDefault Expression"	Added this topic.
"Credit Check"	Added this topic.
"Credit Check Workflow"	Added this topic.
"DB2 Optimization Level"	Added information concerning expert mode in this topic.
"Email Activity Accepted Status Code"	Added this topic.
"Email Activity New Status Code"	Added this topic.
"Email Activity Rejected Status Code"	Added this topic.
"Email Activity Sent Status Code"	Added this topic.
"Email Manager Compatibility Mode"	Added this topic.
"Extended Quantity Field"	Added this topic.
"Named Method n"	Added this topic.
"NoDataHide"	Added this topic.

Table 2. Changes Made in Version 7.5.3

Topic	Revision
"On Field Update Invoke n"	Added this topic.
"On Field Update Set n"	Revised this topic.
"Post Default Created Date To Date Saved"	Added this topic.
"Primary Position Modification"	Added this topic.
"Private Activity Search Spec"	Added this topic.
"Required"	Added this topic.
"Required Position MVField"	Added this topic.
"Share Home Phone Flag Field"	Added this topic.
"swe:form-applet-layout"	Added this topic.
"Applet Form Grid Layout"	Added this topic.
"Applet Popup Form Grid Layout"	Added this topic.
"Customizing Help for Screens with Generic Help Topic Files"	Added step 2 concerning the META tag.
"Customizing Help Content" on page 703	Added note under step 2 concerning redirect tag.
"Customizing the Help Index"	Revised the "To remove access to the index topic" procedure.
"Updating Siebel Topic Files with Custom Content"	Added note under step 2 concerning redirect tag.
"Customizing Help Content" on page 718	Added note under step 2 concerning redirect tag.
"Migrating Help"	Added note under step 4 concerning redirect tag.

Version 7.5, Rev. B

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

Topic	Revision
Book Title	Changed the title of the book, was formerly *Siebel Tools Reference Information*.

Version 7.5, Rev. A

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

Topic	Revision
"Aspect User Properties"	Added this topic.
"Deep Copy n"	Revised example in step 1 of the procedure and modified section title.
"Deep Delete n"	Revised example in step 1 of the procedure and modified section title.
"List Definitions"	Corrected text in .ListRowOn definition.
"View Detail"	Corrected file names in first paragraphs.
"Arithmetic Operators"	Added this topic.

Version 7.5

Table 5. Changes Made in Version 7.5

Topic	Revision
"Online Help Development"	Moved online help development topics from Online Help Development Guide to *Siebel Developer's Reference*.

Siebel Tools Menus and Toolbars 1

The links below provide access to the topics that describe elements of the Siebel Tools user interface:

Right-Click Menus

In this release of Siebel Tools, the navigation has been improved by enhanced use of right-click menus:

- Launch the Web Applet editor from the applet list in the Object List Editor by right-clicking and choosing Edit Web Layout. If an applet Web template does not already exist, the Web Layout wizard appears.
- Launch the Web View editor directly from the view list in the Object List Editor by right-clicking and choosing Edit Web Layout. If a view Web template does not already exist, the Web Layout wizard appears. In the first wizard, you need to choose a view Web template from a combo box.
- The right-click menu in the Object List Editor for Web Template objects includes these choices:
  - View the template, which brings up the appropriate editor with no mappings.
  - Edit the template, which brings up the internal or external template editor.
- Right-clicking a Siebel Tools toolbar displays the names and status of the toolbars (similar to using View > Toolbars).

Menus

The menus in the menu bar operate as standard Microsoft Windows menus. You click a menu to display the menu options. Menu options that are not available due to the current state of the program are disabled.

File Menu

The File menu contains the following options for repository and object definition management:

Option (Shortcut)	Description
Open Repository	When multiple repositories are present in the development directory, the menu option provides the means to open a repository other than the currently open one. The repository chosen using File > Open Repository becomes the default repository opened each time Siebel Tools is launched.
New Object	Invokes the New Object Wizard for the creation of a list applet, form applet, chart applet, tree applet, business component, report, table, command, picklist, MVG, or view.
Close (CTRL + F4)	Closes the Object List Editor window.
Save (CTRL + S)	Saves changes in the current editing window when you are editing Layout, Menu, or Basic Scripts.
Save All	Saves changes in all open editing windows.
Import	Imports text from an external text file into the Siebel VB Editor window. This text should be in an SBL file format. SBL format is generated when it is exported from the Siebel VB editor.
Export	Allows you to create a text file in delimited or HTML format that lists the property values of an object definition or all object definitions currently displayed in the Object List Editor.
Print Setup	Changes the printer and printing options for printing object visualization view diagrams.
Print Preview	Opens a print preview window for display of an object visualization view.
Print (CTRL + P)	Prints the active object visualization view diagram.
Exit	Closes Siebel Tools.

Edit Menu

The Edit menu options apply to individual object definitions in the Object List Editor. The Edit menu consists of the following options:

Option (Shortcut)	Description
Undo (CTRL + Z)	Reverses the last change to a property value in the Object List Editor or Property window before the object definition is committed.
Redo (CTRL + Y)	Reapplies changes after the Undo command has been executed.
Undo Record	Reverses the creation of a new object definition or all modifications to an existing object definition, so long as the record has not yet been committed.
New Record (CTRL + N)	Creates a new object definition in the Object List Editor, with the cursor positioned in the first required property.
Copy Record (CTRL + B)	Creates a new object definition that is a copy of the currently selected object definition, and duplicates all child object definitions. **NOTE:** Avoid using the Copy Record option, except when the reuse and extension of an existing object definition would be impractical.
Delete Record (CTRL + D)	Deletes the currently selected object definition and its child object definitions. **NOTE:** Avoid using the Delete Record option. If you want to remove an object definition from use, set its Inactive property to TRUE.
Cut (CTRL + X)	In a text property, copies the selected text to the clipboard and deletes the existing text. In the Applet Designer, copies the selected control to the clipboard and deletes the existing control.
Copy (CTRL + C)	In a text property, copies the selected text to the clipboard without deleting it. In the Applet Designer, copies the selected control to the clipboard without deleting it.
Paste (CTRL + V)	Inserts text from the clipboard into a text property at the insertion point. Inserts a control from the clipboard in the Applet Designer.
Delete (DEL)	In a text property, deletes the selected text. In the Applet Designer, deletes the selected control.

Option (Shortcut)	Description
Select All (CTRL + A)	Selects the entire document. In the Applet Designer, selects all controls in the applet.
Change Records	Changes multiple records simultaneously.
Find (CTRL + F)	Finds the specified text in the Siebel Script Editor window.
Replace (CTRL + H)	Replaces the specified text with different text in the Siebel Script Editor window.

View Menu

The View menu options are used to change display environment settings, such as which windows and toolbars appear. It also invokes visualization views, which are diagrams showing object definition relationships. The View menu options and suboptions are:

Option (Shortcut)	Suboption (Shortcut)	Description
Windows	Properties Window	Shows or hides the Properties window.
	Applets Window	Shows or hides the Applets window.
	Controls Window	Shows or hides the Controls window.
	Bookmarks Window	Shows or hides the Bookmarks window.
	Web Templates Window	Shows or hides the Web Templates Explorer window.
	Refresh Windows	Requeries and updates the state of dockable windows.

Option (Shortcut)	Suboption (Shortcut)	Description
Editors	Web Applet Editor	Opens the Applet Designer for the currently selected Applet object definition, or the View Designer for the currently selected View object definition.
	Server Script Editor	Opens the Siebel Script Editor. Editor may be specifically defined or be set to a default.
	Browser Script Editor	Opens the Siebel Web Script Editor, which is used to access scripts that control the presentation and behavior of applet controls and list columns in a Web applet template.
Visualize	View Details	Generates and displays a Details visualization view for the currently selected business component or business object.
	View Relationships	Generates and displays a Relationships visualization view for the currently selected business component or table.
	View Descendents	
	View Web Hierarchy	Generates and displays a Hierarchy visualization view for the currently selected applet, application, business component, screen, or view.
Debug Windows	Calls (CTRL + L)	Opens the Calls window for display of the call stack of the Siebel VB or Siebel eScript script currently being debugged.
	Watch (SHIFT + F9)	Opens the Watch window for display of the values of local variables in the Siebel VB or Siebel eScript script currently being debugged.
	Errors	Opens the Errors window for display of the run-time errors in the Siebel VB or Siebel eScript script currently being debugged.
Preview		The preview of a Web view layout depicts the container page, screen bar, and view bar.
ActiveX Methods		Allows you to view the methods for the current ActiveX control in the Applet Designer.

Option (Shortcut)	Suboption (Shortcut)	Description
Toolbars		Displays or hides the various toolbars: Edit, History, List, Debug, Web Controls, and Configuration Context.
Status Bar		Displays or hides the Status bar at the bottom of the Siebel Tools window.
Object Explorer (CTRL + E)		Displays or hides the Object Explorer window.
Options		Opens the Development Tools Options window, in which you can set general preferences and settings for language, check-in and check-out, list views, scripting, Web template editor, debugging, visualization, Object Explorer, and database. Siebel Tools options are stored in a user preference file, which is located in < *Tools_install_root* > \BIN. The user preference filename is *login_ID&application_name*.spf.

Screens Menu

The Screens menu is empty unless you log on to Siebel Tools as a system administrator. If you have system administrator rights, the following menu options and suboptions appear.

Option (Shortcut)	Suboption (Shortcut)	Description
Application Upgrader	Application Upgrade Object List	Activates the Application Upgrades window, with the Object List area selected.
	Application Upgrade Attribute List	Activates the Application Upgrades window, with the Attribute Differences area selected.
	Application Upgrade Database Version	Displays a list applet of all available database versions, types, sequence numbers, and information about possible derivations.

Option (Shortcut)	Suboption (Shortcut)	Description
System Administration	List of Values	Displays the list of values that are available in your repository.
	Strings	Displays the list of message strings that are available in your repository.
	System Preferences	For viewing system preferences (similar to Application Administration screen, System Preferences view in Siebel eBusiness Applications). The system preference values may be changed by the application user. System Preferences is used by the src\server\sfs\common\locate_fulfill\cssconfigagent.cpp file. The system preference is used by fulfillment and part locator engines for cleaning the rows in a temporary table that is shared by the client and server components. The parameter will only be used if you want to clean up the temporary table. In that case, you must explicitly submit a CleanUp request to fulfillment/part locator engines.

Go Menu

The Go menu contains options for moving through a records list (Back, Forward, Previous Record, Next Record, First Record, Last Record). Primarily it is for creating and navigating to *bookmarks*, which flag object definitions for easy return navigation. Bookmarks are a helpful navigation aid, allowing you to move around among the object definitions of different types you are working on. The Go menu contains the following options:

Option (Shortcut)	Description
Back	Returns to the previously displayed screen in the History.
Forward	Returns to the subsequent displayed screen in the History.
Previous Record (CTRL + UP)	Goes to the object definition above the current selection.

Option (Shortcut)	Description
Next Record (CTRL + DOWN)	Goes to the object definition below the current selection.
First Record (CTRL + PGUP)	Goes to the first object definition in the list.
Last Record (CTRL + PGDN)	Goes to the last object definition in the list.
Add Bookmark	Invokes the Add Bookmark dialog box, for creation of a bookmark to the currently selected object definition.
Bookmark List	Opens the Bookmarks dialog box, for selection of an existing bookmark to navigate to. You can also use this dialog box to rename or delete existing bookmarks.

Query Menu

The Query menu options allow you to create and refine Object List Editor queries, which restrict the list of object definitions that appear in the current Object List Editor window. An option is provided that lets you change the sort order of object definitions in the window. The Query menu options are as follows:

Option (Shortcut)	Description
New Query (CTRL + Q)	Allows you to specify restrictions on the set of object definitions to be displayed in the current Object List Editor window.
Refine Query (CTRL + R)	Allows you to add additional restrictions to the query currently in effect.
Execute Query (ENTER)	Executes the query you have just specified, causing the restrictions to take effect. This has the same effect as pressing ENTER.
Sort Order	Invokes the Sort Order dialog box, for specification of sort order criteria for the list of object definitions in the Object List Editor window.

Reports Menu

The Reports menu may be empty or may list one or more available reports of object definitions and properties, depending on which object type is currently active in the Object Explorer:

Option	Description
Applets by BusComp	Generates a report listing the applets assigned to each business component. Available when applet is selected in the Object Explorer.
Business Component and Fields	Generates a report listing all business components and the fields within each. Available when Business Component is selected in the Object Explorer.
Business Object and Components	Generates a report listing all business objects and the business components within each. Available when Business Object is selected in the Object Explorer.
Project List	Generates a report listing all projects. Available when Project is selected in the Object Explorer.
Tables	Generates a report listing all tables and the columns and Indexes within each. Available when Table is selected in the Object Explorer.
Workflow Objects	Generates a report listing all Workflow Policy Objects. Available when Workflow Policy Object is selected in the Object Explorer's Project window.
Repository Dock Objects	Generates a report listing all Dock Objects. Available when Dock Object is selected in the Object Explorer.
Application Upgrade Object List	Generates a report listing all object definition differences between repository versions. Available when the Application Upgrader is active.
Application Upgrade Attributes List	Generates a report listing all attribute differences between repository versions. Available when the Application Upgrader is active.

Format Menu

The Format menu options are used in the Applet Designer. They provide the means to align, resize, and reposition controls; configure the snap grid; and adjust tab or list column order. Options are also provided for performing an Applet Designer Preview. The Format menu options are as follows.

Option	Description
Align	Aligns the selected items with the selected model.
Make Same Size	Makes all selected items the same size as the selected model.
Space Evenly	Spaces the selected items evenly.
Center in Applet	Centers the selected items.
Grid Settings	Sets the horizontal and vertical grid to an integer between 4 and 30. You can also hide the grid.
Set Tab Order	Active only when a form applet is open in the Applet Designer. Allows you to change the tab sequence of the controls in the applet, resetting the values in the Sequence property of each control. Tab sequence is the order in which the Tab key cycles the user through the controls. Numbers appear next to each control, indicating the order of each in the current tab sequence. You click on each control in the order you want used for the tab sequence and this changes the numbers displayed in the Applet Designer and the values in the Sequence properties of the controls.
List Columns	Displays the List Column Fields dialog box, in which you can add, remove, and reorder the list columns.
Preview	Invokes the Applet Designer Preview, which allows you to preview the applet as it would appear at run time in a specified language and with specified viewing options.
Select Truncated Controls	In the Applet Designer Preview, selects all controls whose strings are too long to fit in the control.
Select Untranslated Controls	In the Applet Designer Preview, selects all controls that have not been translated.

Debug Menu

The Debug menu options control the Siebel VB or Siebel eScript debugger, for use when a script is open in the Siebel Script Editor:

Option (Shortcut)	Description
Start (F5)	Starts the application. A dialog box with startup parameters also appears.
Break (CTRL + BREAK)	Stops the execution of the currently running script. If Siebel VB or Siebel eScript is not executing, no operation is performed.
End	Stops the execution of the application and returns to the Siebel Script Editor window.
Restart (SHIFT + F5)	Restarts the application if a break has occurred.
Step Into (F8)	Executes the next line of script code. If this is a subroutine or procedure call, then execution will continue within that procedure.
Step Over (SHIFT + F8)	Advances the application to the script code line just after the current subroutine or procedure. Execution remains at the level of the current procedure.
Step To Cursor (CTRL + F8)	Executes all lines of code up to the line selected by the cursor.
Toggle Breakpoint (F9)	Sets or removes a breakpoint on a specific line of code.
Clear All Breakpoints (CTRL + SHIFT + F9)	Removes all breakpoints from the current script routine.
Check Syntax	Compiles the current script and verifies syntax.

Tools Menu

The options in the Tools menu are as follows:

Option (Shortcut)	Suboption (Shortcut)	Description
Compile (F7)		Opens the Object Compiler dialog box to compile one or more projects, or all projects in the repository, into an SRF file.
Compile Selected Objects (CTRL + F7)		Opens the Object Compiler dialog box to compile one or more projects, or all projects in the repository, into an SRF file.
Check Out (F10)		Opens the Check Out dialog box, to copy one or more projects from the server to the local database.
Check In (CTRL + F10)		Opens the Check In dialog box, to copy one or more projects from the local database to the server.
Lock Project (ALT + L)		Locks the project that the currently selected object definition is assigned to.
Unlock Project (ALT + U)		Unlocks the project that the currently selected object definition is assigned to.
Add To Archive		Opens the Export To Archive dialog box, for adding the selected top-level object definitions or projects to an archive file.
Import From Archive		Initiates the Import wizard for importing object definitions from an archive file.

Option (Shortcut)	Suboption (Shortcut)	Description
Compare Objects	Selected	Compares two selected objects and graphically displays similarities and differences (in object type and instance), with a list of object properties by name and value.
	Selected vs. Repository	Compares the selected object against the corresponding object in the selected repository and graphically displays similarities and differences.
	Selected vs. Archive	Compares the selected object against the corresponding object in the selected archive file and graphically displays similarities and differences.
	Archive vs. Archive	Compares two selected archive files and graphically displays similarities and differences.
Search Repository		Opens the Search Repository dialog box for performing a search for object definitions based on the text in their names (or other properties) and their object types.
Validate Object		From the Validate dialog box, runs validation on a selected object. Lists any errors by severity, rule number, object name, and error description. Allows changing of options for rules, severity, and enforcement.

Option (Shortcut)	Suboption (Shortcut)	Description
Upgrade	Prepare Repository	Opens the Prepare Repository Wizard, which migrates strings from the S_MSG table, merges labels and fields, and merges templates to specified applets, for selected languages.
	Upgrade Application	Navigates to the Application Objects Upgrade List in the Application Upgrader screen of Siebel Tools, and opens the Merge Repositories dialog box. Used to merge standard and customized repositories.
	Generate EIM Processing Columns	Opens the EIM Processing Column Generator dialog box, from which you create missing EIM processing columns and indexes after merging the repository.
	Web Client Migration	Associates web templates to a group of selected applets and views so that they can be used in the web client.
	Merge Labels and Fields	Opens the Label Merge Field dialog box, from which you copy captions from label controls into the caption property of the corresponding field control for the applets specified in an input file.
	Merge Templates	Opens the Applet Web Template Merge dialog box, from which you select templates to be merged.

Option (Shortcut)	Suboption (Shortcut)	Description
Utilities	Check Labels	Opens the Check Labels dialog box to check the labels in the currently selected applet for sufficient horizontal space when translated into a particular language.
	Generate Actuate Report	When a Report object definition is selected, this menu option generates a data stream file for use in the creation of an Actuate report.
	Generate Help IDs	This option generates the sshelp.hm file, containing correspondences between context ID numbers and text help identifiers that have been specified in Help ID object definitions. This option is no longer used for online help implementation.
	Locale Management	Opens the Locale Migration dialog box, from which you perform localization of selected application text strings from a selected source language to a selected target language.
	Map Fax Properties	When the business component object type is selected in the Object Explorer, this option opens the Map Fax Properties dialog box for the current business component object definition. This dialog box is used to create mappings between fields in the business component and fax software property sheet properties. These mappings support customization of the fax cover sheet and message.
	Build Patch	Initiates the Patch Builder wizard to create a patch file.
	Apply Patch	Opens the Apply Patch window to initiate the patch application process.

Window Menu

The Window menu lists the currently open Object List Editor, Application Designer, visualization view, and other windows, and provides the means to navigate to windows that are currently hidden from view.

If one of the windows is open, the first option on the menu is Close. This closes the currently active window.

Help Menu

The options in the Help menu are as follows:

Option	Description
Contents	Opens the Siebel Tools Online Help.
Using Help	Opens the Siebel Tools Online Help.
Technical Support	Displays the Technical Support Information dialog box, informing you of the phone numbers for calling or sending a fax to Siebel Technical Support. The dialog box also displays information that Technical Support will need from you, such as the version number of your Siebel Tools installation and the command line syntax used to open Siebel Tools.
About Record	Opens a dialog box that displays information about the current object definition, including its creator and creation date.
About SRF	Opens a dialog box that displays information about the most recent full incremental compilations.
About View	Opens a dialog box that displays information about the current screen, business object, and view, including applet layout.
About Siebel Tools	Opens a dialog box identifying the version of Siebel Tools.

Toolbars

There are six toolbars in Siebel Tools. The toolbars, like menu items, are active only when the object types or window that uses them is active. You can show and hide toolbars using the Toolbars option in the View menu. You can also rearrange the toolbars using drag-and-drop functionality.

The six toolbars are:

- "Edit Toolbar" on page 49
- "History Toolbar" on page 50
- "List Toolbar" on page 51
- "Debug Toolbar" on page 52
- "Web Controls Toolbar" on page 53
- "Configuration Context Toolbar" on page 56

Edit Toolbar

The Edit toolbar contains edit tools, the New Object wizard, and undo and redo options. It contains the following buttons:

New. Invokes the New Object Wizard, which allows you to create applets, views, charts and other object definitions.

Save. Saves changes in the current editing window when you are editing Layout, Menu, or Basic Scripts.

Save All. Saves changes in all open editing windows.

Cut. In a text property, copies the selected text to the clipboard and deletes the existing text. In the Applet Designer, copies the selected control to the clipboard and deletes the existing control.

Copy. In a text property, copies the selected text to the clipboard without deleting it. In the Applet Designer, copies the selected control to the clipboard without deleting it.

Paste. Inserts text from the clipboard into a text property at the insertion point. In the Applet Designer, inserts a control from the clipboard.

Undo. Reverses the last change to a property value in the Object List Editor or Property window if the object definition has not been committed.

Redo. Reapplies to a property value the change that was just undone.

Several of these tools are also available from the Edit menu. You can also display a menu of edit tools by selecting a field and right-clicking while the cursor is positioned over the Object List Editor window. See "Edit Menu" on page 35 and "Right-Click Menus" on page 32 for more information.

History Toolbar

The History toolbar contains buttons for retracing your steps and for creating and navigating to bookmarks, which flag object definitions for quick return navigation. Bookmarks are a helpful navigation aid, allowing you to move around quickly among the object definitions of different types you are working on. The History toolbar contains the following buttons:

Go Back. Returns to the previously displayed screen.

Go Forward. Returns to the subsequent displayed screen.

Add Bookmark. Opens the Add Bookmark dialog box, so you can add a bookmark for the currently selected object definition.

Bookmark List. Opens the Bookmarks dialog box, so you can select a bookmark to navigate to. You can also use this window to rename or delete existing bookmarks.

List Toolbar

The List toolbar contains options that apply to object definitions in the Object List Editor. The options let you insert new records, move forward and backward, work with queries, and sort object definitions. The List toolbar contains the following buttons:

Add New Record. Creates a new object definition in the Object List Editor, with the cursor positioned in the first required property.

First Record. Goes to the first object definition in the list.

Previous Record. Goes to the object definition above the current selection.

Next Record. Goes to the object definition below the current selection.

Last Record. Goes to the last object definition in the list.

New Query. Allows you to specify one or more restrictions on the set of object definitions to be displayed in the current Object List Editor window.

Execute Query. Executes the query you have just specified, causing the restrictions to take effect. This has the same effect as pressing ENTER.

Sort Ascending. Changes the order in which object definitions appear by sorting them in ascending order on the currently selected property column.

Sort Descending. Changes the order in which object definitions appear by sorting them in descending order on the currently selected property column.

Debug Toolbar

The Debug toolbar contains buttons that let you access Siebel VB and Siebel eScript debugging tools. Some buttons on the toolbar correspond to options in the Debug Menu.

Check Syntax. Compiles the current script and verifies syntax.

Start. Starts the application. A dialog box with startup parameters also appears.

Break. Stops the execution of the currently running script. If Siebel VB or Siebel eScript is not executing, no operation is performed.

End. Stops the execution of the application and returns to the Siebel Script Editor window.

Toggle Breakpoint. Sets or removes a breakpoint on a specific line of code.

Watch. The Tools menu establishes variable watches, so you can monitor the contents of program variables in the Variable window during execution of Siebel VB and Siebel eScript routines.

Calls. Displays the list of Siebel VB or Siebel eScript routine calls executed up to the point where the application was stopped.

Step Into. Executes the next line of script code. If this is a subroutine or procedure call, then execution will continue within that procedure.

Step Over. Advances the application to the script code line just after the current subroutine or procedure. Execution remains at the level of the current procedure.

Web Controls Toolbar

The Web Controls toolbar contains Applet Designer user interface control tools. The Web Controls toolbar becomes active when you are in the Web Layout Editor. You can reposition the toolbar as a floating window anywhere on the screen, or you can place it with the other toolbars at the top of the screen. The toolbar supports drag-and-drop behavior for the creation and placement of new controls. The Web Controls toolbar contains drop-down lists, fields, and buttons.

Drop-Down Lists and Fields

- **Mode.** This drop-down list lets you select the applet mode (Base or Edit).
- **Template.** This field shows the Web template.
- **Button Type.** This drop-down list lets you insert a custom control type into a template. Works together with the Custom Control button.

Buttons

You can select a custom control from the Button Type drop-down list and then drag the custom control button to a placeholder. You set properties (for example, Caption and Method Invoked) for the control using the Properties window.

Change Template. List of other Web templates to select.

Edit Template. Opens the template editor.

Select. Lets you select an object in the layout.

CheckBox. Creates a check box.

ComboBox. Creates a combo box.

Text. Creates a text box.

TextArea. Creates a text area.

Hidden. Creates hidden HTML.

Password. Creates a text box where the user enters a password during logon.

Link. Creates an HTML link on template.

MailTo. Creates a mail-to link on template.

Button. Creates a button on template.

Label. Creates a label on template.

URL. Creates a link to an external URL on template.

ActiveX. Creates an ActiveX control on template.

Text List Column. Creates a list column that contains HTML text.

Checkbox List Column. Creates a list column that contains HTML check boxes.

Custom Control. Creates a custom control on a template. Works together with Button Type.

Shift to previous placeholder. Moves the selected control to the previous placeholder.

Shift to next placeholder. Moves the selected control to the next placeholder.

NOTE: A new property of the Web Template object allows you to restrict picklists to appropriate types and subtypes of templates. For example, you can restrict applet templates to a list applet template or a form applet template.

Configuration Context Toolbar

The Configuration Context toolbar contains drop-down lists that let you define settings for Web browser layout and scripting.

- **Target Browser Group.** A drop-down list from which you select a target browser group for layout editing and for scripting.
- **Application.** A drop-down list that displays the available Siebel applications (identical with the Application object type in Object Explorer).
- **Edit Mode.** Right-clicking an applet in the Applets list displays a pop-up menu that allows you to edit Web menus, Web layouts, server scripts, and browser scripts. The value that appears in the Edit Mode drop-down list is Base or Edit, depending on the browser group, application, and view.
- **Interactivity.**
- **Variable.**

Business Component Classes 2

Business component classes are the types from which business component objects are instantiated.

This section describes the supported use of these business component classes in Siebel applications. The first two classes are generalized classes, from which the specialized classes are derived. The remainder are the most commonly used business component classes in Siebel applications.

Generalized Business Component classes include:

- CSSBusComp Class
- CSSBCBase Class

Specialized Business Component classes include:

- CSSBCAccountSIS Class
- CSSBCActivity Class
- CSSBCContactSIS Class
- CSSBCFile Class
- CSSBCFINNA Class
- CSSBCFINOppty Class
- CSSBCFINSActivity Class
- CSSBCForecast Class
- CSSBCForecastBase Class
- CSSBCForecastItem Class
- CSSBCForecastItemDetail Class

- CSSBCOppty Class
- CSSBCOrder Class
- CSSBCPharmaSpecializedAct Class
- CSSBCProposal Class
- CSSBCQuote Class
- CSSBCServiceRequest Class
- CSSBCUser Class

CSSBusComp Class

CSSBusComp is a base class from which other business component classes are derived. It provides functionalities through business component user properties and an object interface that are useful in many commonly performed tasks and common situations.

Usage Guidelines

The CSSBusComp class provides base business component functionality. It implements business component user properties and object interface methods that are common to many Siebel applications.

Accessible Methods

not applicable

BC User Properties

The following business component user properties are available for use in business component classes derived from CSSBusComp. For more information on these user properties, see "User Properties" on page 165.

- All Mode Sort
- DB2 Optimization Level
- Deep Delete n

Field User Properties

not applicable

Dependencies and Limitations

none

CSSBCBase Class

CSSBCBase is a base class from which other business component classes are derived. It provides functionalities through business component user properties and invoke methods, such as the On Field Update Invoke user property and the Sequence method, that are useful in many common situations.

Usage Guidelines

The CSSBCBase class provides base business component functionality. It implements business component user properties and methods that are common to many Siebel applications.

Parent

CSSBusComp Class

Accessible Methods

The following methods are accessible in business component classes derived from CSSBCBase. For more information on these methods, see "CSSBCBase Methods" on page 62.

- EvalBoolExpr
- EvalExpr
- IsActive
- Revise
- Sequence
- SetAspect

BC User Properties

The following business component user properties are available for use in business component classes derived from CSSBCBase. For more information on these user properties, see "User Properties" on page 165.

- Aspect User Properties
- DB2 Optimization Level
- Deep Copy n
- Deep Delete n
- Encrypt Key Field
- Encrypt Service Name

- Extended Quantity Field
- Named Method n
- On Field Update Invoke n
- On Field Update Set n
- Sequence Field
- Sequence Use Max
- State Model

Field User Properties

The following field-level user properties are available for use in business component classes derived from CSSBCBase. For more information on these user properties, see "User Properties" on page 165.

- Aspect Default Value: Aspect (Aspect User Properties)
- Encrypted
- Encrypt ReadOnly Field
- Encrypt Source Field
- Required

Dependencies and Limitations

none

CSSBCBase Methods

This section describes the methods that are implemented in the CSSBCBase Class.

EvalBoolExpr

The EvalBoolExpr method evaluates the Siebel expression against the current row and returns the appropriate Boolean (Y/N) value.

Argument	Description
expr_string	The expression to be evaluated.

Origin Implemented in CSSBCBase.

Invocable The EvalBoolExpr method can be invoked through InvokeMethod. It can also be called directly.

EvalExpr

The EvalExpr method evaluates the Siebel expression against the current row and returns the value in the result parameter.

Origin Implemented in CSSBCBase.

Invocable The EvalExpr method can be invoked through InvokeMethod. It can also be called directly.

IsActive

The IsActive method determines whether or not the row is active by reading the Active field value.

Origin Implemented in CSSBCBase.

Invocable The IsActive method can be invoked through InvokeMethod. It can also be called directly.

Revise

The Revise method creates a new revision of a record. This method is similar to the CopyRecord method, except that Revise increments the revision number.

Origin Implemented in CSSBCBase.

Invocable The Revise method can be invoked through InvokeMethod. It can also be called directly.

Sequence

The Sequence method updates number sequences in the Sequence field from *m* to *n*, where *m* is the Predefault value (or 1), and *n* is *m* plus the number of lines.

Origin Implemented in CSSBCBase.

Invocable The Sequence method can be invoked through InvokeMethod. It can also be called directly.

SetAspect

The SetAspect method sets the current aspect for Aspect-based user properties.

Argument	Description
aspect	The name of the aspect to set as current.

Origin Implemented in CSSBCBase.

Invocable The SetAspect method can be invoked through InvokeMethod only. It cannot be called directly.

CSSBCAccountSIS Class

CSSBCAccountSIS is one of many in the hierarchy of classes that make up the Account Module in Siebel Industry Applications. The primary purpose of this class is to manage a hierarchical account through its life cycle. In the Account hierarchy, if a Parent Account is deleted or made a child of another Parent, then the CSSBCAccountSIS class maintains the correct relationships between accounts (for example, making sure that a child account is correctly related to its immediate parent and the ultimate parent). Thus the hierarchical relationship between accounts is maintained when changes are made to any element in the hierarchy.

This class also contains functionality used in Siebel Life Sciences applications that automatically schedules Account calls based on the Account's best times to call.

Usage Guidelines

CSSBCAccountSIS can only be used for Account business components because the internal class methods perform tasks specific to the Account data element, and require a specific set of fields to evaluate the hierarchy.

Parent

CSSBCBase Class

Accessible Methods

There are no accessible methods implemented in CSSBCAccountSIS. However, the inherited CSSBCBase Methods are available from this class.

BC User Properties

The following business component user properties are available for use in CSSBCAccountSIS. For more information on these and other user properties, see "User Properties" on page 165.

- Maintain Master Account (required)
- Master Account Field (required)
- Parent Account Field (required)
- Update Parent BC
- Validate Parent Account (required)

Field User Properties

not applicable

Dependencies and Limitations

The functionality of this class is highly specialized. It is not recommended that you use this class for typical business components. The functionality in this class relies on specific field names and other business components to fully accomplish its tasks. Additionally, for the Auto Schedule functionality, the names of the controls in the Auto Schedule pop-up applet are hard coded in the class, so they must not be changed.

CSSBCActivity Class

CSSBCActivity is a base class for Action-related business components. This class provides support for the creation and manipulation of Actions. It also acts as the specialized backend data supplier for other components, such as Siebel Calendar, Siebel Scheduler, and Siebel Email Response.

Usage Guidelines

The CSSBCActivity class is used for Activity-related business components. You can use the business component user properties to enable behaviors for CSSBCActivity. The methods listed below are mainly defined for coding purpose. Invoking methods directly requires extensive analysis and testing.

Refer to Dependencies and Limitations below for a list of required fields.

Parent

CSSBCBase Class

Accessible Methods

The following methods are accessible from CSSBCActivity. For more information on these methods, see "CSSBCActivity Methods" on page 70.

- BuildActivityArray
- CalendarNewMode
- CleanEmployeeOnNewRecord
- ClearAllowOverrides
- ClearGridBeginEndDate
- ClearModifyFlag
- CompleteActivity
- CreateCommActivity
- CreateExceptionRecord
- CreateOverrideRecord
- DeleteAll
- DeleteInstance
- DeleteThisOne
- EmptyActivityArray
- GetLogin
- IsParticipantsChanged
- IsPrimaryInMVG
- IsRecordRepActivity
- IsRepActivityId
- PositionToNewRecFromAux
- RefreshFields
- RemLoginOnNewRecord
- RemoveFromAuxTeam
- RemoveFromTeam
- ResetInited
- SaveThisOne
- SaveWebCollabData
- SetActivityContact
- SetAlarmFilter
- SetAllowOverrides

- GetPrimaryPositionId
- GetSqlExecutedFlag
- InitCalendar
- IsCalendarMode
- IsInited
- IsInitedLocale
- IsOnlyLoginUser
- IsOriginalRepActivityId
- SetAvailCalUpdateIds
- SetCommTypes
- SetCurrentViewDate
- SetEmployeeId
- SetEmployeeList
- SetGridBeginEndDate
- SetModifyFlag
- SetSkipSqlExecuteFlag

BC User Properties

The following business component user properties are available for use in CSSBCActivity. For more information on these user properties, see "User Properties" on page 165.

- Contact MVG PreDefault Expression
- Email Activity Accepted Status Code (*required*)
- Email Activity New Status Code (*required*)
- Email Activity Rejected Status Code (*required*)
- Email Activity Sent Status Code (*required*)
- Email Manager Compatibility Mode
- Private Activity Search Spec

Field User Properties

not applicable

Dependencies and Limitations

The following fields are required for Activity and Calendar behaviors:

- Display—indicates where this Activity record is displayed.
  Values:
  - **Calendar and Activity**: display in both Calendar and Activity.
  - **To Do and Activities**: display in both To Do and Activity.
  - **Activities Only**: display in Activity only.
- Type—indicates the type of Activity that this record is.

The fields listed in Table 6 are required for CSSBCActivity.

Table 6. Fields Required by CSSBCActivity

Field Name	Description
Primary Owned By Primary Owner Id	Key fields to visibility control.
Orig Appt Id	Original Appointment Id.
Done Due Due Date Exchange Date No Sooner Than Date Planned Planned Completion Repeating Expires Started	Key date field that both Activity and Calendar are dependent on.
Duration Hours Duration Minutes	Duration fields.
Alarm	
Description	
Email Format Email Body	Required for Email response.
Primary Attachment Id	
Display	Indicates where this Activity record should show up.
Appt Alarm Time Min	
Owned By	MVG field for visibility control.
Contact Id Contact First Name	
Status Done Flag	
Repeating Type Repeating	Calendar repeating activity related.

Table 6. Fields Required by CSSBCActivity

Field Name	Description
Percent Complete	
Personal Postal Code Service Region	
Previous Activity Id	

The CSSBCActivity class is required when following classes are defined for an applet:

- CSSFrameAlarmList
- CSSFrameAlarmSeeOtherList
- CSSFrameCalGrid
- CSSFrameCalRerouteBase
- CSSFrameCECalAddModify
- CSSFrameCEGridDay
- CSSFrameCEGridMonth
- CSSFrameCEGridWeek
- CSSFrameCEMultPart
- CSSFrameGanttActivity
- CSSFrameGanttActivityBusyFree
- CSSFrameListCommSrc
- CSSFramePopupCalAppt
- CSSFrameSRActivity
- CSSSWECalToDoFrameList
- CSSSWEFrameActHICalendar
- CSSSWEFrameAlarmListSch

- CSSSWEFrameGanttActivityFs
- CSSSWEFrameGanttHiMode
- CSSSWEFrameInMail
- CSSSWEFrameInMailBody

CSSBCActivity Methods

This section describes the methods that are implemented in the CSSBCActivity Class.

BuildActivityArray

The BuildActivityArray method adds the appointments record from the SQL object, and generates a list of repeating appointment instances that have been overridden.

Origin Implemented in CSSBCActivity.

Invocable You can invoke BuildActivityArray from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

CalendarNewMode

The CalendarNewMode method specifies whether or not the business component is in Calendar New mode.

Origin Implemented in CSSBCActivity.

Invocable You can invoke CalendarNewMode from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

CleanEmployeeOnNewRecord

The CleanEmployeeOnNewRecord method triggers the business component to clear the logged-in User Id from the Owner MVG if the logged-in user is different from employee set on the current business component.

Origin Implemented in CSSBCActivity.

Invocable You can invoke CleanEmployeeOnNewRecord from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

ClearAllowOverrides

The ClearAllowOverrides method turns off the ability for the user to override the search specification on the Activity List view.

Origin Implemented in CSSBCActivity.

Invocable You can invoke ClearAllowOverrides from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

ClearGridBeginEndDate

The ClearGridBeginEndDate method turns off the instantiated flag and clears the sort specification for the business component.

Origin Implemented in CSSBCActivity.

Invocable You can invoke ClearGridBeginEndDate from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

ClearModifyFlag

The ClearModifyFlag method turns off the repeating activity modification flag, which indicates that this is a nonrepeating activity.

Origin Implemented in CSSBCActivity.

Invocable You can invoke ClearModifyFlag from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

CompleteActivity

The CompleteActivity method commits the current record, and passes the current Activity Id to the Complete Activity business service to complete current activity. Once complete, the business component updates the following business components:

- Action
- FS Activity Parts Movement
- FS Expense Item
- Time Sheet Daily Hours

This method can only be called in nonquery mode.

Origin Implemented in CSSBCActivity.

Invocable You can invoke CompleteActivity from applets, business components, and business services.

You can also invoke this method through custom buttons and commands.

CreateCommActivity

The CreateCommActivity method triggers the business component to create a new activity record with the values specified in the arguments. The user will get an error report if one or more fields cannot be set, unless other errors are returned.

The arguments for CreateCommActivity comprise a Field Value Array.

Argument	Description
create_attachment	Indicates whether or not to create attachments for Description and Comment if they are too long.
arg2, ..., argX	Named field values in the form "name = value" that are set into the new Activity record.

Origin Implemented in CSSBCActivity.

Invocable You can invoke CreateCommActivity from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

CreateExceptionRecord

The CreateExceptionRecord method creates a new record as an exception of a repeating appointment.

Origin Implemented in CSSBCActivity.

Invocable You can invoke CreateExceptionRecord from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

CreateOverrideRecord

The CreateOverrideRecord method creates a new ghost record that overrides an instance of a repeating appointment.

Origin Implemented in CSSBCActivity.

Invocable You can invoke CreateOverrideRecord from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

DeleteAll

The DeleteAll method deletes the appointment if the user is the primary owner, or dissociates the user from the appointment if the user is not the primary owner.

For repeating and nonrepeating appointments, the DeleteAll method acts on the record.

Origin Implemented in CSSBCActivity.

Invocable You can invoke DeleteAll from applets, business components, and business services.

You can also invoke this method through custom buttons and commands.

DeleteInstance

The DeleteInstance method calls the DeleteThisOne method to delete an instance of the appointment if the user is the primary owner, or dissociates the user from an instance of the appointment if the user is not the primary owner.

For a repeating appointment, the DeleteInstance method acts on a single instance. For a nonrepeating appointment, the DeleteInstance method acts on the record.

Origin Implemented in CSSBCActivity.

Invocable You can invoke DeleteInstance from applets, business components, and business services.

You can also invoke this method through custom buttons and commands.

DeleteThisOne

The DeleteThisOne method deletes an instance of the appointment if the user is the primary owner, or dissociates the user from an instance of the appointment if the user is not the primary owner. This method is called by DeleteInstance.

Origin Implemented in CSSBCActivity.

Invocable You can invoke DeleteThisOne from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

EmptyActivityArray

The EmptyActivityArray method clears the activity array for calendar.

Origin Implemented in CSSBCActivity.

Invocable You can invoke EmptyActivityArray from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

GetLogin

The GetLogin method returns a string that consists of the logged-in user's name and ID. You can use string-to-array conversion to reference the values separately.

Origin Implemented in CSSBCActivity.

Invocable You can invoke GetLogin from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

GetPrimaryPositionId

The GetPrimaryPositionId method returns a string that consists of the primary position id of the logged in user, and Y/N system preference value of Auto Calendar Manager Access for this position.

Origin Implemented in CSSBCActivity.

Invocable You can invoke GetPrimaryPositionId from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

GetSqlExecutedFlag

The GetSqlExecutedFlag method returns a string value of TRUE/FALSE that indicates whether or not Execute will be skipped for this business component.

Origin Implemented in CSSBCActivity.

Invocable You can invoke GetSqlExecutedFlag from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

InitCalendar

The InitCalendar method initializes the calendar with the specified parameters.

Argument	Description
view_date	View Date for the calendar.
begine_date	Begin Date for the calendar.
end_date	End Date for the calendar.
begin_time	Begin Time for the calendar.
end_time	End Time for the calendar.
grid	(Y/N) Indicates whether or not to include a grid layout.
type	Calendar Type (DAY, WEEK, MONTH).

Origin Implemented in CSSBCActivity.

Invocable You can invoke InitCalendar from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

IsCalendarMode

The IsCalendarMode method returns a Y/N value to indicate whether or not the application is in Calendar Mode. This method is used for wireless applications only.

Origin Implemented in CSSBCActivity.

Invocable You can invoke IsCalendarMode from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

IsInited

The IsInited method verifies whether or not the calendar is initialized. If initialized, this method returns a string value that consists of following values in sequence:

- View Date
- Begin Date
- End Date
- Begin Time
- End Time
- Grid? (Y/N)
- Calendar Type (DAY, WEEK, MONTH)

You can use string-to-array conversion to reference these values separately.

Origin Implemented in CSSBCActivity.

Invocable You can invoke IsInited from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

IsInitedLocale

The IsInitedLocale method behaves the same as the IsInited method. However, all values are returned as formatted string values according to current locale.

Origin Implemented in CSSBCActivity.

Invocable You can invoke IsInitedLocale from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

IsOnlyLoginUser

The IsOnlyLoginUser method returns a Y/N value to indicate whether or not the logged-in user is the only owner in the Owned By MVG business component, and that there are no records in Contact MVG.

Origin Implemented in CSSBCActivity.

Invocable You can invoke IsOnlyLoginUser from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

IsOriginalRepActivityId

The IsOriginalRepActivityId method returns a Y/N value to indicate whether or not the specified Activity Id is the original repeating Activity Id. If the specified Activity is the original repeating Activity Id, the IsOriginalRepActivityId method returns Y; otherwise it returns N.

Argument	Description
activityId	The Activity Id to check.

Origin Implemented in CSSBCActivity.

Invocable You can invoke IsOriginalRepActivityId from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

IsParticipantsChanged

The IsParticipantsChanged method returns a Y/N value to indicate whether or not new records have been added to either the Owned By MVG or the Contact MVG.

Origin Implemented in CSSBCActivity.

Invocable You can invoke IsParticipantsChanged from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

IsPrimaryInMVG

The IsPrimaryInMVG method returns a Y/N value to indicate whether or not the current employee ID is the primary in the MVG of the specified Field Name.

Argument	Description
fieldname	The name of the field to check.

Origin Implemented in CSSBCActivity.

Invocable You can invoke IsPrimaryInMVG from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

IsRecordRepActivity

The IsRecordRepActivity method returns a Y/N value to indicate whether or not the current activity record is a repeating activity.

Origin Implemented in CSSBCActivity.

Invocable You can invoke IsRecordRepActivity from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

IsRepActivityId

The IsRepActivityId method returns a string value of the Repeating Id of a specified Activity Id, and a Y/N value indicating whether or not the specified Activity Id is a repeating activity. You can use string-to-array conversion to reference the returned values separately.

Argument	Description
activityId	The Activity Id to check.

Origin Implemented in CSSBCActivity.

Invocable You can invoke IsRepActivityId from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

PositionToNewRecFromAux

The PositionToNewRecFromAux method moves the position to the repeat activity record that was created using the SqlWriteRecord method. This method is used for wireless applications only.

Origin Implemented in CSSBCActivity.

Invocable You can invoke PositionToNewRecFromAux from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

RefreshFields

The RefreshFields method calls the notifynewdata method for each field name specified.

Argument	Description
arg1, ..., argX	Each argument for RefreshFields is a field name.

Origin Implemented in CSSBCActivity.

Invocable You can invoke RefreshFields from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

RemLoginOnNewRecord

The RemLoginOnNewRecord method removes the logged-in user's ID from the Owned By MVG.

Origin Implemented in CSSBCActivity.

Invocable You can invoke RemLoginOnNewRecord from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

RemoveFromAuxTeam

The RemoveFromAuxTeam method removes the employees listed in the arguments from Action Employee, where the employee's Activity Id matches the current Activity Id.

Argument	Description
arg1, ..., argX	Each argument for RemoveFromAuxTeam is an Employee Id.

If no Employee Id is specified, this method uses the employee that has been set on the current business component.

Origin Implemented in CSSBCActivity.

Invocable You can invoke RemoveFromAuxTeam from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

RemoveFromTeam

The RemoveFromTeam method removes the employees listed in the arguments from Action Employee, where the employee's Activity Id matches the current Activity Id.

Argument	Description
arg1, ..., argX	Each argument for RemoveFromTeam is an Employee Id.

If no Employee Id is specified, this method uses the employee that has been set on the current business component.

Origin Implemented in CSSBCActivity.

Invocable You can invoke RemoveFromTeam from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

ResetInited

The ResetInited method resets the calendar initialization flag to FALSE.

Origin Implemented in CSSBCActivity.

Invocable You can invoke ResetInited from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

SaveThisOne

The SaveThisOne method creates an exception record and an override record for a repeating appointment.

Origin Implemented in CSSBCActivity.

Invocable You can invoke SaveThisOne from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

SaveWebCollabData

The SaveWebCollabData method saves Web collaboration chat script and other data.

Origin Implemented in CSSBCActivity.

Invocable You can invoke SaveWebCollabData from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

The arguments for SaveWebCollabData comprise a list of chat script and other data.

SetActivityContact

The SetActivityContact method adds a Contact record for CSSBCBase. It can be either a multivalue field (MVF) on an intersection table plus a single-value field (SVF) on the action Primary Contact Id, or just a SVF on the Action business component.

Argument	Description
contactId	The Contact Id to use for the new Contact record.

Origin Implemented in CSSBCActivity.

Invocable You can invoke SetActivityContact from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

SetAlarmFilter

The SetAlarmFilter method turns the Alarm Filter on or off.

Origin Implemented in CSSBCActivity.

Invocable You can invoke SetAlarmFilter from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

SetAllowOverrides

The SetAllowOverrides method allows the ability for the user to override the search specification on the Activity List view.

Origin Implemented in CSSBCActivity.

Invocable You can invoke SetAllowOverrides from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

SetAvailCalUpdateIds

The SetAvailCalUpdateIds method saves the User Id of each user whose calendar the current user can update.

Origin Implemented in CSSBCActivity.

Invocable You can invoke SetAvailCalUpdateIds from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

SetCommTypes

The SetCommTypes method sets the communication type.

Argument	Description
communicationType	The type of communication to set.

Origin Implemented in CSSBCActivity.

Invocable You can invoke SetCommTypes from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

SetCurrentViewDate

The SetCurrentViewDate method sets the current view date for calendar.

Argument	Description
date	The date to set as the current view.

Origin Implemented in CSSBCActivity.

Invocable You can invoke SetCurrentViewDate from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

SetEmployeeId

The SetEmployeeId method sets the Employee Login Id and Employee Login Name for the current Activity business component.

Argument	Description
emp_login_id	The Employee Login Id to set.
emp_login_name	The Employee Login Name to set.

Origin Implemented in CSSBCActivity.

Invocable You can invoke SetEmployeeId from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

SetEmployeeList

The SetEmployeeList method sets Employee Ids for the current business component to support multiple employees. The arguments for SetEmployeeList consist of Employee Ids for each employee to put in the list.

Origin Implemented in CSSBCActivity.

Invocable You can invoke SetEmployeeList from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

SetGridBeginEndDate

The SetGridBeginEndDate method sets the Begin Date and End Date for the grid.

Argument	Description
beginDate	The Begin Date for the grid.
endDate	The End Date for the grid.

Origin Implemented in CSSBCActivity.

Invocable You can invoke SetGridBeginEndDate from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

SetModifyFlag

The SetModifyFlag method sets the repeating activity modification flag.

Origin Implemented in CSSBCActivity.

Invocable You can invoke SetModifyFlag from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

SetSkipSqlExecuteFlag

The SetSkipSqlExecuteFlag method sets a flag to prevent execution of the business component.

Origin Implemented in CSSBCActivity.

Invocable You can invoke SetSkipSqlExecuteFlag from applets, business components, and business services.

However, you cannot invoke this method through custom buttons and commands.

CSSBCContactSIS Class

This class supports contact functionality for Siebel Life Sciences and Siebel eAutomotive applications.

For Siebel Life Sciences applications, this class provides functionality to support the following:

- scheduling
- showing all and affiliated contacts
- updating Position Join fields and making them editable
- setting the Last Call date to the later of the call dates for all positions when contacts are merged

For Siebel eAutomotive applications, this class provides functionality to support the following:

- invoking new correspondence from buttons instead of the application menu bar
- reassigning contacts as well as any associated opportunities and activities
- automatically creating opportunities and sales steps when new records are created (for eDealer applications only)

Usage Guidelines

You can use CSSBCContactSIS to implement Contact functionality in Siebel Life Sciences and Siebel eAutomotive applications.

Parent

CSSBCUser Class

Accessible Methods

The following methods are accessible from CSSBCContactSIS. For more information on these methods, see "CSSBCContactSIS Methods" on page 92.

- AffiliatedContacts
- AllContacts
- NewCorrespondence
- NewPrPostnAssign

BC User Properties

The following business component user properties are available for use in CSSBCContactSIS. For more information on these and other user properties, see "User Properties" on page 165.

- AutoAssignSearch
- BC Opportunity
- BC Position
- Contact-Activity BC Name
- Contact-Opportunity BC Name
- eAuto Status Field Name
- eAuto Status Field Value
- Enable Create Opportunity
- Opportunity Name
- Position Join Fields

Field User Properties

The following field-level user properties are available for use in CSSBCContactSIS. For more information on these and other user properties, see "User Properties" on page 165.

- Affiliated Account Id Field
- ParentBC Account Id Field

Dependencies and Limitations

This class uses the Pharma Professional Position business component; do not inactivate Pharma Professional Position or remove any fields from it.

CSSBCContactSIS Methods

This section describes the methods that are implemented in the CSSBCContactSIS Class.

AffiliatedContacts

The AffiliatedContacts method shows affiliated contacts.

Origin Implemented in CSSBCContactSIS.

Invocable You can invoke AffiliatedContacts from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

AllContacts

The AllContacts method shows all contacts.

Origin Implemented in CSSBCContactSIS.

Invocable You can invoke AllContacts from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

NewCorrespondence

The NewCorrespondence method creates a new correspondence record.

Origin Implemented in CSSBCContactSIS.

Invocable You can invoke NewCorrespondence from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

NewPrPostnAssign

The NewPrPostnAssign method assigns a position to be the Primary Sales Rep for a contact, and sets that position as the primary on all Opportunities associated with the contact.

Argument	Description
Destination Position Id	The Position Id to assign.
Primary Employee Id	The Primary Employee Id.

Origin Implemented in CSSBCContactSIS.

Invocable You can invoke NewPrPostnAssign from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

CSSBCFile Class

The CSSBCFile class is the business component implementation for file attachments and file replication.

Usage Guidelines

You can use this class to transfer files to and from the Siebel File System.

Parent

CSSBCBase Class

Accessible Methods

The following methods are accessible from CSSBCFile. For more information on these methods, see "CSSBCFile Methods" on page 96.

- CopyPlain
- CreateFile
- CreateFileFromPtr
- CreateTempFile
- DeleteFile
- DeleteFileLink
- DoesSiebelFileExist
- DoesTempFileExist
- GetFile
- GetFileDirect
- GetFileName
- GetFileToDir
- GetFSMFileName
- GetFSMFileNameFast
- GetSiebelFile
- GetUserDirectory
- PutFile

- SetFileSizeAndTime
- UpdateSrcFromLink
- Inherited CSSBCBase Methods

BC User Properties

not applicable

Field User Properties

not applicable

Dependencies and Limitations

CSSBCFile requires the fields listed in Table 7. In this table, < Prefix > indicates the unique prefix of the field name. Each file attachment business component has a unique field name prefix. For example, the Account Attachment applet has fields named AccntDockStatus, AccntFileDate, AccntFileName, and so on.

Table 7. Fields Required by CSSBCFile

Field Name	Value Type
< Prefix > DockStatus	BOOL
< Prefix > FileAutoUpdFlg	BOOL
< Prefix > FileDate	UTCDATETIME
< Prefix > FileDeferFlg	TEXT
< Prefix > FileDockReqFlg	BOOL
< Prefix > FileExt	TEXT
< Prefix > FileName	TEXT
< Prefix > FileRev	ID
< Prefix > FileSize	NUMBER
< Prefix > FileSrcPath	TEXT
< Prefix > FileSrcType	TEXT
< Prefix > FileStatFlg	BOOL

CSSBCFile Methods

This section describes the methods that are implemented in the CSSBCFile Class.

CopyPlain

The CopyPlain method creates a copy of a file in the Siebel File System. Proposal Generator requires this method.

Argument	Description
src	The source file.
dest	The destination file.
bKeepSrcTimeStamp	Boolean indicator of whether or not to keep the timestamp from the source file.
bSetReadOnly	Boolean indicator of whether or not to set the new file to read-only.
bSrcInternal	Boolean indicator of whether or not the source file is internal.
bDestInternal	Boolean indicator of whether or not the destination file is internal.

Origin Implemented in CSSBCFile.

Invocable You can invoke CopyPlain from server script and business services.

You can also invoke this method through custom buttons and commands.

CreateFile

The CreateFile method creates a file in the Siebel File System from an external source.

Argument	Description
srcFilePath	The path to the source file.
keyFileName	The name of the file.
bKeepLink	Boolean indicator of whether or not to keep a link to the external file.
altSrcFileName	An alternative filename.

Origin Implemented in CSSBCFile.

Invocable You can invoke CreateFile from server script and business services.

You can also invoke this method through custom buttons and commands.

CreateFileFromPtr

The CreateFileFromPtr method creates a file in the Siebel File System from an external source.

Argument	Description
pFile	
keyFileName	The name of the file.
fileName	The external name of the file.
bKeepLink	Boolean indicator of whether or not to keep a link to the external file.
bNoFile = FALSE	

Origin Implemented in CSSBCFile.

Invocable You can invoke CreateFileFromPtr from server script and business services.

You can also invoke this method through custom buttons and commands.

CreateTempFile

The CreateTempFile method creates a file in the user's temp directory from an external source.

Argument	Description
srcFilePath	The path to the source file.
keyFileName	The name of the file.
bKeepLink	Boolean indicator of whether or not to keep a link to the external file.

Origin Implemented in CSSBCFile.

Invocable You can invoke CreateTempFile from server script and business services.

You can also invoke this method through custom buttons and commands.

DeleteFile

The DeleteFile method deletes a file. Proposal Generator requires this method.

Argument	Description
filename	The name of the file to delete.
bInternal	Boolean indicator of whether or not the file is internal.

Origin Implemented in CSSBCFile.

Invocable You can invoke DeleteFile from server script and business services.

You can also invoke this method through custom buttons and commands.

DeleteFileLink

The DeleteFileLink method deletes the link between a file in the Siebel File System and an external source.

Argument	Description
keyFieldName	The field that contains the link to be deleted.

Origin Implemented in CSSBCFile.

Invocable You can invoke DeleteFileLink from server script and business services.

You can also invoke this method through custom buttons and commands.

DoesSiebelFileExist

The DoesSiebelFileExist method verifies whether or not a file exists in the Siebel File System.

Argument	Description
keyFieldName	The internal file name.
bExists	A Boolean indicator of whether or not it exists.

Origin Implemented in CSSBCFile.

Invocable You can invoke DoesSiebelFileExist from server script and business services. You can also invoke this method through custom buttons and commands.

DoesTempFileExist

The DoesTempFileExist method verifies whether or not the temporary file exists on the Siebel Server.

Argument	Description
keyFieldName	The internal file name.
bExists	A Boolean indicator of whether or not it exists.
fileName	The external filename.

Origin Implemented in CSSBCFile.

Invocable You can invoke DoesTempFileExist from server script and business services. You can also invoke this method through custom buttons and commands.

GetFile

The GetFile method copies a file from the Siebel File System into a specific directory.

Argument	Description
outFilePath	The destination path.
keyFieldName	The file to copy.
returnVal	The return value.

Origin Implemented in CSSBCFile.

Invocable You can invoke GetFile from server script and business services.

You can also invoke this method through custom buttons and commands.

GetFileDirect

The GetFileDirect method copies a file from the Siebel File System into the user's temp directory. Proposal Generator requires this method.

Argument	Description
outFilePath	The destination path.
externalFileName	The name of the destination file.
idValue	The User Id of the current user.
tableName	The name of the table.
fileExtension	The extension of the file.
fileType	The type of file.
fileRev	The revision number of the file.
returnVal	The return value.
bUnqFileName	Boolean indicator of whether or not the filename must be unique.

Origin Implemented in CSSBCFile.

Invocable You can invoke GetFileDirect from server script and business services.

You can also invoke this method through custom buttons and commands.

GetFileName

The GetFileName method creates a record for a new file in the business component and returns the internal filename that can be used.

Argument	Description
fileName	The name of the file.
extFileName	The external filename.
keyFieldName	The key field name.
returnVal	The return value.

Origin Implemented in CSSBCFile.

Invocable You can invoke GetFileName from server script and business services.

You can also invoke this method through custom buttons and commands.

GetFileToDir

The GetFileToDir method copies a file from the Siebel File System into a specific directory.

Argument	Description
pOutFileDir	A pointer to the output file directory.
keyFieldName	The key field name.
outFilePath	The destination path.
returnVal	The return value.
bUnqFileName	Boolean indicator of whether or not the filename must be unique.

Origin Implemented in CSSBCFile.

Invocable You can invoke GetFileToDir from server script and business services.

You can also invoke this method through custom buttons and commands.

GetFSMFileName

The GetFSMFileName method returns the internal filename.

Argument	Description
fileName	The name of the file.
keyFieldName	The key field name.
returnVal	The return value.

Origin Implemented in CSSBCFile.

Invocable You can invoke GetFSMFileName from server script and business services.

You can also invoke this method through custom buttons and commands.

GetFSMFileNameFast

Similar to GetFSMFileName, the GetFSMFileNameFast method returns the internal filename, but does not verify the existence of the file in the file system.

Argument	Description
fileName	The name of the file.
keyFieldName	The key field name.
returnVal	The return value.

Origin Implemented in CSSBCFile.

Invocable You can invoke GetFSMFileNameFast from server script and business services.

You can also invoke this method through custom buttons and commands.

GetSiebelFile

The GetSiebelFile method gets the name of a file in the Siebel internal format.

Argument	Description
outFilePath	The path to the file.
keyFieldName	The key field name.

Origin Implemented in CSSBCFile.

Invocable You can invoke GetSiebelFile from server script and business services. You can also invoke this method through custom buttons and commands.

GetUserDirectory

The GetUserDirectory method gets the user directory.

Argument	Description
userDirectory	The user directory.

Origin Implemented in CSSBCFile.

Invocable You can invoke GetUserDirectory from server script and business services. You can also invoke this method through custom buttons and commands.

PutFile

The PutFile method updates a file in the Siebel File System from the user's temp directory.

Argument	Description
fileName	The name of the file.
keyFieldName	The key field name.

Origin Implemented in CSSBCFile.

Invocable You can invoke PutFile from server script and business services.

You can also invoke this method through custom buttons and commands.

SetFileSizeAndTime

The SetFileSizeAndTime method sets the FileSize and ModificationTime fields of a file recently created in the pending business component record.

Argument	Description
keyFieldName	The key field name.

Origin Implemented in CSSBCFile.

Invocable You can invoke SetFileSizeAndTime from server script and business services.

You can also invoke this method through custom buttons and commands.

UpdateSrcFromLink

The UpdateSrcFromLink method updates a file in the Siebel File System from its external link.

Argument	Description
keyFieldName	The key field name.

Origin Implemented in CSSBCFile.

Invocable You can invoke UpdateSrcFromLink from server script and business services. You can also invoke this method through custom buttons and commands.

CSSBCFINNA Class

The class allows you to implement Needs Analysis in Siebel Financial Services applications.

Usage Guidelines

You can use this class to find a recommended product, based on the user's answers to Siebel SmartScript questions.

Parent

CSSBCBase Class

Accessible Methods

The following method is accessible from CSSBCFINNA. For more information on this method, see "CSSBCFINNA Methods" on page 109.

- Recommend

BC User Properties

not applicable

Field User Properties

not applicable

Dependencies and Limitations

none

CSSBCFINNA Methods

This section describes the methods that are implemented in the CSSBCFINNA Class.

Recommend

The Recommend method generates a list of recommended products and adds them to the result business component. The recommended products are sorted based on product scores. The score for each product is defined in the Product Administration screen. For more information on Needs Analysis, see *Siebel eFinance Guide*.

The FIN NA Recommend BC user property specifies the name of the business component where recommendations are written. If this user property is not defined, the default business component, FIN NA RET Recommendation, is used.

Origin Implemented in CSSBCFINNA.

Invocable You can also invoke this method only through custom buttons and commands.

CSSBCFINOppty Class

The CSSBCFINOppty class provides behaviors to support special cases related to Opportunities.

Usage Guidelines

You can use the CSSBCFINOppty class to implement a specialized Opportunity business component that executes special cases, such as the following:

- When you want no search specification on the business component so that the user can see all Opportunities in the secure Admin view.
- When you want Secured Opportunities to be viewed only by the associated Sales Rep.

Additionally, Siebel eAutomotive uses this class to enable Send Letter (CTRL + L) functionality directly from the business component.

The Name field (name of the Opportunity) is required by this class.

Parent

CSSBCOppty Class

Accessible Methods

The following methods are accessible from CSSBCFINOppty. For more information on these methods, see "CSSBCFINOppty Methods" on page 111.

- CanUpdate
- CreateSalesStep
- HasFieldChanged
- IsAutoCreateSalesStepEnabled
- NewCorrespondence
- OnWriteRecord
- SetSecureAdminView

BC User Properties

The following business component user properties are available for use in CSSBCFINOppty. For more information on these and other user properties, see "User Properties" on page 165.

- Application Name
- BC eAuto Sales Step

- BC eAuto Sales Step Admin
- BO eAuto Sales Step Admin
- Calc Actual OnWriteRecord
- eAuto Enable Create Sales Step
- Non-SalesRep View Mode SearchSpec
- TypeRetailNew
- TypeRetailUsed

Field User Properties

not applicable

Dependencies and Limitations

none

CSSBCFINOppty Methods

This section describes the methods that are implemented in the CSSBCFINOppty Class.

CanUpdate

The CanUpdate method evaluates the sales stage of the current record and returns a Boolean value indicating whether or not the record can be updated. If the sales stage is Delivered or Won, the CanUpdate method returns FALSE.

Argument	Description
fieldName	The name of the field that contains the sales stage status.

Origin

Overridden from CSSBCOppty.

Invocable

You can invoke CanUpdate from server script, browser script, and business services.

However, you cannot invoke this method through custom buttons and commands.

CreateSalesStep

The CreateSalesStep method populates the Contact Opportunity Sales Step view with the sales steps defined for the organization and creates an opportunity in the Sales Step Administration screen.

Origin Implemented in CSSBCFINOppty.

Invocable You can invoke CreateSalesStep from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

HasFieldChanged

The HasFieldChanged method returns a Boolean value indicating whether or not the specified field has changed.

Argument	Description
BusComp	The name of the business component that contains the field.
fieldName	The name of the field.
newValue	The new value to put in the field.
oldValue	The previous value of the field.

Origin Implemented in CSSBCFINOppty.

Invocable You can invoke HasFieldChanged from server script, browser script, and business services.

However, you cannot invoke this method through custom buttons and commands.

IsAutoCreateSalesStepEnabled

The IsAutoCreateSalesStepEnabled method returns a Boolean value indicating whether or not the eAuto Enable Create Sales Step user property is enabled.

Origin Implemented in CSSBCFINOppty.

Invocable You can invoke IsAutoCreateSalesStepEnabled from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

NewCorrespondence

The NewCorrespondence method opens a new correspondence. It is used for enabling Send Letter (CTRL-L) directly from the business component.

Origin Implemented in CSSBCFINOppty.

Invocable You can invoke NewCorrespondence from browser script, server script, and business services.

You can also invoke this method through custom buttons and commands.

OnWriteRecord

The OnWriteRecord method is overridden to update the WorkPlan's ActualNum of Primary Employee of the Opportunity according to Sales Stage, and Type; and to update StoreBudget's ActualNum according to Organization.

Origin Overridden from CSSBCOppty.

Invocable You can invoke OnWriteRecord from server script, browser script, and business services.

However, you cannot invoke this method through custom buttons and commands.

SetSecureAdminView

When invoked on the business component, the SetSecureAdminView method places the business component into the Secured Admin view mode if the current business component is in the admin mode. This, in turn, clears the search specification of the business component.

Origin Implemented in CSSBCFINOppty.

Invocable You can invoke SetSecureAdminView from browser script, server script, and business services.

You can also invoke this method through custom buttons and commands.

CSSBCFINSActivity Class

This class provides specific functionality to several Siebel Financial Services applications for different types of activities.

Usage Guidelines

You can use the CSSBCFINSActivity class to access the functionality provided by the business component user properties listed below.

Parent

CSSBCActivity Class

Accessible Methods

There are no accessible methods implemented in CSSBCFINSActivity. However, the inherited CSSBCActivity Methods are available from this class.

BC User Properties

The following business component user properties are available for use in CSSBCFINSActivity. For more information on these and other user properties, see "User Properties" on page 165.

- Set Primary Sales Rep As Owner
- Set User As Contact
- Update Status To Synchronized
- Update Status To Synchronized Types
- WorkFlow Behaviour

Field User Properties

not applicable

Dependencies and Limitations

In general, the Action business component should be used when dealing with activities. Few instances call for creating another Activity business component that requires this class. There are many dependencies on field names, LOV values, and user properties embedded in this class.

CSSBCForecast Class

This class provides the functionality for generating forecasts and handles rolling up forecast numbers to the summary records and top-level forecasts, as well as deleting forecasts. CSSBCForecast, the main class in Forecasting, is a highly specialized class for creating forecasts and associating subordinates' forecasts to their manager's forecast.

Usage Guidelines

You can use the CSSBCForecast class to create, modify, delete, and display forecasts. There is additional functionality provided for the Analysis views (such as My Forecast Analysis) and for the Subordinates View where you can add and delete subordinates' forecasts from the manager's forecasts.

Parent

CSSBCForecastBase

Accessible Methods

The following methods are accessible from CSSBCForecast. For more information on these methods, see "CSSBCForecast Methods" on page 117.

- ForecastGenerate
- ForecastSetDefaultForecastSeries
- ForecastSetDefaultForecastSeriesDate
- RollupForecast

BC User Properties

The following business component user properties are available for use in CSSBCForecast. For more information on these and other user properties, see "User Properties" on page 165.

- Associate: Completion Timeout (Client)
- Associate: Completion Timeout (Server)
- Associate: Sleep Time Between Attempts
- Forecast Analysis BC (required)
- Forecast Rollup
- Named Search: Forecast Series Date Range
- Revenue Aggregation Field n
- Skip Existing Forecast Series Date

Field User Properties not applicable

Dependencies and Limitations The Forecast business components are tightly integrated together. The business components that start with Forecast 2000 are heavily dependent on each other. Even small changes can lead to many unexpected behaviors. The Revenue business component is also tied into this, so be careful when making changes to the Revenue business component.

CSSBCForecast Methods

This section describes the methods that are implemented in the CSSBCForecast Class.

ForecastGenerate

The ForecastGenerate method generates the detail and summary records for a Forecast 2000 – Forecast record.

Argument	Description
copyFcstId	Forecast Row Id.
bForce	Boolean indicator of whether or not to force regeneration if forecast already exists.
bAutoForecast	Boolean indicator of whether or not to include subordinates' forecasts.
bNeedRollup	Boolean indicator of whether or not to roll up the generated forecast automatically.
bImplicitCreateFcst	Boolean indicator of whether or not to create a missing subordinate's forecast.

Origin Implemented in CSSBCForecast.

Invocable You can invoke ForecastGenerate from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

ForecastSetDefaultForecastSeries

The ForecastSetDefaultForecastSeries method sets the default forecast series for a user. This method is invoked when a user clicks the New button or changes the owner on a forecast record to auto-populate the Forecast Series field. This method can only be invoked while there is a new record pending.

Origin Implemented in CSSBCForecast.

Invocable You can invoke ForecastSetDefaultForecastSeries from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

ForecastSetDefaultForecastSeriesDate

The ForecastSetDefaultForecastSeriesDate method sets the default forecast series date for a user. This method is invoked when a user clicks the New button or changes the Forecast Series field on a Forecast record to auto-populate the Forecast Series Date field. This method can only be invoked while there is a new record pending.

Origin Implemented in CSSBCForecast.

Invocable You can invoke ForecastSetDefaultForecastSeriesDate from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

RollupForecast

The RollupForecast method calculates the summary records based on the detail records. Summary records are not updated dynamically; they must be manually updated by users when they have finished making changes to their detail records. This method must have a current active row.

Origin Implemented in CSSBCForecast.

Invocable You can invoke RollupForecast from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

CSSBCForecastBase Class

CSSBCForecastBase is strictly a base class. It should be used only as a parent class and no business component should be an instance of CSSBCForecastBase. The CSSBCForecast Class is one example of a subclass of CSSBCForecastBase.

Usage Guidelines

You can use the CSSBCForecastBase class for adding and deleting detail and summary records in forecasts.

Parent

CSSBCAdjustable

Accessible Methods

The following methods are accessible from CSSBCForecastBase. For more information on these methods, see "CSSBCForecastBase Methods" on page 120.

- RollupParentForecast
- SqlCreateAssocList

BC User Properties

not applicable

Field User Properties

not applicable

Dependencies and Limitations

The Forecast business components are tightly integrated together. The business components that start with Forecast 2000 are heavily dependent on each other. Even small changes can lead to many unexpected behaviors. The Revenue business component is also tied into this, so be careful when making changes to the Revenue business component.

CSSBCForecastBase Methods

This section describes the methods that are implemented in the CSSBCForecastBase Class.

RollupParentForecast

The RollupParentForecast method calls the RollupForecast method on the business component.

Origin Implemented in CSSBCForecastBase.

Invocable You can invoke RollupParentForecast from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

SqlCreateAssocList

The SqlCreateAssocList method is overridden to set the association revenue list for the details applet. This method applies search specifications that are retrieved from the Forecast Series by way of the specialized association list.

Origin Overridden from CSSBCAdjustable.

Invocable The SqlCreateAssocList method is only invoked from CreateAssocList on the business component.

CSSBCForecastItem Class

CSSBCForecastItem is used with the CSSBCForecastItemDetail Class to store the detail and summary records for forecasts. The CSSBCForecastItem class specifically stores data relevant to certain aspects of the records (for example, Opportunity, Account, and Revenue Class). Records of type CSSBCForecastItem are also used in the spreadsheet view to determine each row of the spreadsheet while CSSBCForecastItemDetail is used for the individual columns within the spreadsheet.

Usage Guidelines

You can use the CSSBCForecastItem class for adding and deleting of detail and summary records to forecasts. It should not be used in other instances.

Parent

CSSBCAdjustable

Accessible Methods

The following method is accessible from CSSBCForecastItem. For more information on this method, see "CSSBCForecastItem Methods" on page 122.

- SqlDeleteRecord

BC User Properties

The following business component user properties are available for use in CSSBCForecastItem. For more information on these and other user properties, see "User Properties" on page 165.

- Revenue Associate List
- Revenue Field Map: fieldname

Field User Properties

not applicable

Dependencies and Limitations

The Forecast business components are tightly integrated together. The business components that start with Forecast 2000 are heavily dependent on each other. Even small changes can lead to many unexpected behaviors. The Revenue business component is also tied into this, so be careful when making changes to the Revenue business component.

The Forecast 2000 -- Forecast Item business component is used internally by the forecast engine, while the Forecast 2000 -- Forecast Item DynCol business component is used to display the Forecast Details spreadsheet. Each record in the Forecast Item DynCol business component corresponds to a row in the spreadsheet, with the child forecast details forming the dynamic columns (for example, dates). For information on the Dynamic Columns user properties, see *Siebel Forecasting Guide*.

CSSBCForecastItem Methods

This section describes the methods that are implemented in the CSSBCForecastItem Class.

SqlDeleteRecord

The SqlDeleteRecord method deletes the current CSSBCForecastItem record. The CSSBCForecastItem record can be deleted only if the record is owned by the current user. Managers may have the ability to delete subordinates' details within their own forecast, but the CSSBCForecastItem records should only be deleted when the owner of the record deletes it.

Origin Overridden from CSSBCAdjustable.

Invocable The SqlDeleteRecord method is only invoked when a delete record is called.

CSSBCForecastItemDetail Class

CSSBCForecastItemDetail is used with the CSSBCForecastItem Class to store the detail and summary records for forecasts. The CSSBCForecastItemDetail class specifically stores data relevant to certain aspects of the records (for example, Quantity, Price, and Revenue). There can be many CSSBCForecastItemDetail records for each CSSBCForecastItem record. Records of type CSSBCForecastItem are also used in the spreadsheet view to determine each row of the spreadsheet while the CSSBCForecastItemDetail is used to determine the individual columns within the spreadsheet.

Usage Guidelines

You can use the CSSBCForecastItemDetail class for adding and deleting of detail and summary records to forecasts. It should not be used in other instances.

Parent

CSSBCAdjustable

Accessible Methods

The following methods are accessible from CSSBCForecastItemDetail. For more information on these methods, see "CSSBCForecastItemDetail Methods" on page 124.

- AutoAdjust
- UpdateSelectionFromRevn
- UpdateSelectionToRevn

BC User Properties

The following business component user properties are available for use in CSSBCForecastItemDetail. For more information on these and other user properties, see "User Properties" on page 165.

- Revenue Field Map: fieldname

Field User Properties

not applicable

Dependencies and Limitations

The Forecast business components are tightly integrated together. The business components that start with Forecast 2000 are heavily dependent on each other. Even small changes can lead to many unexpected behaviors. The Revenue business component is also tied into this, so be careful when making changes to the Revenue business component.

The Forecast 2000 -- Forecast Item Detail business component is used internally by the forecast engine, while the Forecast 2000 -- Forecast Item Detail Flat business component joins both the Forecast Item and Forecast Item Detail business components to form a flat view (the Forecast Item business component usually stores the nonnumeric properties which can be shared by child details). In general, this means that new fields (and user properties referencing that field) should be added to both business components.

CSSBCForecastItemDetail Methods

This section describes the methods that are implemented in the CSSBCForecastItemDetail Class.

AutoAdjust

The AutoAdjust method updates the current record to have the same values as this record had for the previous forecast. It is called on individual detail records. For example, if you create a forecast one month and update one of the detail records, you can use the AutoAdjust method to automatically update next month's forecast to have the same value. This method can only be called when there is a current active row.

Origin Implemented in CSSBCForecastItemDetail.

Invocable You can invoke AutoAdjust from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

UpdateSelectionFromRevn

The UpdateSelectionFromRevn method synchronizes Forecast Detail records from Revenue records. Since Forecast Detail records are copies of Revenue records, if a Revenue record is updated you might want to synchronize that record with the current detail record. This method can only be called when there are one or more current active rows.

Origin Implemented in CSSBCForecastItemDetail.

Invocable You can invoke UpdateSelectionFromRevn from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

UpdateSelectionToRevn

The UpdateSelectionToRevn method synchronizes Forecast Detail records to Revenue records. Since Forecast Detail records are copies of Revenue records, if the detail record is updated you might want to synchronize that record with the Revenue record. This method can only be called when there are one or more current active rows.

Origin Implemented in CSSBCForecastItemDetail.

Invocable You can invoke UpdateSelectionToRevn from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

CSSBCOppty Class

CSSBCOppty is the specialized business component class for Opportunities.

Usage Guidelines

You can use this class to invoke specialized Opportunities behaviors, such as estimating compensation.

Parent

CSSBCBase Class

Accessible Methods

The following methods are accessible from CSSBCOppty. For more information on these methods, see "CSSBCOppty Methods" on page 127.

- EstimateCmpnsForOneOppty
- EstimateCmpns
- Inherited CSSBCBase Methods

BC User Properties

The following business component user property is available for use in CSSBCOppty. For more information on this user property, see "User Properties" on page 165.

- Primary Position Modification

Field User Properties

not applicable

Dependencies and Limitations

none

CSSBCOppty Methods

This section describes the methods that are implemented in the CSSBCOppty Class.

EstimateCmpnsForOneOppty

The EstimateCmpnsForOneOppty method estimates compensation for a single opportunity.

Origin Implemented in CSSBCOppty.

Invocable You can invoke EstimateCmpnsForOneOppty from server script and business services.

You can also invoke this method through custom buttons and commands.

EstimateCmpns

The EstimateCmpns method estimates compensation for multiple opportunities.

Origin Implemented in CSSBCOppty.

Invocable You can invoke EstimateCmpns from server script and business services.

You can also invoke this method through custom buttons and commands.

CSSBCOrder Class

CSSBCOrder is the specialized business component class for Order Management.

Usage Guidelines

You can use this class to create an Order header and invoke specialized Order behaviors.

Parent

CSSBCBase Class

Accessible Methods

The following methods are accessible from CSSBCOrder. For more information on these methods, see "CSSBCOrder Methods" on page 128.

- AutoOrderSales
- AutoOrderService
- Inherited CSSBCBase Methods

BC User Properties

not applicable

Field User Properties

not applicable

Dependencies and Limitations

none

CSSBCOrder Methods

This section describes the methods that are implemented in the CSSBCOrder Class.

AutoOrderSales

The AutoOrderSales method generates a sales order from quote to order.

Origin

Implemented in CSSBCOrder.

Invocable

You can invoke AutoOrderSales from browser and server script, business services, business components, and applets.

You can also invoke this method through custom buttons and commands.

AutoOrderService

The AutoOrderService method generates a service order from quote to order.

Origin Implemented in CSSBCOrder.

Invocable You can invoke AutoOrderService from browser and server script, business services, business components, and applets.

You can also invoke this method through custom buttons and commands.

CSSBCPharmaSpecializedAct Class

CSSBCPharmaSpecializedAct is one of many in the hierarchy of classes that make up the call-reporting module in Siebel Life Sciences. This class provides functionality for the following:

- Creating inter-table records in S_ACT_EMP for attendee calls
- Calculating time off territory based on business hours
- Making sure that the Start Date, End Date, Duration, Planned, and Planned Completion fields are synchronized by recalculating when one is changed so that the calendar displays activities correctly

Usage Guidelines

This class cannot be used for any other business components other than the Pharma call-reporting business components in Siebel Life Sciences because many of the business component and field names are hard coded in the class methods.

Parent

CSSBCSubmitBusComp

Accessible Methods

not applicable

BC User Properties

The following business component user property is available for use in CSSBCPharmaSpecializedAct. For more information on this and other user properties, see "User Properties" on page 165.

- Update Planned Field On Set: StartDate, StartTime

Field User Properties

not applicable

Dependencies and Limitations

The functionality of CSSBCPharmaSpecializedAct is highly specialized, and it is not recommended that customers use this class for typical business components. The functionality in this class relies on specific field names, other business components, and other classes in the call-reporting hierarchy to fully accomplish its tasks.

CSSBCProposal Class

The CSSBCProposal class provides functionality for the Proposal and Presentation features, including automatically creating proposals, copying proposals, and building proposal content structures. It also provides functionality to support the Proposal Generation business service.

Usage Guidelines

The CSSBCProposal class requires the following fields:

- Name (name of the Proposal record)
- Template Name (name of the Proposal template)

Parent

CSSBCFile Class

Accessible Methods

The following methods are accessible from CSSBCProposal. For more information on these methods, see "CSSBCProposal Methods" on page 132.

- GenerateProposal
- RemoveExistingProposal
- RequestAllFiles
- TemplateAlreadyExists

BC User Properties

not applicable

Field User Properties

not applicable

Dependencies and Limitations

none

CSSBCProposal Methods

This section describes the methods that are implemented in the CSSBCProposal Class.

GenerateProposal

The GenerateProposal method provides auto-proposal capability, which automatically picks the default proposal template and copies the content structure over to the proposal record.

NOTE: If a proposal template has already been picked on the current proposal, the RemoveExistingProposal method is called to remove the existing proposal content structure.

Argument	Description
bRecordExists	If FALSE, then a new record will be created and used to create a new proposal. If TRUE, the current selected proposal is used. The AutoProposal button calls this method with this argument set to FALSE.
bReplace	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.
strTemplateValue	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. The default value of this argument is NULL.

Origin Implemented in CSSBCProposal.

Invocable You can invoke GenerateProposal from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

RemoveExistingProposal

The RemoveExistingProposal method removes the existing content structure (sections, components, and so on) defined on the current proposal record based on a proposal template that has already been picked. After removing the existing structure, this method regenerates a proposal based on the new template. This method is typically called when re-picking a template or when the original proposal template has changed.

Origin Implemented in CSSBCProposal.

Invocable You can invoke RemoveExistingProposal from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

RequestAllFiles

The RequestAllFiles method marks proposals and literature files for downloading to mobile Web clients. It goes through the complete content structure of the current proposal and sets the File Dock Request Flag for the template file and literature files specified under section components. The next time the user synchronizes, the marked files are downloaded to the mobile Web client.

Origin Implemented in CSSBCProposal.

Invocable You can invoke RequestAllFiles from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

TemplateAlreadyExists

The TemplateAlreadyExists method determines if a template has been picked for the current proposal record.

Origin Implemented in CSSBCProposal.

Invocable You can invoke TemplateAlreadyExists from server script, browser script, and business services.

You can also invoke this method through custom buttons and commands.

CSSBCQuote Class

The class CSSBCQuote is for the Quote business component in the Quote module. Quote is part of the Order Management process that allows users to enter products that they want to order as line items, and applies pricing information to the products. The Quote module integrates with other modules to allow users to convert a quote into an order. This class defines specialized functionality applied to Quotes.

Usage Guidelines

You can use this class to invoke specialized Quote behaviors for the Quote business component and other business components that behave the same way and use the same table as the Quote business component.

The Extended Quantity Field user property is required for CSSBCQuote. If the Credit Check user property is set to Y, then the Credit Check Workflow user property is required.

Parent

CSSBCBase Class

Accessible Methods

The following methods are accessible from CSSBCQuote. For more information on these methods, see "CSSBCQuote Methods" on page 135.

- Quote
- UpdateOppty
- Verify
- Inherited CSSBCBase Methods

BC User Properties

The following business component user properties are available for use in CSSBCQuote. For more information on these user properties, see "User Properties" on page 165.

- Credit Check
- Credit Check Workflow
- Extended Quantity Field

Field User Properties

not applicable

Dependencies and Limitations none

CSSBCQuote Methods

This section describes the methods that are implemented in the CSSBCQuote Class.

Quote

The Quote method generates a quote from an opportunity product.

Origin Implemented in CSSBCQuote.

Invocable You can invoke Quote from browser and server script, business services, business components, and applets.

You can also invoke this method through custom buttons and commands.

UpdateOppty

The UpdateOppty method updates the associated opportunity products with the current quote.

Origin Implemented in CSSBCQuote.

Invocable You can invoke UpdateOppty from browser and server script, business services, business components, and applets.

You can also invoke this method through custom buttons and commands.

Verify

The Verify method verifies the current quote with specific criteria.

Origin Implemented in CSSBCQuote.

Invocable You can invoke Verify from browser and server script, business services, business components, and applets.

You can also invoke this method through custom buttons and commands.

CSSBCServiceRequest Class

CSSBCServiceRequest is the class that the Service Request business component is based on. It provides special functionality that is present in the Service Request module. The Service Request module is used to keep track of issues and problems that either the customer or the employee encounter. Among the functionality that is tied to the Service Request are entitlement verification, checking warranty, and calculating commit time.

Usage Guidelines

Service Request is used to log and track problems that need to be resolved. A Service Request has a life cycle, beginning as Open in status to ultimately being Closed when it is resolved. The user should update the status as events occur in the processing of the Service Request. The state model, when enabled, guides the user in navigating to the next possible state to reach a resolution.

The Status and Sub-Status fields are required by this class.

Parent

CSSBCBase Class

Accessible Methods

The following methods are accessible from CSSBCServiceRequest. For more information on these methods, see "CSSBCServiceRequest Methods" on page 137.

- CheckServiceRequestWarranty
- Inherited CSSBCBase Methods

BC User Properties

The following business component user property is available for use in CSSBCServiceRequest. For more information on this user property, see "User Properties" on page 165.

- Post Default Created Date To Date Saved (not required)

Field User Properties

not applicable

Dependencies and Limitations

Both the Entitlement Verification and Commit Time business services depend on the fields and functionality within the Service Request object to perform their respective functions. As a result, the CSSBCEntitlement and the CSSCommitTimeService classes interact frequently with the CSSBCServiceRequest class.

CSSBCServiceRequest Methods

This section describes the methods that are implemented in the CSSBCServiceRequest Class.

CheckServiceRequestWarranty

The CheckServiceRequestWarranty method takes the Asset specified on the Service Request and checks whether or not the warranty applies. If the warranty applies, this method sets the appropriate warranty flags on the record.

Origin Implemented in CSSBCServiceRequest.

Invocable You can invoke CheckServiceRequestWarranty from browser and server script, business services, business components, and applets.

You can also invoke this method through custom buttons and commands.

CSSBCUser Class

CSSBCUser is the base class for people-related business components such as User, Employee, Contact, and Personal Contact. This class provides functionality for user creation and management, external security adapter integration, Sync List for PIM devices, Personal Contact promotion, and other general contact and employee tasks.

Usage Guidelines

The CSSBCUser class is used for people-related business components. Such business components are defined with S_PARTY as the base table and S_CONTACT as the primary extension table. (This is specified with a business component user property Inner Join Extension Table 1 with S_CONTACT as the value.)

Some behaviors can be enabled with user properties. With the exception of Personal Contacts, the records in people-related business components are ultimately Contacts and can show up in Contact views. This includes Employees where extra data sensitivity should be considered.

CSSBCUser has integrated support for external authentication systems such as LDAP and ADSI. There are additional steps that you must perform to enable this for your system. Once enabled, users created and passwords entered are automatically propagated to the external authentication systems upon committing or invoking WriteRecord.

CSSBCUser imposes only the configurable constraint as described under the Required Position MVField user property. Other required fields come from either the business component configuration or the database layer. Typically for people-related business components, this means First Name, Last Name, and Person UId. Additionally, CSSBCUser does not allow the Login field to be cleared once it has been filled in. However, its value can be changed.

Parent

CSSBCBase Class

Accessible Methods

The following methods are accessible from CSSBCUser. For more information on these methods, see "CSSBCUser Methods" on page 139.

- AddToSyncList
- RemoveFromSyncList
- PromotePersonalContact

- Inherited CSSBCBase Methods

BC User Properties

The following business component user properties are available for use in CSSBCUser. For more information on these user properties, see "User Properties" on page 165.

- AutoPopulateResponsibility
- Required Position MVField
- Share Home Phone Flag Field

Field User Properties

not applicable

Dependencies and Limitations

none

CSSBCUser Methods

This section describes the methods that are implemented in the CSSBCUser Class.

AddToSyncList

The AddToSyncList method adds the current user to the Sync List MVG. The Sync List MVG is used by the system to determine which Contacts to sync down to the user's PIM devices (for example, a PDA).

Origin Implemented in CSSBCUser.

Invocable You can invoke AddToSyncList from business components.

You can also invoke this method through custom buttons and commands.

RemoveFromSyncList

The RemoveFromSyncList method removes the current user from the Sync List MVG. The Sync List MVG is used by the system to determine which Contacts to sync down to the user's PIM devices (for example, a PDA).

Origin Implemented in CSSBCUser.

Invocable You can invoke RemoveFromSyncList from business components.

You can also invoke this method through custom buttons and commands.

PromotePersonalContact

The PromotePersonalContact method is used to turn a Personal Contact into a general Contact. Personal Contacts only appear in the Personal Contact list view. When a Personal Contact is promoted to a general Contact, it is no longer accessible in the Personal Contact list view, and instead is accessible in the My Contacts view. This method is called implicitly when a user unchecks the Personal Contact field (labeled as Private on the applets).

Origin Implemented in CSSBCUser.

Invocable You can invoke PromotePersonalContact from business components.

You can also invoke this method through custom buttons and commands.

Applet Classes 3

Applet classes are the types from which frame objects are instantiated. Applet classes are the building blocks of the user interface for Siebel applications.

This section describes the supported use of the most commonly used specialized Applet classes in Siebel applications. The following classes are described:

Relationship Between SWE and Non-SWE Classes

In release 7.x and beyond, the Siebel Web Engine converts all "CSSFrame..." classes, and calls to their contained functionality, to their "CSSSWEFrame..." counterpart at run time. Therefore, the description of these classes in this document apply to both.

If you are configuring legacy applications that use the non-SWE version of the class, you can use the functionality described in this document. However, if you are creating new applets, you should use the SWE version of the class.

CSSFrame and CSSSWEFrame Classes

The CSSFrame and CSSSWEFrame classes represent an applet object in the Siebel UI Frameworks.

The discussion of these classes focuses on the SWE version. However, it applies to both the SWE and the non-SWE version. For more information on the relationship between these classes, see "Relationship Between SWE and Non-SWE Classes" on page 142.

Usage Guidelines

You can use this class to invoke generalized applet methods.

Parent

CSSSWEBase

Accessible Methods

The following methods are accessible from CSSSWEFrame. For more information on these methods, see "CSSFrame and CSSSWEFrame Methods" on page 144.

- ExecuteQuery
- NewQuery
- OnActionsBasedOnLast
- OnActionsCancelQuery
- OnActionsCancelRecord
- OnActionsDeleteitem
- OnActionsDeleteQuery
- OnActionsNewrecord
- OnActionsNextrecord
- OnActionsNextset
- OnActionsPostChanges
- OnActionsPreviousrecord
- OnActionsPreviousset
- OnActionsRefineQuery
- OnActionsWriterecord
- OnDrillDown
- Inherited CSSSWEBase Methods

Applet User Properties

not applicable

Control User Properties

not applicable

Dependencies and Limitations

The CSSSWEFrame class requires CSSSWEFrameMgr and underlying business component objects.

CSSFrame and CSSSWEFrame Methods

This section describes the methods that are available for use in CSSFrame and CSSSWEFrame Classes.

OnActionsBasedOnLast

The OnActionsBasedOnLast method copies a record.

Origin Implemented in CSSSWEFrame.

Invocable You can invoke OnActionsBasedOnLast from server script and business services.

You can also invoke this method through custom buttons and commands.

OnActionsNewrecord

The OnActionsNewrecord method creates a record.

Origin Implemented in CSSSWEFrame.

Invocable You can invoke OnActionsNewrecord from server script and business services.

You can also invoke this method through custom buttons and commands.

OnActionsDeleteitem

The OnActionsDeleteitem method deletes a record.

Origin Implemented in CSSSWEFrame.

Invocable You can invoke OnActionsDeleteitem from server script and business services.

You can also invoke this method through custom buttons and commands.

OnDrillDown

The OnDrillDown method drills down into a field.

Origin Implemented in CSSSWEFrame.

Invocable You can invoke OnDrillDown from server script and business services.

You can also invoke this method through custom buttons and commands.

NewQuery

The NewQuery method starts a new query and changes mode to query mode.

Origin Implemented in CSSSWEFrame.

Invocable You can invoke NewQuery from server script and business services.

You can also invoke this method through custom buttons and commands.

ExecuteQuery

The ExecuteQuery method executes the query.

Origin Implemented in CSSSWEFrame.

Invocable You can invoke ExecuteQuery from server script and business services.

You can also invoke this method through custom buttons and commands.

OnActionsCancelQuery

The OnActionsCancelQuery method cancels the query.

Origin Implemented in CSSSWEFrame.

Invocable You can invoke OnActionsCancelQuery from server script and business services.

You can also invoke this method through custom buttons and commands.

OnActionsRefineQuery

The OnActionsRefineQuery method refines the query.

Origin Implemented in CSSSWEFrame.

Invocable You can invoke OnActionsRefineQuery from server script and business services.

You can also invoke this method through custom buttons and commands.

OnActionsDeleteQuery

The OnActionsDeleteQuery method deletes a saved query.

Origin Implemented in CSSSWEFrame.

Invocable You can invoke OnActionsDeleteQuery from server script and business services.

You can also invoke this method through custom buttons and commands.

OnActionsCancelRecord

The OnActionsCancelRecord method performs an undo on the record.

Origin Implemented in CSSSWEFrame.

Invocable You can invoke OnActionsCancelRecord from server script and business services.

You can also invoke this method through custom buttons and commands.

OnActionsWriterecord

The OnActionsWriterecord method commits a record.

Origin Implemented in CSSSWEFrame.

Invocable You can invoke OnActionsWriterecord from server script and business services.

You can also invoke this method through custom buttons and commands.

OnActionsPostChanges

The OnActionsPostChanges method posts changes in the current record's field data to the business component.

Origin Implemented in CSSSWEFrame.

Invocable You can invoke OnActionsPostChanges from server script and business services.

You can also invoke this method through custom buttons and commands.

OnActionsPreviousrecord

The OnActionsPreviousrecord method navigates to the previous record.

Origin Implemented in CSSSWEFrame.

Invocable You can invoke OnActionsPreviousrecord from server script and business services.

You can also invoke this method through custom buttons and commands.

OnActionsNextrecord

The OnActionsNextrecord method navigates to the next record.

Origin Implemented in CSSSWEFrame.

Invocable You can invoke OnActionsNextrecord from server script and business services.

You can also invoke this method through custom buttons and commands.

OnActionsPreviousset

The OnActionsPreviousset method navigates to previous set of records.

Origin Implemented in CSSSWEFrame.

Invocable You can invoke OnActionsPreviousset from server script and business services.

You can also invoke this method through custom buttons and commands.

OnActionsNextset

The OnActionsNextset method navigates to the next set of records.

Origin Implemented in CSSSWEFrame.

Invocable You can invoke OnActionsNextset from server script and business services.

You can also invoke this method through custom buttons and commands.

CSSFrameBase and CSSSWEFrameBase Classes

The CSSFrameBase and CSSSWEFrameBase classes provide functionalities through applet user properties and invoke methods, such as Aspect user properties and the GotoView method, that are useful in many common situations.

The discussion of these classes focuses on the SWE version. However, it applies to both the SWE and the non-SWE version. For more information on the relationship between these classes, see "Relationship Between SWE and Non-SWE Classes" on page 142.

NOTE: The FrameListBase classes contain the same functionality and behaviors as their FrameBase counterparts. Therefore, the descriptions that follow also apply to CSSFrameListBase and CSSSWEFrameListBase.

Usage Guidelines

The CSSSWEFrameBase class is used for base frame functionality. It implements applet user properties and methods that are common to many applications.

Parent

CSSSWEFrame

Accessible Methods

The following methods are accessible from CSSSWEFrameBase. For more information on these methods, see "CSSFrameBase and CSSSWEFrameBase Methods" on page 149.

- GotoView
- DrillDownToView

Applet User Properties

The following applet user properties are available for use in CSSSWEFrameBase. For more information on these user properties, see "User Properties" on page 165.

- Aspect User Properties
- DeDuplication Results Applet

Control User Properties

not applicable

Dependencies and Limitations

none

CSSFrameBase and CSSSWEFrameBase Methods

This section describes the methods that are available for use in CSSFrameBase and CSSSWEFrameBase Classes.

GotoView

The GotoView method changes the display to the specified view from within a Named Method.

Origin Implemented in CSSSWEFrameBase.

Invocable You can invoke GotoView only through a Named Method.

DrillDownToView

The DrillDownToView method drills down from the applet into a view, using the specified drilldown.

Argument	Description
drilldownName	The name of the drilldown.

Origin Implemented in CSSSWEFrameBase.

Invocable You can invoke DrillDownToView only through InvokeMethod.

CSSFrameList and CSSSWEFrameList Classes

The CSSFrameList and CSSSWEFrameList classes represent a List Applet object in the Siebel UI Frameworks.

The discussion of these classes focuses on the SWE version. However, it applies to both the SWE and the non-SWE version. For more information on the relationship between these classes, see "Relationship Between SWE and Non-SWE Classes" on page 142.

Usage Guidelines

You can use this class to invoke applet-specific methods.

Parent

CSSSWEFrame

Accessible Methods

- Inherited CSSFrame and CSSSWEFrame Methods

Applet User Properties

not applicable

Control User Properties

not applicable

Dependencies and Limitations

CSSSWEFrameList requires a List object.

CSSFrameListBase and CSSSWEFrameListBase Classes

The CSSSWEFrameListBase and CSSSWEFrameListBase classes are used for base frame functionality in list applets.

Usage Guidelines

The FrameListBase classes contain the same functionality and behaviors as their FrameBase counterparts. For a description of these classes, see "CSSFrameBase and CSSSWEFrameBase Classes" on page 148.

CSSFrameListFile and CSSSWEFrameListFile Classes

The CSSFrameListFile and CSSSWEFrameListFile classes are the file attachment frame classes for Siebel UI Frameworks.

The discussion of these classes focuses on the SWE version. However, it applies to both the SWE and the non-SWE version. For more information on the relationship between these classes, see "Relationship Between SWE and Non-SWE Classes" on page 142.

Usage Guidelines

You can use this class to invoke file attachment methods.

Parent

CSSSWEFrameList

Accessible Methods

The following methods are accessible from CSSSWEFrameListFile. For more information on these methods, see "CSSFrameListFile and CSSSWEFrameListFile Methods" on page 153.

- AttachAttachment
- GetOutOfDateFile
- NewFileAttachment
- RechooseFile
- ReplaceFile
- RqstSyncFile
- ShowUrlPopup

Applet User Properties

not applicable

Control User Properties

not applicable

Dependencies and Limitations

CSSSWEFrameListFile requires the underlying business component to use CSSBCFile.

CSSFrameListFile and CSSSWEFrameListFile Methods

This section describes the methods that are available for use in CSSFrameListFile and CSSSWEFrameListFile Classes.

AttachAttachment

The AttachAttachment method parses an HTTP request and attaches the file.

Origin Implemented in CSSSWEFrameListFile.

Invocable You can invoke AttachAttachment from server script and business services. You can also invoke this method through custom buttons and commands.

GetOutOfDateFile

The GetOutOfDateFile method gets an out-of-date file.

Origin Implemented in CSSSWEFrameListFile.

Invocable You can invoke GetOutOfDateFile from server script and business services. You can also invoke this method through custom buttons and commands.

NewFileAttachment

The NewFileAttachment method creates an attachment record in the database and attaches the file.

Origin Implemented in CSSSWEFrameListFile.

Invocable You can invoke NewFileAttachment from server script and business services. You can also invoke this method through custom buttons and commands.

RechooseFile

The RechooseFile method rechooses a file in case of a filename conflict.

Origin Implemented in CSSSWEFrameListFile.

Invocable You can invoke RechooseFile from server script and business services. You can also invoke this method through custom buttons and commands.

ReplaceFile

The ReplaceFile method replaces the existing file with a new one in case of filename conflict.

Origin Implemented in CSSSWEFrameListFile.

Invocable You can invoke ReplaceFile from server script and business services.

You can also invoke this method through custom buttons and commands.

RqstSyncFile

The RqstSyncFile method processes the request to schedule file synchronization. When the local client is unable to get the file attachment from the local file system, you can use this method to request a file from the server on the next synch session.

Origin Implemented in CSSSWEFrameListFile.

Invocable You can invoke RqstSyncFile from server script and business services.

You can also invoke this method through custom buttons and commands.

ShowUrlPopup

The ShowUrlPopup method displays a popup to allow the user to add a URL. This method is only valid in high interactivity applications.

Origin Implemented in CSSSWEFrameListFile.

Invocable You can invoke ShowUrlPopup from server script and business services.

You can also invoke this method through custom buttons and commands.

CSSFrameListWeb and CSSSWEFrameListWeb Classes

The CSSFrameListWeb and CSSSWEFrameListWeb classes are specialized frame classes for ERM, ePortal, and eBriefing applications. CSSSWEFrameListWeb is used by ERM-related modules such as Compensation Planning, Group News, and eContent. It provides functionality for hiding an applet with no data, field-level visibility control, and other useful functionality for ERM applications.

The discussion of these classes focuses on the SWE version. However, it applies to both the SWE and the non-SWE version. For more information on the relationship between these classes, see "Relationship Between SWE and Non-SWE Classes" on page 142.

Usage Guidelines

You can use this class only in ERM-related applications. If you want to use a modified version the CSSSWEFrameListWeb class, then create a new class based on this class. Do not make modifications to the base class.

Parent

CSSSWEFrameList

Accessible Methods

not applicable

Applet User Properties

The following applet user property is available for use in CSSSWEFrameListWeb. For more information on this user property, see "User Properties" on page 165.

- NoDataHide

Control User Properties

not applicable

Dependencies and Limitations

This class should only be used for applets belonging to the ERM module.

CSSFrameSalutation and CSSSWEFrameSalutation Classes

The CSSFrameSalutation and CSSSWEFrameSalutation classes are the Salutation frame classes for SWE.

The discussion of these classes focuses on the SWE version. However, it applies to both the SWE and the non-SWE version. For more information on the relationship between these classes, see "Relationship Between SWE and Non-SWE Classes" on page 142.

Usage Guidelines

You can use the CSSSWEFrameSalutation class to display a salutation, typically on the home page. This applet class should contain a control named Explorer, which it uses to display data. When the frame shows the Explorer control, it pulls out all the field values from the Result Text column, and displays them together.

Parent

CSSSWEFrame

Accessible Methods

Inherited CSSFrame and CSSSWEFrame Methods

Applet User Properties

not applicable

Control User Properties

not applicable

Dependencies and Limitations

none

CSSSWEFrameContactOrgChart Class

CSSSWEFrameContactOrgChart is the Contact Organization Chart frame class.

Usage Guidelines

This class is used as the frame class for the Contact Organization Chart applet.

Parent

CSSSWEFrame

Accessible Methods

The following method is accessible from CSSSWEFrameContactOrgChart. For more information on this method, see "CSSSWEFrameContactOrgChart Methods" on page 157.

- MsgFcDeleteOrgBox

Applet User Properties

The following applet user properties are available for use in CSSSWEFrameContactOrgChart. For more information on these and other user properties, see "User Properties" on page 165.

- Contact Relationship Type
- Default Display Field
- Parent Id Field
- Political Analysis Field

Control User Properties

not applicable

Dependencies and Limitations

none

CSSSWEFrameContactOrgChart Methods

This section describes the methods that are implemented in the CSSSWEFrameContactOrgChart Class.

MsgFcDeleteOrgBox

The MsgFcDeleteOrgBox method selects a box in Contact Organization Chart and then deletes it. The user typically invokes this behavior by right-clicking a box, and then selecting Edit > Delete.

Origin Implemented in CSSSWEFrameContactOrgChart.

Invocable You can invoke this method through custom buttons and commands.

However, you cannot invoke MsgFcDeleteOrgBox from server script, browser script, and business services.

CSSSWEFrameFINApplication and CSSSWEFrameListFINApplication Classes

The CSSSWEFrameFINApplication and CSSSWEFrameListFINApplication classes provide specialized functionality in applets used for personal or business applications, such as applying for credit or opening an account.

Usage Guidelines

These classes should only be used for applets based on the Opportunity business component. This class predefaults the value of the Application Flag field to TRUE.

Parent

CSSSWEFrameBase and CSSSWEFrameListBase

Accessible Methods

not applicable

Applet User Properties

The following applet user property is available for use in CSSSWEFrameFINApplication and CSSSWEFrameListFINApplication. For more information on this and other user properties, see "User Properties" on page 165.

- FINS Query Mode Disabled Method n

Control User Properties

not applicable

Dependencies and Limitations

none

CSSSWEFrameListDocGen Class

The CSSSWEFrameListDocGen class provides the functionality for Siebel Proposal and Presentation features. This class is responsible for setting the document context (for example, Opportunity Proposal, Account Proposal, and so on) and application context (Microsoft Word or Microsoft PowerPoint). It can also submit requests to the Document Server server component to generate proposals.

CSSSWEFrameListDocGen is the base class for both Proposals and Presentations. The two specialized subclasses of this class (CSSSWEFrameListProposal and CSSSWEFrameListPresentation) set the type of document (Proposal or Presentation) and derive all other functionality from the CSSSWEFrameListDocGen class.

Usage Guidelines

CSSSWEFrameListProposal is used for Proposal applets to generate Microsoft Word documents, and CSSSWEFrameListPresentation is used for Presentation applets to generate Microsoft PowerPoint documents.

Parent

CSSSWEFrameFile

Accessible Methods

The following methods are accessible from CSSSWEFrameListDocGen. For more information on these methods, see "CSSSWEFrameListDocGen Methods" on page 161.

- AutoProposal
- RequestAllFiles
- RefreshTemplate
- GenerateDraft

Applet User Properties

The following applet user properties are available for use in CSSSWEFrameListDocGen. For more information on these and other user properties, see "User Properties" on page 165.

- ApplicationContextType
- DataSourceBuscompName
- DisableNewRecord
- DocumentContextType

Control User Properties not applicable

Dependencies and Limitations none

CSSSWEFrameListDocGen Methods

This section describes the methods that are implemented in the CSSSWEFrameListDocGen Class.

AutoProposal

The AutoProposal method passes a call to the CSSBCProposal business component class to generate a proposal using the default template.

Origin Implemented in CSSSWEFrameListDocGen.

Invocable You can invoke AutoProposal from server script and browser script. You can also invoke this method through custom buttons and commands.

RequestAllFiles

The RequestAllFiles method passes a call to the CSSBCProposal business component class to set the File Dock Request Flag for template files and literature files specified in component definitions under the current proposal content structure. For more information, see the description of RequestAllFiles in "CSSBCProposal Methods" on page 132.

Origin Implemented in CSSSWEFrameListDocGen.

Invocable You can invoke RequestAllFiles from server script and browser script. You can also invoke this method through custom buttons and commands.

RefreshTemplate

The RefreshTemplate method checks whether a template has already been picked on the current Proposal record. If the template has been picked, it removes the existing content structure, and then invokes the GenerateProposal method to generate the new content structure based on the new template selected.

Origin Implemented in CSSSWEFrameListDocGen.

Invocable You can invoke RefreshTemplate from server script and browser script. You can also invoke this method through custom buttons and commands.

GenerateDraft

This method checks if the application is running as Mobile Web client or Web client (server mode). If the Mobile Web client is running, the GenerateDraft method calls the Proposal Integrator business service and generates the proposal document. If the Web client is running, it submits a server request to the Document Server server component to generate the proposal document.

Origin Implemented in CSSSWEFrameListDocGen.

Invocable You can invoke GenerateDraft from server script and browser script. You can also invoke this method through custom buttons and commands.

CSSSWEFrameUserRegistration Class

CSSSWEFrameUserRegistration is an applet class that provides functionality for the User Registration module to have applet-level required fields.

Usage Guidelines

In the User Registration process, it is often desirable to require customers to provide pieces of information that are not necessarily a requirement in the creation of the business component record. You can use this applet class to enforce this kind of requirement.

Parent

CSSSWEFrame

Accessible Methods

The following method is accessible from CSSSWEFrameUserRegistration. For more information on this method, see "CSSSWEFrameUserRegistration Methods" on page 164.

- FrameEventMethodxxxxx

Applet User Properties

The following applet user property is available for use in CSSSWEFrameUserRegistration. For more information on this and other user properties, see "User Properties" on page 165.

- Show Required n

Control User Properties

not applicable

Dependencies and Limitations

CSSSWEFrameUserRegistration should be used in the context of the User Registration module. This class assumes that the underlying business component uses the CSSBCUser class (for example, the User Registration business component).

When using the Show Required user property, the corresponding business component must use the CSSBCUser class. Additionally, the control specified in the value of this user property must be present on the applet user interface; otherwise the user has no way of entering a value for the required control.

CSSSWEFrameUserRegistration Methods

This section describes the methods that are implemented in the CSSSWEFrameUserRegistration Class.

FrameEventMethod*xxxxx*

FrameEventMethod*xxxxx* is a method pattern that provides the applet class with a hook that can be used to trigger a run-time event (where *xxxxx* represents the name of the run-time event). For example, a method called FrameEventMethodCancel triggers the Cancel run-time event. Although any method can be used to trigger run-time events, this method type is not tied to any functionality and therefore has no side effects.

Origin Implemented in CSSSWEFrameUserRegistration.

Invocable You can invoke FrameEventMethod*xxxxx* from server script and business services. You can also invoke this method through custom buttons and commands.

However, you cannot invoke this method from browser script.

User Properties 4

This section describes the user properties supported for customer use in Siebel applications.

User properties are object definitions that are added as children to an applet, business component, control, field, or list column to configure specialized behavior beyond what is configured in the parent object definition's properties. These user properties belong to the following Siebel object types:

- Applet
- Business Component
- Business Service
- Integration Component
- Integration Component Field
- Integration Object
- Virtual Business Component

Only supported user properties are documented. Expand the User Properties list in the Contents tab to see a list of links to descriptions of user properties.

Affiliated Account Id Field

This user property allows you to specify the name of the field that stores the Account Id of affiliated Contacts. It is used in Siebel Life Sciences applications to show affiliated contacts.

Syntax	The value for the Affiliated Account Id Field user property must be the name of a field on the business component.
Usage	If this user property is not defined, the application looks for the parent business component Account Id field (see "ParentBC Account Id Field" on page 264). You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.
Parent Object Type	Field
Functional Area	Contact

All Mode Sort

Parent Object Type Business Component

Description Standard Siebel application behavior is to override the sort specifications on an All view to force it to ORDER BY the standard user key. This user property determines whether or not the Siebel application overrides the sort specification and, if so, determines the sort (if any) that is applied to the business component for All or Manager views. The values are the following:

- **Normal.** Uses the business component-defined sort specifications. This setting also allows the user to run a PDQ (that incorporates a SORT).
- **TRUE.** Overrides the business component sort specifications and uses the U1 index (the standard user key). If the standard user key is defined on the primary extension table, especially for S_PARTY-based business components, the behavior reverts to Normal.
- **FALSE.** Removes all sorting.

NOTE: Some users have tried to set up the default sort order so that it sorts on an All or Manager view. Be aware that doing so may expose large quantities of data that in general should be sorted only by user keys.

If you choose to override this behavior using All Mode Sort, first consult your database administrator for guidance on how the sort by fields are being indexed.

You should also conduct careful performance testing on the view itself and on reports run against the view.

Functional Area SSE Service Request, Action, and Activity Plan Action

Always Enable Child [BusComp name]

Parent Object Type	Business Component
Description	Used in freezing Service Requests. Closing a service request does not prevent updates to the Customer Satisfaction business component, so you can survey your customers even if a service request has been resolved. The user property for the Service Request business component, named Always Enable Child: Customer Survey, allows you to enable and disable this behavior. You can disable the freeze behavior for other business components by adding user properties to the Service Request business component and substituting the business component name for the appropriate component. Unfreezing Service Requests - To unfreeze a closed service request and make the service request and its child business components accessible for additions and edits, change the status to Open.
Functional Area	SSE Service Request business component

ApplicationContextType

This user property specifies the application to use for generating proposal and presentation documents.

Syntax Values are:

- MSWord Specifies Microsoft Word for proposals.
- MSPpt Specifies Microsoft PowerPoint for presentations.

Usage You cannot inactivate or modify the values for this user property, nor can you create new instances of this user property.

Parent Object Type Applet

Functional Area Proposal and Presentation

Application Name

The Application Name user property specifies the name of the application.

Syntax	The value of this user property must be a valid application name.
Usage	You can inactivate and modify values for this user property. You can also create new instances of this user property as needed.
Parent Object Type	Business Component
Functional Area	CSSBCFINOppty

Aspect User Properties

Aspect user proprieties are optional.

Aspect (CSSBCBase)

This property provides a dynamic way to use the CSSBCBase class. When a particular aspect has been set from the applet level or CSSBCBase's default, CSSBCBase will change its behavior based on the aspect-related setting described in the user property:

- If Aspect BC ReadOnly: *Aspect* is defined, and the current aspect is *Aspect*, the current record will become read-only when the value of *Field Name* is Y.
- If Aspect Child BC ReadOnly: *Aspect* is defined, and the current aspect is *Aspect*, the current record's child business components will become read-only when the value of *Field Name* is Y.
- If Aspect BC NoInsert: *Aspect* is defined, and the current aspect is *Aspect*, the user cannot insert new records when the value of *Field Name* is Y.

  NOTE: To use Aspect BC NoInsert: *Aspect*, you need to make sure that the value [Field Name] (normally a calculated field) will not be changed when the user steps to a different record; otherwise runtime behavior will confuse the end user. Improper configuration will make inserts dependent on the state of current records, which is most likely not what you want.

- For a particular business component field, if Aspect Default Value: *Aspect* is defined, and the current aspect is *Aspect,* the predefault value for this field will be the expression defined in the user property value.

  For example, if an activity business component has a Planned field with a field user property Aspect Default Value: Planned defined that has a value of Timestamp(), the predefault value for the activity's planned start date and time will be Timestamp().

Aspect user properties to use with a parent object type of Business Component are listed in Table 8.

Table 8. Aspect User Properties for the Business Component Object Type

User Property Name	Value
Default Aspect	*Aspect*
Aspect BC ReadOnly: *Aspect*	*Field Name*
Aspect Child BC ReadOnly: *Aspect*	*Field Name*
Aspect BC NoInsert: *Aspect*	*Field Name*

Aspect user properties to use with a parent object type of Field are listed in Table 9.

Table 9. Aspect User Properties for the Field Object Type

User Property Name	Value
Aspect Default Value: Aspect	Expression

Aspect (CSSSWEFrameBase and CSSSWEFrameListBase)

If the user property View Aspect* is defined for the current view, set and pass the aspect name to the underlying (CSSBCBase) business component.

If View Aspect* is not defined for the current view and Default Aspect is defined, set and pass the aspect name to the underlying (CSSBCBase) business component.

Aspect user properties for use with a parent object type of Business Component are listed in Table 10.

Table 10. Aspect User Properties for the Business Component Object Type

User Property Name	Value
View Aspect: *View Name*	*Aspect Name*
View Aspect	*"View Name", "Aspect Name"*

Table 10. Aspect User Properties for the Business Component Object Type

User Property Name	Value
View Aspect 1	*"View Name", "Aspect Name"*
View Aspect 2	*"View Name", "Aspect Name"*
Default Aspect	*Aspect Name*

Associate: Completion Timeout (Client)

This user property specifies the maximum amount of time (in seconds) to wait (on the client) for a subordinate's forecast to complete before skipping the association and throwing an error.

Syntax	The value of the Associate: Completion Timeout (Client) user property is an integer greater than 0.
Usage	If you inactivate the Associate: Completion Timeout (Client) user property, the default value is used. You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Forecast

Associate: Completion Timeout (Server)

This user property specifies the maximum amount of time (in seconds) to wait (on the server) for a subordinate's forecast to complete before skipping the association and throwing an error.

Syntax	The value of the Associate: Completion Timeout (Server) user property is an integer greater than 0.
Usage	If you inactivate the Associate: Completion Timeout (Server) user property, the default value is used. You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Forecast

Associate: Sleep Time Between Attempts

This user property specifies the amount of time to wait (in seconds) after each attempt at checking the subordinate's forecast for completion before checking again.

Syntax	The value of the Associate: Sleep Time Between Attempts user property is an integer greater than 0.
Usage	If you inactivate the Associate: Sleep Time Between Attempts user property, the default value is used. You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Forecast

Association

Parent Object Type	Integration Component Field
Description	Indicates that the integration component is an association business component.
Functional Area	eBusiness Application Integration Test

AssocFieldName [Field Name]

Parent Object Type	Integration Component Field User
Description	Denotes the name of the business component field as it appears on the MVG business component.
Functional Area	eBusiness Application Integration Test

AutoAssignSearch

This user property specifies a search specification that is applied during reassignment in Siebel eAutomotive applications.

Syntax	The value for the AutoAssignSearch user property must be a valid search specification for an Opportunity (for example, [CloseOut Flg] = 'N').
Usage	You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Contact

AutoPopulateResponsibility

This user property specifies whether to automatically associate a responsibility with a new user when a record is created.

Syntax

Values:

- TRUE Automatically associates a responsibility with a new user when a record is created.

Usage

The responsibility to be associated is specified in the New Responsibility field for the current user (the user creating the new user record).

The AutoPopulateResponsibility user property requires that the responsibility MVF is named Responsibility.

This user property is ignored when the business component is used within the EAI or Siebel Adapter context.

You can inactivate and modify the values for this user property. You can also create new instances of this user property as needed.

Parent Object Type

Business Component

Functional Area

User

BAPIAdapterService

Parent Object Type	Business Service
Description	Name of the BAPI adapter business services. Values are eAI, SAP, BAPI Adapter.
Functional Area	eBusiness Application Integration Business Services

BatchSize

Parent Object Type Business Service

Description Sets the number of IDOCs to read from SAP in a single call. The integer value of this property depends on the available system resources in your environment. Extremely high values can cause performance degradation, given insufficient system resources. Default value is 3000.

Functional Area

BC eAuto Sales Step

This user property specifies the name of the business component for eAuto Opportunity Sales Step.

Syntax	The value of the BC eAuto Sales Step user property must be a business component name in the Opportunity business object.
Usage	You can inactivate or modify the values for this user property. You can also create new instances of this user property as needed.
Parent Object Type	Business Component
Functional Area	CSSBCFINOppty

BC eAuto Sales Step Admin

This user property specifies the name of the business component for eAuto Sales Step Admin.

Syntax	The value of the BC eAuto Sales Step Admin user property must be a business component name in the eAuto Sales Step Admin business object.
Usage	You can inactivate or modify the values for this user property. You can also create new instances of this user property as needed.
Parent Object Type	Business Component
Functional Area	CSSBCFINOppty

BC Opportunity

This user property allows you to specify the name of the Opportunity business component to be used during reassignment in Siebel eAutomotive applications.

Syntax	The value of the BC Opportunity user property must be the name of an Opportunity business component.
Usage	You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Contact

BC Position

This user property allows you to specify the name of the Position business component to be used when automatically creating an Opportunity in Siebel eAutomotive applications.

Syntax	The value of the BC Position user property must be the name of a Position business component.
Usage	You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Contact

BC Read Only Field

Parent Object Type	Business Component
Description	Specifies a TRUE/FALSE field in the record that, when TRUE, causes the current record to become read-only. Setting the Admin Mode Flag property to TRUE circumvents a Read Only field defined at the business component object type user property.
Functional Area	Data-Driven Access Control (Time Sheet)
Where Documented	*Siebel Tools Reference*

BO eAuto Sales Step Admin

This user property specifies the name of the business object for eAuto Sales Step Admin. It is used to define the view for setting Sales Steps.

Syntax	The value of the BO eAuto Sales Step Admin user property must be a valid business object name.
Usage	You can inactivate or modify the values for this user property. You can also create new instances of this user property as needed.
Parent Object Type	Business Component
Functional Area	CSSBCFINOppty

Calc Actual OnWriteRecord

This user property specifies whether or not to execute Actual Number calculation when a record is written.

Syntax Values:

- Y Indicates that the Actual Number will be calculated when a record is written.
- N Indicates that the Actual Number is not calculated.

Usage You can inactivate or modify the values for this user property. You can also create new instances of this user property as needed.

Parent Object Type Business Component

Functional Area CSSBCFINOppty

Contact-Activity BC Name

This user property allows you to specify the name of the business component to be used when reassigning Activities in Siebel eAutomotive applications.

Syntax	The value of the Contact-Activity BC Name user property must be the name of a business component (for example, Action).
Usage	You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Contact

Contact MVG PreDefault Expression

The predefault contact record specified in this user property is added to the Contact multivalue field of the Action business component.

Syntax

```
'[Parent Buscomp1 Name].[Parent Buscomp1 Field Name]', '[Parent Buscomp2 Name].[Parent Buscomp2 Field Name]'
```

Alternatively, you can use "Parent: " at the beginning of the value as the preconfigured user property for Action business component. For example:

```
Parent: 'Contact Id', 'Service Request.Contact Id', 'Service Agreement.Contact Person Id'
```

Usage You can inactivate and modify the values for this user property. You can also create new instances of this user property.

Parent Object Type Business Component

Functional Area Activity

Contact-Opportunity BC Name

This user property allows you to specify the name of the business component to be used for Opportunity reassignment in Siebel eAutomotive applications.

Syntax	The value of the Contact-Opportunity BC Name user property must be the name of a business component (for example, Opportunity).
Usage	You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Contact

Contact Relationship Type

This user property specifies a list of relationship types that indicate the contact has a line of influence.

Syntax

The value of the Contact Relationship Type user property consists of one or more relationship types. Multiple relationship types should be separated by a comma and a space (for example, Influencer, TAS Influencer). These types typically come from the CONTACT_RELATIONSHIP_TYPE LOV.

Usage

The value of this user property populates the RELATION_TYPE_CD column of the S_CONTACT_REL table (the intersection table between Contact and Contact Relationship business components) when one box in an organization chart is dragged and dropped onto another box while the Ctrl key is pressed.

For example, when Box A in an organization chart is dragged and dropped on Box B while the Ctrl key is pressed, CONTACT_ID will be the row id of Box A and REL_CONTACT_ID will be the row id of Box B.

You can modify the values for this user property. However, you cannot inactivate or create new instances of this user property.

Parent Object Type

Applet

Functional Area

Organization Chart

Copy Contact

Parent Object Type Business Component

Description This user property is currently defined for DBM Campaign business component. The values are TRUE or FALSE. If this property is set to TRUE when the status of a campaign is changed from Planned to Active, it will copy all a contact's associations in the cell for that campaign, over to that campaign.

Functional Area DBM Campaigns

Credit Card User Properties

The following user properties store information that Siebel applications, especially Siebel eSales, use in the Mod 10 (LUHN) algorithm for credit card validation:

- Credit Card Expired Month
- Credit Card Expired Year
- Credit Card Number
- Credit Card Type

These user properties are valid only for business components based on the CSSBCBase class or on classes that inherit from or are based on CSSBCBase, such as CSSBCQuote.

For example, you might want to validate credit card information in the CUT Billing Account Payment Information business component. This will not work for Siebel 7 releases before 7.0.4 because this business component was based on the CSSBusComp class in earlier releases.

For more information on payment validation, see *Siebel eSales Administration Guide*.

NOTE: All four of the fields defined by these user properties must have data for validation to occur.

Credit Card Expired Month

Parent Object Type	Business Component
Description	Expiration month
Functional Area	Personal Payment Profile, Quote

Credit Card Expired Year

Parent Object Type	Business Component

Description Expiration year
Functional Area Personal Payment Profile, Quote

Credit Card Number

Parent Object Type Business Component
Description Account number
Functional Area Personal Payment Profile, Quote

Credit Card Type

Parent Object Type Business Component
Description Account type
Functional Area Personal Payment Profile, Quote

Credit Check

This user property enables and disables the feature of Credit Check during verification.

Syntax

Values:

- Y Perform credit check during Verify.
- N Do not perform credit check during Verify.

Usage

You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.

Parent Object Type

Business Component

Functional Area

Quote

Credit Check Workflow

This user property specifies the name of the workflow to use for Credit Check when it is enabled (for example, Credit Check – Quotes).

Usage

If the Credit Check user property is set to Y, the Credit Check Workflow user property is required. You can inactivate this user property only if the Credit Check user property is set to N.

You can modify the value for this user property. However, you cannot create new instances of it.

Parent Object Type

Business Component

Functional Area

Quote

DataCleansing Field *n*

Parent Object Type	Business Component
Description	Specifies a correspondence between a field name in the Siebel Firstlogic Connector and a field name in the Siebel application. Used for Siebel Data Quality, which performs address verification, name and address standardization, and duplicate record identification, in real-time and batch modes. Value: "[Firstlogic Field Name]", "[Siebel Field Name]" **NOTE:** All data quality user properties require components from Firstlogic Corporation.
Functional Area	Data Quality

DataCleansing Type

Parent Object Type Business Component

Description Specifies to the Siebel Firstlogic Connector what kind of data is being validated in the Data Cleansing Field.

The data types are Contact, Account, or Address. *Contact* indicates that the data consists of person name records; *Account* indicates that data consists of business or office name records; *Address* indicates that data consists of postal addresses. All types have capitalization validated.

Data cleansing operates differently on each of these types. For example, business components with Address cleansing have reconciliation performed between address fields and the ZIP (Postal Code) field.

Functional Area Data Quality

DataSourceBuscompName

This user property allows you to specify the name of the business component in the business object that represents the parent business component.

When the Document Server receives a request to generate a proposal or presentation, it tries to restore the request context before processing the request. It does this so that business components from which the Proposal business component retrieves data are positioned on the proper record set.

Syntax

The value for the DataSourceBuscompName user property is the name of a business component in the current business object.

Usage

The specified business component will be positioned on the correct record first (the record under which the proposal request was submitted) to make sure other business components return proper records based on the link specification, from which the Proposal business component fetches.

In most cases, this user property does not need to be specified because it defaults to the parent business component of the Proposal business component, which is typically the parent business component for related business components.

You need to specify this user property only when the data to be retrieved for various sections in the Proposal is from business components whose parent business component defined in the current business object is not the same as the parent of the Proposal business component. This user property causes data fetched from those business components to be restricted by the link to the business component defined in the user property rather than the default value of the Proposal's parent.

You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.

Parent Object Type

Applet

Functional Area

Proposal and Presentation

DB2 Optimization Level

Parent Object Type Business Component

Description Currently, the DB2 connector uses an optimization level of 0 for optimizing client SQL statements. In some cases, certain SQL statements may perform suboptimally using optimization level 0. This user property allows an implementor to change the optimization level of all of the SQL statements produced by the given business component.

Because this setting affects the whole business component, changing it may adversely affect the performance of other SQL statements produced by the same business component. Before using this option, it is extremely important to analyze slow performing SQL statements, and then consider all of the options available for tuning the statement. Some of these tuning methods are as follows:

1. Make sure there is an index which addresses the needs of both the where conditions on the driving tables of the query, and order by clause.
2. If the business component has been customized, simplify the customization.
3. Remove one or more columns from the order by clause.
4. Change the optimization level.

The DB2 Optimization Level user property can only be used in expert mode.

The first step to analyzing performance is to start the Siebel client with the `/s <filename>` option to log all of the SQL statements. This log file shows the time spent executing each SQL statement. Identify the statements that are slow.

NOTE: The DB2-specific SQL generator adds an "optimize for 1 row" clause to the end of the SQL statement. This clause will not show up in the SQL log.

Check the business component that produced the SQL statement to determine if there is already a DB2 Optimization Level User Property. If there is, then use that optimization level to explain the SQL of the slow query. Otherwise, use optimization level 0.

Paste the suspected slow query into one of the explain utilities, add the "optimize for 1 row" clause, and set the optimization level to the appropriate value. Generate a query plan for the statement and analyze it. If you conclude that changing the optimization level is the best approach to increasing the performance of the query, then re-explain it using a different optimization level. Trying level 3 first is recommended, because some complex queries that perform poorly with optimization level 0 often perform better with optimization level 3.

If the new optimization level solves the slow query performance, then add the user property name to the business component.

DeDup Token Value

Parent Object Type Business Component

Description Modifies the dedup token calculation expression for a business component. Changes the calculation expression to adjust the filtering that generates the candidate set. A larger or more restricted candidate set may be desired, or a field may need to be added to or removed from the expression. Calculation expressions for dedup tokens follow the same syntax rules as calculated fields.

Calculation expressions for Dedup tokens follow the same syntax rules as calculated Fields. If the calculation value is changed, you will need to regenerate the token values in the database for existing data. This is accomplished by running deduplication in batch mode.

The Dedup token is generated based on the following calculations.

Contacts: Dedup token consists of a concatenated string of the cleansed five-digit ZIP Code of the account address, the first letter of the cleansed account name, and the first letter of the cleansed last name. The calculation expression is

```
IfNull (Left ([Postal Code], 5), "_") + IfNull (Left
([Account],1), "_") + IfNull (Left ([Last Name],
1), "_")
```

Accounts and Sub-Accounts: Dedup token consists of a concatenated string of the cleansed five-digit ZIP Code of the account address, the first letter of the cleansed account name, and the first nonblank nonnumeric character of the cleansed street name. The calculation expression is

```
IfNull (Left ([Postal Code], 5), "_") + IfNull (Left
([Name], 1),"_") + IfNull (Mid ([Street Address],
FindNoneOf ([Street Address], "1234567890 "), 1),
"_").
```

Prospects: Dedup token consists of a concatenated string of the cleansed 5-digit ZIP Code of the account address, the first letter of the cleansed account name, and the first letter of the cleansed last name. The calculation expression is

```
IfNull (Left ([Postal Code], 5), "_") + IfNull (Left
([Primary Account Name], 1), "_") + IfNull (Left
([Last Name], 1), "_")
```

Functional Area Data Quality

Deduplication User Properties

Deduplication detects possible matches to records in specified business components during record creation and update. The matching process begins with the Dedup token, which is an identifier calculated for each account, contact, or prospect in the database as well as the newly created or modified record. Based on the value of the Dedup token, the Siebel application passes to the data quality matching engine a short list of prequalified possible matches for further refinement.

Deduplication, like data cleansing, is configured in two business component user properties, but also affects certain views and applets. The deduplication feature is disabled or enabled for the application through settings in the .cfg file. Once turned on at the application level, deduplication can be turned off for a specific business component by deactivating all of the child user properties (by setting the Inactive property to TRUE). Deduplication cannot be turned off for individual records. If you configure a business component for deduplication, it must also be configured for data cleansing and data cleansing must be turned on. (The reverse is not necessarily true; you can configure data cleansing for a business component without configuring deduplication.)

Data deduplication works only on applets based on the CSSFrameBase and CSSFrameListBase classes, and classes derived from these. Data cleansing works for applets based on any class.

NOTE: Components from Firstlogic Corporation must be installed for this functionality to work.

DeDuplication CFG File

Parent Object Type Business Component

Description	Firstlogic CFG file used for Account deduplication.
Functional Area	Data Cleansing

DeDuplication Field *n*

Parent Object Type	Business Component
Description	Sets up a correspondence between a Firstlogic Connector data field and a Siebel business component data field. The value consists of a pair of quoted strings in double quotes, separated by a comma, with the first string identifying the Firstlogic field name and the second string identifying the Siebel name. The set of fields mapped in DeDuplication Field user properties is the set of fields that is passed in records in the candidate set to the Firstlogic Connector. The candidate set consists of records with a dedup token exactly or partially matching the calculated dedup token of the record being added or modified, and therefore representing possible duplicates.
Functional Area	Data Quality

Deduplication has a set of numbered user properties that set up correspondences between Firstlogic fields and Fields in Business Components. These field mapping properties have names of the form DeDuplication Field *n*, where *n* is an integer value (for example, DeDuplication Field 7). The syntax for the Value property in a DeDuplication Field user property is the same as for a DataCleansing Field user property: the value consists of a pair of quoted strings in double quotes, separated by a comma, with the first string identifying the Firstlogic field name and the second string identifying the Siebel name. The set of fields mapped in DeDuplication Field user property is the set of fields that is passed in records in the candidate set to the Firstlogic Connector. The candidate set consists of records with a dedup token exactly or partially matching the calculated dedup token of the record being added or modified, and therefore representing possible duplicates.

The Prospects business component requires additional user property configuration beyond that required for Contacts and Accounts. Prospects share name processing capabilities in the Firstlogic Connector with Contacts, and Contact data (rather than Prospect data) is assumed by the system to be present. In order to specify that Prospect data is being processed, two additional two user properties must be added, DeDuplication Results business component and DeDuplication Results applet.

DeDuplication Results

Parent Object Type Business Component

Description This user property name is added to the Prospects business component upon which deduplication is being performed. The value in the user property name is the name of the business component that will hold the returned data, typically DeDuplication Results (Prospect).

Functional Area Data Quality

DeDuplication Results Applet

Parent Object Type Applet

Description This user property name is added to the Prospects applet from which deduplication is invoked. The value in the user property name should be the name of the pick applet used to prompt the user to resolve duplicates, typically DeDuplication Results (Prospect) List Applet.

Functional Area Data Quality

Deep Copy *n*

Parent Object Type Business Component

Description Allows child business components and respective child business components to be copied automatically when selecting the Copy option. Normally, Copy option only copies one level. This feature allows multiple levels to be copied like a cascade copy.

To use Deep Copy, do the following:

1. In the parent business component, create a user property for each child business component to be included in the deep copy. The child business component user properties are:

   Name: Deep Copy 1
   Value: [Child BusComp Name]

   Name: Deep Copy 2
   Value: [Child BusComp Name]

2. Add a multivalue link in the parent business component for each child business component.
3. Create a multivalue field in the parent business component from each child business component.
4. Set the Field No Copy attribute in the multivalue link to TRUE or an SQL error occurrence: "A Duplicate Record Exists."

Each business component in the Deep Copy chain takes care of its own children. So the parent business component will have Deep Copy properties for each of its direct children, and each child business component will have Deep Copy properties for each of the relevant grandchildren.

There is an analogous Deep Delete user property to do a deep cascade delete. Typically, use Deep Copy and Deep Delete together.

Functional Area CSSBCBase

Deep Delete *n*

Parent Object Type Business Component

Description Normally, the Delete option only deletes one level. Deep Delete allows child business components and their respective child business components to be deleted automatically when selecting the Delete option. This feature allows multiple levels to be deleted like a cascade delete.

1 Create a user property for each child business component to be included in the deep delete. The child business component user properties are:

   Name: Deep Delete 1
   Value: [Child BusComp Name]

   Name: Deep Delete 2
   Value: [Child BusComp Name]

2 Add a multivalue link for each child business component. Set the No Delete user property value to:

   TRUE: allows deep delete for child business component
   FALSE: does not allow deep delete for child business component

Create a multivalue field in the parent business component, using the multivalue link. This field is usually not displayed on the screen but needs to be present on the business component.

This is analogous to the Deep Copy user property. Use Deep Delete to do a deep cascade delete. Typically, use Deep Copy and Deep Delete together.

Functional Area CSSBCBase

Default Display Field

This user property allows you to specify one or more fields to be displayed in a given box of an organization chart. If no field is specified, the following fields are displayed:

- Full Name
- Job Title
- Work Phone#

Syntax	The value of the Default Display Field user property consists of one or more business component field names. If you define multiple fields, use a comma and a space to separate fields (for example, Full Name, Job Title, Work Phone#).
Usage	You can modify the values for this user property. However, you cannot inactivate or create new instances of this user property.
Parent Object Type	Applet
Functional Area	Organization Chart

DisableNewRecord

This user property allows you to prevent NewRecord from being invoked on the current applet in the specified Siebel application.

Syntax

The value for the DisableNewRecord user property is an application name.

The value must match the application name exactly.

Usage

If the specified application name is the same as the current running application name, NewRecord is disabled on this applet. For example, a value of *Siebel Sales Enterprise* will disable NewRecord on the applet when running within Siebel Sales Enterprise application.

You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.

Parent Object Type

Applet

Functional Area

Proposals and Presentations

Display Mask Char

Parent Object Type Field

Description Masks credit card numbers. Create a calculated field with no calculation and with this user property and Encrypt Source Field user property as child objects. Enter a value of 9999999999999999 into the CC stored field and xxxxxxxxxxxxx9999 appears.

DocumentContextType

This user property allows you to specify the context of a proposal. The specified value is compared with the value of the Category field in the Proposal Template, so that only templates within the same category are used in this applet.

Syntax	Value is a string that defines a proposal template category. If the value specified matches one of the values in PROPOSAL_TEMPLATE_TYPE LOV, the templates with the same category value can be used in this applet (they show up in the Template Picklist). For example, a value of *Account Proposal* will filter proposal templates so that only proposal templates with the Account Proposal defined for the Category field will be available for the user to choose in the Template picklist in this applet. The value must exactly match a value in PROPOSAL_TEMPLATE_TYPE LOV.
Usage	You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.
Parent Object Type	Applet
Functional Area	Proposal and Presentation

Drilldown Visibility

Parent Object Type Applet

Description Defines the visibility applied to the new view during a single record drill down. This user property is currently used in the vanilla Siebel product. However, its function has been superseded by the Visibility Type attribute of the drilldown object.

eAuto Enable Create Sales Step

This user property specifies whether or not to populate Opportunity Sales Steps for a Siebel eAutomotive application.

Syntax

The value of the eAuto Enable Create Sales Step user property consists of two quoted parameters separated by a comma and a space, as follows:

```
"ApplicationName", "bPopulate"
```

where *ApplicationName* specifies the name of the Application and *bPopulate* is a Boolean (Y/N) value indicating whether or not to populate the Opportunity Sales Steps. For example, the following value for the eAuto Enable Create Sales Step user property would cause the Opportunity Sales Step to be populated for Siebel eDealer:

```
"Siebel eDealer", "Y"
```

Usage

You can inactivate or modify the values for this user property. You can also create new instances of this user property as needed.

Parent Object Type

Business Component

Functional Area

CSSBCFINOppty

eAuto Status Field Name

This user property specifies the name of the field that stores the Activity status. It is used during reassignment in Siebel eAutomotive applications.

Syntax	The value for the eAuto Status Field Name user property must be the name of a field in the business component.
Usage	You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Contact

eAuto Status Field Value

This user property allows you to specify the value criteria of the Activity status to be used during reassignment in Siebel eAutomotive applications.

Usage	You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Contacts

eGanttChart Busy Free Time Applet User Properties

Table 11 contains the user properties for eGanttChart Busy Free Time Applet:

Table 11. eGanttChart Busy Free Time Applet User Properties

Name	Value	Comments
Major Time Unit	Day	Major time unit in the X-axis of the Gantt chart. The possible values are Day, Week, Month, and Year.
Minor Time Unit	Hour	Minor time unit in the X-axis of the Gantt chart. The possible values are Hour, Day, Week, and Month. The possible combinations of (major,minor) time units are (Day,Hour), (Week,Day), (Month,Day), (Month,Week), and (Year,Month).
X-BC	Action (Busy Free Time)	Business component for the X-axis of the Gantt chart.
X-Color Field	Priority	Cells of the Gantt chart can be colored differently. This user property specifies which field in the X business component determines the coloring. For the field that you choose, make sure there is a corresponding LOV defined. See X-Color LOV Name for details.
X-Color LOV Name	ACTIVITY_PRIORITY	LOV that defines the possible values of the field chosen for X-Color Field. From this LOV, you can determine the Language Independent Code (LIC) of the field value.
X-LOV Map	#1-ASAP#2-High# 3-Medium#	Defines the LIC of the LOV values that can be mapped to colors.
X-Color Map	#GanttChartRed #GanttChartBlue #GanttChartGreen#	Defines the colors of the different values of the LIC. The values must be in the same order as in X-LOV Map. For example, 2-High maps to GanttChartBlue. The values of X-Color Map are the style sheet classes defined in Gantt.css.
X-Date InvokeMethod	SetGridBeginEndDate	Maps to an internal X-business component method to calculate the date-time range for the X-business component data. You should not modify this.
X-Display Constraint Map	08:00:00#17:00:00	Applies only when the minor time unit is Hour. Instead of displaying a 24-hour interval in the Gantt chart, this user property defines the lower and upper bounds of the time displayed in the Gantt chart.

Table 11. eGanttChart Busy Free Time Applet User Properties

Name	Value	Comments
X-Display Duration	30	Specifies the time increment for each cell in the Gantt chart. The unit is Minutes.
X-DrillDown Field	Description	Specifies on which field in the X business component you can drill down. A corresponding Drilldown Object will need to configured for the applet in Siebel Tools.
X-End DateTime Field	Planned Completion	Field in the X business component that specifies the end of the datetime range.
X-Join Field	Primary Owner Id	Field in the X business component that links it to the Y business component. This is used in a search specification if X-Join InvokeMethod is not specified.
X-Join InvokeMethod	SetEmployeeList	Maps to an internal X business component method to link the Y and X business components. You should not modify this.
X-Num Slots	3	Specifies the number of “slots” for a given cell. For example, you might have conflicting activities that start and end at the same time. If X-Num Slots is 3, the cell that represents this time range can be split into a maximum of three slots to contain the conflicting activities.
X-Sort Spec	Owner Last Name	Specifies the sort specification for the X business component.
X-Start DateTime Field	Planned	Field in the X business component that specifies the start of the datetime range.
X-Tooltips 1	Planned	First field that is displayed in the tooltip.
X-Tooltips 2	Planned Completion	Second field that is displayed in the tooltip. User properties can be defined to extend this to X-Tooltips *n*.
Y-BC	Employee (MM)	Business component for the Y-axis of the Gantt chart.
Y-BC ViewSet Size	7	Maximum number of records for the Y-axis.

Table 11. eGanttChart Busy Free Time Applet User Properties

Name	Value	Comments
Y-Constraint		Search specification for the Y business component. **NOTE:** By default, the Y business component in the base Gantt chart class does not have any search specification defined. If it is not explicitly define here the Y business component will be constrained by a link in a parent-child relationship.
Y-DrillDown Field	Id	Source field for the Y drilldown.
Y-DrillDown View	All Employees across Organizations	Destination view of the Y drilldown. This can be modified.
Y-Join Field	Id	Field in the Y business component that links it to both the X and Z business components.
Y-Label	Employee	Label for the Y-axis.
Y-Legend	Full Name	Field in the Y business component that is displayed in the Gantt chart.
Y-SortSpec	Last Name	Sort specification for the Y business component.
Z-BC	Schedule	Business component for the Z-axis of the Gantt chart. Acts as a layer painted on the Gantt chart before the X-axis is constructed. For example, each employee has a working schedule of Monday through Friday, 8:00 to 17:00. The Gantt chart is painted white where there is a schedule and gray where there is no schedule. The activity cells are drawn on top of this schedule layer. The Z business component determines the state of a cell: ■ GanttStateNone: No schedule ■ GanttStateOff: A schedule exists ■ GanttStateOn: Coloring of the cell controlled by X-Color Map
Z-Date InvokeMethod	SetGridBeginEndDate Time	Maps to an internal Z business component method to calculate the datetime range for the Z business component data. You should not modify this.

Table 11. eGanttChart Busy Free Time Applet User Properties

Name	Value	Comments
Z-End DateTime Field	End DateTime	Field in the Z business component that specifies the end of the datetime range.
Z-Join Field	Employee Id	Field in the Z business component that links to the Y business component. This is used as a search specification if Z-Join InvokeMethod is not specified.
Z-Join InvokeMethod	SetEmployeeList	Maps to an internal Z business component method to link the Y and Z business components together. You should not modify this.
Z-Start DateTime Field	Start DateTime	Field in the Z business component that specifies the start of the datetime range.

Email Activity Accepted Status Code

This user property specifies the code in the EVENT_STATUS_LOV that corresponds to completely sent outbound email activity.

Syntax Language-independent code listed in the EVENT_STATUS_LOV. Example value:

- Done

Usage This user property is used by eMail Response client to set the status of outbound email activities that have been completed.

Do not inactivate this user property. You can modify the values for this user property. However, you cannot create new instances of this user property.

Parent Object Type Business Component

Functional Area Activity

Email Activity New Status Code

This user property specifies the code in the EVENT_STATUS_LOV that corresponds to unsent/unprocessed outbound email activity.

Syntax Language-independent code listed in the EVENT_STATUS_LOV. Example value:

- Not Started

Usage This user property is used by eMail Response client to set the status of new outbound email activities that have not yet been sent.

Do not inactivate this user property. You can modify the values for this user property. However, you cannot create new instances of this user property.

Parent Object Type Business Component

Functional Area Activity

Email Activity Rejected Status Code

This user property specifies the code in the EVENT_STATUS_LOV that corresponds to unsuccessful outbound email activity.

Syntax Language-independent code listed in the EVENT_STATUS_LOV. Example value:

- Cancelled

Usage This user property is used by eMail Response client to set the status of outbound email activities that cannot be processed because there is a problem when sending out the email.

Do not inactivate this user property. You can modify the values for this user property. However, you cannot create new instances of this user property.

Parent Object Type Business Component

Functional Area Activity

Email Activity Sent Status Code

This user property specifies the code in the EVENT_STATUS_LOV that corresponds to in-progress outbound email activity.

Syntax

Language-independent code listed in the EVENT_STATUS_LOV. Example value:

- Queued

Usage

This user property is used by the eMail Response client to set the status of outbound email activities that are in process and waiting for Communications Outbound Manager or Email Manager to send out the email.

Do not inactivate this user property. You can modify the values for this user property. However, you cannot create new instances of this user property.

Parent Object Type

Business Component

Functional Area

Activity

Email Manager Compatibility Mode

This user property specifies whether or not to use Mail Manager to send emails. It is used to support Mail Manager compatibility for version 6.x applications.

Syntax Value:

- Y use Mail Manager to send emails

Usage If the value is Y, then the business component uses Mail Manager to sent out emails, otherwise it uses Outbound Communications Manager.

You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.

Parent Object Type Business Component

Functional Area Activity

Enable Create Opportunity

This user property allows you to specify whether or not to permit automatic creation of Opportunities in Siebel eAutomotive applications.

Syntax Values:

- Y Indicates that opportunities can be automatically created.
- N Indicates that opportunities cannot be automatically created.

Usage You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.

Parent Object Type Business Component

Functional Area Contact

Encryption User Properties

Encryption can be controlled using the following business component field user properties: Encrypted, Encrypt Key Field, Encrypt Service Name, Encrypt ReadOnly Field, and Encrypt Source Field. For more information on encryption keys and how they are managed, see *Siebel Server Administration Guide*.

A field is encrypted by setting the ID, encryption flag, and encryption service. Siebel applications come preconfigured with two business services that you can use to encrypt data fields: the Standard Encryptor, based on a proprietary algorithm, and the RC2 Encryptor, based on RSA encryption.

NOTE: To use the RSA encryptor service on a field that was previously unencrypted or that was encrypted using the Standard Encryptor, you need to run upgrade scripts. For more information, see the upgrade guide for the operating system you are using.

When encryption is turned on, data written to the field will be in the encrypted format and data read from the field will be decrypted. Therefore, all business component fields that are mapped to the same database column must also have encryption turned on with consistent user property specifications.

The ID, encryption flag, and encryption service properties are described in the following four sections.

To turn on encryption

1. Select the business component containing the field.
2. In the field user properties, set the Encrypt Key Field property and Encrypt Service Name property to Active.
3. Set the Encrypted property to Y.

NOTE: In the Order Entry -- Orders, Quote, and Agreements business components screens, it may be to desirable turn off the encryption in the Credit Card field so that the user can see what was typed.

To turn off encryption

1. Select the business component containing the field.
2. In the field user properties, set the Encrypt Key Field property and Encrypt Service Name property to Inactive.
3. Set the Encrypted property to N.

See Also

- "Encrypted" on page 229
- "Encrypt Key Field" on page 229
- "Encrypt Service Name" on page 229
- "Encrypt ReadOnly Field" on page 230
- "Encrypt Source Field" on page 230

Encrypted

Parent Object Type Field

Description Denotes whether or not field is encrypted. Boolean values Y or N.

Functional Area

Encrypt Key Field

Parent Object Type Field

Description The default is ID. When using the Standard Encryptor, use ID. When using the RC2 Encryptor, designate the field on the business component where the encryption key index is stored.

Functional Area

Encrypt Service Name

Parent Object Type Field

Description The default for the business service is Standard Encryptor. Valid values are “Standard Encryptor” and “RC2 Encryptor.”

Functional Area

Encrypt ReadOnly Field

Parent Object Type Field

Description References a calculated field that has the Calculated Value property left blank. This user property must be configured on the field to be encrypted. Otherwise, data will not be written to the database. See the Credit Card Number field in the Order Entry - Orders business component for an example.

Functional Area

Encrypt Source Field

Parent Object Type Field

Description The Siebel Application field to be encrypted. Use the format Contact [*field name*].

Functional Area

Extended Quantity Field

This user property is defined in the line item business component of a quote (for example, Quote Item).

Syntax	The value for Extended Quantity Field is the field name for the Extended Quantity in the line item (for example, Extended Quantity Requested).
Usage	Do not inactivate this user property, or create new instances of it. However, you can modify its value.
Parent Object Type	Business Component
Functional Area	Quote

Field OutputCol: Closed Opportunity Count

Parent Object Type Virtual Business Component

Description The output column of a Sagent Plan that can be referenced from within a business component. For example, “ClsdOptyCnt” is one of the output values that a Sagent Plan produces.

NOTE: Sagent is no longer supported after the Siebel 6.0 release.

Functional Area

Field Part: Acct Emp Size

Parent Object Type Virtual Business Component

Description References any column in a star schema from a Virtual Business Component. For example, in [[Part:M_PERSON_STAR.W_ACCNT_DM.NUM_EMPLOYEES_CAT]]. The NUM_EMPLOTEES_CAT column of W_ACCNT_DM table, belonging to M_PERSON_STAR meta view as defined in Sagent repository, is referenced.

NOTE: Sagent is no longer supported after the Siebel 6.0 release.

Functional Area

Field QueryOnly: Julian Month

Parent Object Type Virtual Business Component

Description Defines the query-only columns that are sent to Sagent. Boolean: TRUE or FALSE.

NOTE: Sagent is no longer supported after the Siebel 6.0 release.

Functional Area

Field Read Only Field: [*fieldname*]

Parent Object Type	Business Component
Description	Sets specific fields in a business component to be read-only. The Value property contains a field that is Boolean. When TRUE, the field specified by [fieldname] in the current record is read-only.
Functional Area	Data Driven Access
Where Documented	*Siebel Tools Reference*

Field SqlQuery: Product ID

Parent Object Type Virtual Business Component

Description If the Sagent plan has one or more SQL query, you can refer to the columns in the SELECT clause of that plan by this property. For example, Product ID is sent in as the input value for the select clause of SQL Query.

NOTE: Sagent is no longer supported after the Siebel 6.0 release.

Functional Area

FileMustExist

Parent Object Type	Business Component
Description	This is a TRUE/FALSE value indicating whether or not the user can enter the name of a file to be provided later. Typically set to TRUE, indicating that the file must already exist in order to add it as an attachment.
Functional Area	Asset Management
Where Documented	*Siebel Tools Reference*

FINS Query Mode Disabled Method *n*

This user property allows you to specify a method to be disabled when the applet is in query mode.

Syntax The value of the FINS Query Mode Disabled Method user property is the name of a method.

Usage When the current applet is in query mode, the specified method is disabled.

You can create additional instances of this user property as needed. If you have more than one instance of this user property for an applet, they are executed sequentially by number (for example, FINS Query Mode Disabled Method 1, then FINS Query Mode Disabled Method 2, and so on).

You can also inactive or modify the values for this user property.

Parent Object Type Applet

Functional Area CSSSWEFrameFINApplication

Forecast Analysis BC

This user property specifies the business component whose records are displayed in the Analysis view of the child applet when parent records are selected for comparison.

Syntax	The value of the Forecast Analysis BC user property is a business component name (for example, Forecast 2000 – Forecast Item Detail Flat).
Usage	You can modify the values for this user property. However, you cannot inactivate or create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Forecast

Forecast Rollup

This user property specifies a named search for rollup of forecasts. When activated, this search specification is applied on the Forecast Detail business component during rollup.

Syntax The value of the Forecast Rollup user property must be a valid search specification.

Usage Field names must be contained in square brackets. For example, the following value returns forecast details owned by the current user and his or her subordinates' records that roll into the current user forecast.

```
[Link Type] =
LookupValue('FCST_FCSTITEM_LINK_TYPE','Own
Item') OR [Link Type] =
LookupValue('FCST_FCSTITEM_LINK_TYPE','Item')
```

Additionally, you can use the Forecast 2000 -- Forecast Series business component to further restrict the rollup search specification on a per series basis.

You can inactivate or modify the values for this user property. However, you cannot create new instances of this user property.

Parent Object Type Business Component

Functional Area Forecast

Group Visibility

Parent Object Type	Business Component
Description	If set to TRUE, “group + team” visibility will be applied.
Functional Area	Campaign

Group Visibility Only

Parent Object Type	Business Component
Description	If set to TRUE, only group visibility will be applied. If you set this to TRUE, set Group Visibility to FALSE. You might also want to inactivate Buscomp View Mode, so that it will not do an inner join to the S_SRC_POSTN table.
Functional Area	Campaign

Logical Message Type

Parent Object Type Integration Object

Description When used with the SAP product, this user property is ignored by the SAP Connector in the inbound direction (receiving IDOCs from SAP). In the outbound direction (sending IDOCs to SAP), the SAP Connector populates the control segment MESTYP field using the value of this user property before sending each IDOC to SAP.

Values: Logical message type corresponding to the IDOC. For example, DEBMAS or MATMAS.

Functional Area

Maintain Master Account

This user property allows you to specify whether or not the Master Account Id in an Account hierarchy should be maintained.

Syntax Values:

- Y Indicates that the Master Account Id will be maintained.
- N Indicates that the Master Account Id will not be maintained.

Usage If you change the value of this user property from its default (Y), the Master Account Id in the Account hierarchy will not be maintained.

You cannot inactivate or create new instances of this user property.

Parent Object Type Business Component

Functional Area Account

Manager List Mode

Parent Object Type Business Component

Description In a manager view, see the records for all people who report to a manager, not only the primaries.

Values: Primary (the default) or Team.
When set to Team, it performs a subquery for the My Team's views to retrieve the accounts, opportunities, and so on, where the reps are anywhere on the team, not just the primary.

Performance is slower but yields more data.

When using Manager List Mode, you must also inactivate the Primary docking rules and activate the Team docking rules:

- Opportunities. Activate rule #13.
- Accounts. Activate rule #18.
- Contacts. Activate rule #24.

Functional Area eBusiness Application Integration Test

Master Account Field

This user property specifies the name of the field that stores the Master Account Id.

Syntax	The value of the Master Account Field user property must be the name of a field on the business component.
Usage	The CSSBCAccountSIS class uses the value of the field specified in this user property as the Master Account Id for the record. It is not recommended that you change the value of this user property from its default. You cannot inactivate or create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Account

MVG

Parent Object Type	Integration Component
Description	When used with the SAP product, this user property indicates that the integration component is a MVG business component. Boolean: Y or N.
Functional Area	eBusiness Application Integration Test

MVGFieldName [*field name*]

Parent Object Type	Integration Component Field
Description	Denotes the name of the business component field that maps to this field from the association MVG business component.
Functional Area	eBusiness Application Integration Test

MVG Set Primary Mode

Parent Object Type Business Component

Description Setting this user property to All allows the Primary team member to be altered by someone other than the Manager or Siebel Administrator. The default behavior is Manager.

Only Siebel Administrators (in Admin mode) and Managers (in Manager view mode) have the ability to change the Primary team member on opportunities, accounts and contacts.

Usage Used in Siebel 6.x. For Siebel 7.0, use MVG Set Primary Restricted: *visibility_mvlink_name*.

MVG Set Primary Restricted: *visibility_mvlink_name*

Parent Object Type

Business Component

Description

Setting this user property to FALSE allows the Primary team member to be altered by someone other than the Manager or Siebel Administrator.

If this user property is not set, only Siebel Administrators (in Admin mode) and Managers (in Manager view mode) have the ability to change the Primary team member on opportunities, accounts and contacts.

Usage

MVG Set Primary Restricted: *visibility_mvlink_name*,

where

visibility_mvlink_name is the value of the Visibility MVLink property of the BusComp View Mode child object of the business component for which you want to allow or restrict setting the primary.

For example, if you want to allow sales representatives to set the primary sales team members for contacts:

1. Create a user property for the Contact Business component called MVG Set Primary Restricted: Position.
   Position is the value of Visibility MVLink for the Sales Rep view mode for Contact.
2. Set the value of MVG Set Primary Restricted: Position to FALSE.

Named Method *n*

This user property allows you to invoke a method from a business component or business service, or set a field value.

Parent Object Type Business Component

Syntax The value you provide for the Named Method user property depends on the action you want to perform.

For setting a field value, the value consists of three quoted parameters separated by a comma and a space, as follows:

```
"[Name]", "SET", "[Field]", "[Expression]"
```

When [Name] is called, the value of [Field] is set using [Expression].

For invoking a business component method, the value consists of four quoted parameters separated by a comma and a space, as follows:

```
"[Name]", "[Action]", "[BusComp]", "[Method]"
```

When [Name] is called, [Method] is invoked on the [BusComp] business component based on the defined [Action]. For a list of actions, see Table 12 on page 252.

For invoking a business service method, the value consists of five quoted parameters separated by a comma and a space, as follows:

```
"[Name]", "[Action]", "[BusComp]", "[Service]", "[Method]"
```

When [Name] is called, [Method] from the [Service] business service is invoked on the [BusComp] business component based on the defined [Action]. For a list of actions, see Table 12 on page 252.

You can optionally append an additional parameter that defines an expression. If you use a business service action, the expression is passed as a property set, so you must use name value pairs rather than an array of strings ("NameExpr", "ValueExpr").

Usage You can create additional instances of this user property as needed. If you have more than one instance of this user property for a business component, each instance is executed sequentially by number (for example, On Field Update Invoke 1, then On Field Update Invoke 2, and so on). If there is only one such user property, then no number is required.

Functional Area CSSBCBase

Table 12. Action Values for Named Method

Action	Method Type	Functional Implication
INVOKE	business component	invokes the method
INVOKESEL	business component	saves the state and invokes the method once for each selected record
INVOKEALL	business component	saves the state, requeries, and invokes the method once for each record
INVOKESAVE	business component	saves the state, requeries, and invokes the method
INVOKESVC	business service	invokes the method
INVOKESVCSEL	business service	saves the state and invokes the method once for each selected record
INVOKESVCALL	business service	saves the state, requeries, and invokes the method once for each record
INVOKESVCSAVE	business service	saves the state, requeries, and invokes the method

Named Search: Forecast Series Date Range

This user property specifies a search specification that is applied on the Revenue business component during forecast creation. This is typically activated to make sure the revenues returned by the search are within the Forecast Date range.

Syntax The value of the Named Search: Forecast Series Date Range must be a valid search specification.

Usage If you inactivate the Named Search: Forecast Series Date Range user property, you can make sure the revenues that enter the forecast are limited by date by modifying the Auto and Assoc search specifications of the Forecast Series to include similar criteria.

This search specification (as well as the Auto and Assoc search specifications) can use special variables as defined in the Forecast Series and Forecast Series Date business components. For example, the default value of the Named Search: Forecast Series Date Range user property is

```
[Date] >= '&FCST_DATE_LOWER_BOUND' and [Date] <=
'&FCST_END_DATE'
```

which returns values between the Date - Lower Bound field and the End Date field of the Forecast 2000 -- Forecast Series Date business component. In this case the variable &FCST_DATE_LOWER_BOUND represents the Date - Lower Bound field, which is the History View Date if it has a value. If the History View Date does not have a value, the Date - Lower Bound field is the History Edit date if it has a value or the Start Date if the History Edit date does not have a value.

You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.

Parent Object Type Business Component

Functional Area Forecast

NoDataHide

This user property hides the applet when it contains no data.

Syntax Value:

- Y If no data, applet is hidden.
- N Applet is shown even if no data.

Usage You can inactivate and modify the values for this user property. You can also create new instances of this user property as needed.

Parent Object Type Applet

NoDelete

Parent Object Type	Integration Component
Description	Instructs the Siebel eAI connector not to perform deletes on the business component that the integration component represents.
Functional Area	SAP Account

NoDelete Field

This user property allows you to restrict the deletion of records based on the value of the specified field. When you specify a field in this user property, the business component will not allow records to be deleted that have a value of Y in the specified field.

Syntax	The value of the NoDelete Field user property must be the name of a field in the business component.
Usage	You can inactivate or modify the values for this user property. You can also create new instances of this user property as needed.
Parent Object Type	Business Component
Functional Area	CSSBCBase

NoInsert

Parent Object Type	Integration Component
Description	Instructs the Siebel eAI connector that it should not perform inserts on the business component that the integration component represents.
Functional Area	Siebel Workflow

Non-SalesRep View Mode SearchSpec

This user property allows you to specify the search specification of the business component when the application is *not* in Sales Rep mode.

Syntax

The value for this user property must be a valid search specification for the business component. The fields must exist in the business component and the values for the fields must be valid.

For example, "[Secure Flag] = 'N' OR [Secure Opty Id] IS NOT NULL" is a valid search specification. This value references two fields in the business component (Secure Flag and Secure Opty Id) and has valid values for these fields ('N' and 'IS NOT NULL'). Also, from a business perspective, this value for the user property makes sense because it will only display records from the business component that are not Secure, or records that have a value specified for the Secure Opty Id field (indicating that the current record is not Secure).

Usage

You can inactivate and modify the values for this user property. You can also create new instances of this user property as needed.

Parent Object Type

Business Component

Functional Area

CSSBCFINOppty

NoUpdate

Parent Object Type Integration Component Field

Description Instructs the Siebel eAI connector that it should not perform updates on the field that the integration component field represents.

Functional Area

On Field Update Invoke *n*

This user property allows you to invoke a business component method when the specified field is updated.

Parent Object Type Business Component

Syntax The value for On Field Update Invoke consists of three quoted parameters separated by a comma and a space, as follows:

```
"[FieldToCheck]", "[BusCompName]", "[MethodName]"
```

[MethodName] is invoked on the [BusCompName] business component when [FieldToCheck] is updated. If [FieldToCheck] is not defined, the method is invoked when the user saves the record.

You can optionally use a fourth parameter that defines a condition. If you define a condition, the method is only invoked if the condition evaluates to TRUE.

Usage You can create additional instances of this user property as needed. If you have more than one instance of this user property for a business component, they are executed sequentially by number (for example, On Field Update Invoke 1, then On Field Update Invoke 2, and so on). If there is only one such user property, then no number is required.

Functional Area CSSBCBase

On Field Update Set *n*

This user property allows you to set the value of a field in the business component when another field is updated.

Parent Object Type Business Component

Syntax The value for On Field Update Set consists of three quoted parameters separated by a comma and a space, as follows:

```
"[FieldToCheck]", "[FieldToSet]", "[Value]"
```

[FieldToSet] is set to [Value] when [FieldToCheck] is updated. If the Value parameter is not defined, [FieldToSet] is set to the value of [FieldToCheck].

An expression can optionally be used for the Value parameter. In the following example, the Done field is set using the expression when the Done Flag field is updated.

```
"Done Flag", "Done", "IIF ([Done Flag] = ""Y"",
Today (), """")"
```

Additionally, if you use an expression, you can include a fourth parameter that defines a condition.

NOTE: If you use an expression, it must evaluate to the data type of the targeted field. In the following example, the ToChar function is used to convert the date to a string before concatenating with another string and setting value of the field.

```
"Agreement Start Date", "Name", "ToChar([Agreement
Start Date]) + [Agreement Type]"
```

Usage You can create additional instances of this user property as needed. If you have more than one instance of this user property for a business component, they are executed sequentially by number (for example, On Field Update Set 1, then On Field Update Set 2, and so on). If there is only one such user property, then no number is required.

Functional Area CSSBCBase

Opportunity Name

This user property allows you to specify the name of the Opportunity business component to be used for automatically creating opportunities in Siebel eAutomotive applications.

Syntax	The value of the Opportunity Name user property is the name of a business component (for example, Opportunity).
Usage	You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Contact

Parent Account Field

This user property specifies the name of the field that stores the Parent Account Id.

Syntax	The value of the Parent Account Field user property must be the name of a field on the business component.
Usage	The CSSBCAccountSIS class uses the value of the field specified in this user property as the Parent Account Id for the record. It is not recommended that you change the value of this user property from its default. You cannot inactivate or create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Account

ParentBC Account Id Field

This user property allows you to specify the name of the field in the parent business component that stores the Account Id to be used to show affiliated contacts. It is used in Siebel Life Sciences applications.

Syntax	The value of the ParentBC Account Id Field user property must be the name of a field on the parent business component (for example, Account ID).
Usage	If the ParentBC Account Id Field user property is not defined, the application issues an error when the user clicks the Affiliated Accounts button. You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Contact

Parent Id Field

This user property allows you to specify the name of the field in the current business component that is populated with the parent row id when a parent-child relationship is created in an organization chart.

When a parent-child relationship is created in an organization chart (drag and drop one box over another), the specified field in the child is populated with the parent row id.

Syntax	The value of the Parent Id Field user property must be the name of a field on the current business component.
Usage	If the specified field is not defined, the default value of the Manager Id field from the Contact business component is used. You can inactivate this user property. However, you cannot modify values or create new instances of this user property.
Parent Object Type	Applet
Functional Area	Organization Chart

Parent Read Only Field

Parent Object Type	Business Component
Description	Specifies a TRUE/FALSE test business component/field combination in the parent chain (parent, grandparent, and so on) that, when TRUE, causes the target business component to become read-only.
Functional Area	Data Driven Access
Where Documented	*Siebel Tools Reference*

Political Analysis Field

This user property allows you to specify the name of the field on the business component that indicates the Level of Influence for a Contact. The field specified in this user property must be mapped to a LOV that has the following values:

- Low: No Color
- Political Structure (Medium): Light Grey
- Inner Circle (High): Dark Grey

The Political Analysis Field user property is used in the organization chart to display the Level of Influence for each contact by shading the box of the contact with the appropriate level of gray.

Syntax	The value of the Political Analysis Field user property must be the name of a field on the current business component.
Usage	If a field is not specified, the default value of Political Analysis is used. You can modify the values for this user property. However, you cannot inactivate or create new instances of this user property.
Parent Object Type	Applet
Functional Area	Organization Chart

Position Join Fields

This user property allows you to specify the position join fields to be used for updates in Siebel Life Sciences applications.

Syntax	The value of the Position Join Fields user property consists of one or more field names. Multiple field names should be contained in quotes and separated by a comma and a space (for example, "Rep Specialty", "Rep TOP", "Primary Address Id"). There is no limit to the number of field names you can specify.
Usage	You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Contact

Post Default Created Date To Date Saved

This user property specifies whether or not to set the Created Date to the Saved Date when the record is saved. The default behavior is to set the Created Date only when the record is first created.

Parent Object Type Business Component

Syntax Values:

- TRUE Sets the Created Date to the Saved Date whenever the record is saved.
- FALSE Created Date is not changed when the record is saved.

Usage You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.

Functional Area Service Request

Primary Position Modification

Parent Object Type Business Component

Description Specifies whether or not Sales Method for an Opportunity can only be modified by the primary position.

Values:

- Y Sales Method can only be modified by primary position.
- N Sales Method can be modified everyone.

Usage You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.

Functional Area Opportunity

Private Activity Search Spec

This search specification is applied to non-Calendar activities.

Syntax	Same as the Search Spec user property.
Usage	You can inactivate and modify the values for this user property. You can also create new instances of this user property as needed.
Parent Object Type	Business Component
Functional Area	Activity

Recipient Communications User Properties

Recipient Email Address Field and Recipient Fax Address Field are used by Communications Server to send email packages. Generic email and fax recipients that appear in the Pick Recipient applet are obtained through user properties. The list of recipients that appear in the Pick Recipient applet consists of generic names such as Service Request Owner and Contact Name, rather than the actual names of the persons, which are then obtained from the business component records.

These generic names are configured in user properties in each business component. Just as the set of templates that is listed in the Body drop-down list in Send Email and Send Fax dialog boxes is configurable, the list of generic recipients in that dialog box is also configurable. However, recipients are added through business component user property child object definitions. Generic recipient means a generic name for the person, rather than the person's name, email address, or fax number.

For example, when the user generates an email or fax for a service-request record, the user has the choice of Service Request Owner or Service Request Contact for recipients. For a contact record, the user might see only Contact Name. The actual names are not listed in the Pick Recipient applet. The name and corresponding fax number or email address is often extracted from specific fields in the current business component record. Contact records are an example of this, as they contain name, email address, and fax number fields.

Alternatively, the recipient information may be obtained from a record in another business component through a Join. An example of this is service requests. The Service Request Owner recipient information comes from an employee record and the Service Request Contact information comes from a contact record. These fields are based on Joins from the service request record. To configure nonjoined generic recipients, configure the user property child object definitions of the business component.

See Also

- "Recipient First Name Field" on page 273
- "Recipient Last Name Field" on page 273
- "Recipient Email Address Field" on page 273
- "Recipient Fax Address Field" on page 273

Recipient First Name Field

Parent Object Type Business Component

Description Set the value to the business component field that contains the first name of the recipient.

Functional Area Email/Fax/Page (communication requests)

Recipient Last Name Field

Parent Object Type Business Component

Description Set the value to the business component field that contains the last name of the recipient.

Functional Area Email/Fax/Page (communication requests)

Recipient Email Address Field

Parent Object Type Business Component

Description Set the value to the business component field that contains the email address.

Functional Area Email (Send Email, email communication requests)

Recipient Fax Address Field

Parent Object Type Business Component

Description Set the value to the business component field that contains the fax address. The fax address must be in a format appropriate for your fax server. For more information, see the chapter about Driver Parameters for Internet SMTP/POP3 Server in *Siebel Communications Server Administration Guide*.

Functional Area Fax (Send Fax, fax communication requests)

Recipient Preferred Medium Field

Parent Object Type Business Component

Description Sets the value to the business component field that stores the recipient's communications channel preference. If the setting Only Send Preference is specified for a communication request, then a communications template is sent to a recipient if the template's channel type corresponds to the value in the field indicated by this user property. If this user property is not set, then the preference is retrieved from the Preferred Communications field, if the business component includes such a field.

Functional Area Email/Fax/Wireless Messaging (wireless messaging, communication requests)

Recipient Id Field *n*

Parent Object Type Business Component

Description A comma-delimited list of three required values and an optional fourth value. The values are:

Id Field Name. Identifies the foreign key field in the parent business component that points to records in the joined business component.

Business Component. Identifies the joined business component.

Label. The text of the label to appear for this generic recipient in the Pick Recipient dialog box, such as "Service Request Owner."

(Optional) Field Name in Target Business Component. Include this value if the field name is other than ID.

Functional Area Send Email/Fax/Wireless Message

Remote Source

Parent Object Type Business Component

Description Optional. Names external data source used by the business service. For example, DSN = EXCELABCCust.

Functional Area

Required

This user property allows you to make the parent field a required field under certain conditions. You specify the condition by defining a calculated expression for the value of the user property.

Syntax	*Expression*
Usage	When the expression evaluates to Y, the parent field is required.
Parent Object Type	Field
Functional Area	This user property is valid when used for business components based on or inherited from the class CSSBCBase; it is not valid for business components based on CSSBusComp.

Required Position MVField

This user property modifies the behavior of the WriteRecord method to check and require that the current employee must hold at least one position.

Syntax

```
"[Employee Flag]", "[Position MVField]"
```

Quotes are required.

Values:

- [Employee Flag] Specifies the Employee Flag field that sits on S_CONTACT.EMP_FLG.
- [Position MVField] Specifies the multivalue field for the positions that the employee holds.

Usage

The relationship of the multivalue field should go through S_PARTY_PER table. Do not confuse this with the positions that can see the Employee or Contact record; that relationship goes through the S_POSTN_CON table.

This user property is ignored when the business component is used within the EAI or Siebel Adapter context.

You can inactivate and modify the values for this user property. You can also create new instances of this user property as needed.

Parent Object Type

Business Component

Functional Area

This user property applies only to employee-related business components.

Response Type Call Back

Parent Object Type	Business Component
Description	Defined for Offer, Email Offer, and Web Offer business components. If this property is set to Y, it creates a response type "Response Type Call Back" when you create a new Email/Web Offer.
Functional Area	Siebel eMarketing

Response Type More Info

Parent Object Type	Business Component
Description	Defined for Offer, Email Offer, and Web Offer business components. If this property is set to Y, it creates a response type “Response Type More Info” when you create a new Email/Web Offer.
Functional Area	Siebel eMarketing

Response Type Unsubscribe

Parent Object Type	Business Component
Description	Defined for Offer, Email Offer, and Web Offer business components. If this property is set to Y, it creates a response type "Response Type Unsubscribe" when you create a new Email/Web Offer.
Functional Area	Siebel eMarketing

Revenue Aggregation Field n

This user property specifies a field in the business component that is rolled up into the summary records from the detail records.

Syntax	The value of the Revenue Aggregation Field user property must be a field name in the Forecast 2000 -- Forecast Item Detail business component. For example, a Revenue Aggregation Field user property with the value *Amount Revenue* sums the Amount Revenue field from the detail records and stores the sum in the summary record.
Usage	You can create additional instances of this user property as needed. If you have more than one instance of this user property for a business component, they are executed sequentially by number (for example, Revenue Aggregation Field 1, then Revenue Aggregation Field 2, and so on). If there is only one such user property, then no number is required. You can also inactivate this user property.
Parent Object Type	Business Component
Functional Area	Forecast

Revenue Associate List

This user property causes the Associate applet to show the Revenue business component rather than a standard child business component. Additionally, it applies the Associate search specification and the Forecast Date Range search specification when querying the Revenue business component.

Syntax

Values:

- Y Causes the Associate applet to show the Revenue business component.
- blank Causes the Associate applet to show a standard child business component.

Usage

If you inactivate this user property, the behavior defaults to standard association; the business component is defined in the Associate applet.

You cannot create new instances of this user property.

Parent Object Type

Business Component

Functional Area

Forecast

Revenue Field Map: *fieldname*

The Revenue Field Map user property specifies a field to be copied from the Revenue business component to the Forecast 2000 -- Forecast Item (or Forecast 2000 -- Forecast Item Detail) business component. This user property is used during detail creation.

Syntax	The field specified in the Name of the user property is a Forecast 2000 -- Forecast Item field, and the field specified in the Value of the user property is a Revenue business component field.
Usage	You can inactivate and modify the values for this user property. You can also create new instances of this user property as needed.
Parent Object Type	Business Component
Functional Area	Forecast

Sequence Field

Parent Object Type Business Component

Description Used in configuring a sequence field to create a sequential auto-generating line number on new record and copy record events. A sequence business component must be defined in the business object.

Set the Name property value to Sequence Field.

Set the Value property to Line Number. The value of Line Number references the DTYPE_NUMBER field named Line Number that holds the resulting sequence value.

Set the Insert Position property to LAST for the applets that display records from the numbered detail business component.

NOTE: In older versions of Siebel applications, if this property was left blank, it defaulted to LAST. In Siebel 6.0 and higher versions, it does not default to LAST; leaving this property blank could cause unexpected behavior in the line numbers generated in the applet.

Functional Area

Where Documented *Siebel Tools Reference*

Sequence Use Max

Description

Determines whether the sequence number generated is based upon the current record position or the maximum sequence number. If user property Sequence Use Max is *n*, then generate the sequence based on the current record position.

For example, you want Quote Items to have an incrementing line number.

Quote Item business component has the following fields/columns:

Line Number/ATTRIB_14 (or any Number field) (Predefault = 1)

1. Add a business component user property to the Quote Item business component with the following properties:
   Name: Sequence Field
   Value: Line Number
   Set the Insert Position property to LAST for the applets that display records from the numbered detail business component.

   NOTE: In older versions of Siebel applications, if this property was left blank, it defaulted to LAST. In Siebel 6.0 and later versions, it does not default to LAST; leaving this property blank could cause unexpected behavior in the line numbers generated in the applet.
2. Create the sequence business component with the following properties:
   Name: Quote Item.Line Number (Sequence)
   Class: CSSSequence
   Table: S_QUOTE_ITEM
   Field/column: Quote Id/SD_ID
   Field/column: Sequence/LN_NUM
3. Create Link from Quote Item to Quote Item.Line Number (Sequence) with the following properties:
   Link Name: Quote Item / Quote Item.Line Number (Sequence)
   Dest field: Quote Id
   Source field: Id
4. Add Quote Item.Line Number (Sequence) business component to the Quote business object using the link created in step 3.
5. Expose the fields in a list applet, and create a Parent-Child view with a form and a list applet.

You can renumber all records by invoking ReSequence () method from a button.

Sequence Use Max Example:

NAME SEQUENCE Record

1. Record

2. Record

Suppose that Record 2 is the current record. The new record sequence depends on your insert position. Create a New record called MYREC.

If you insert LAST, then MYREC sequence number = 1.

If you insert AFTER, MYREC sequence number = 2.

If you insert BEFORE, MYREC sequence number = 1.

If you insert FIRST, MYREC sequence number = 0.

Parent Object Type Business Component

Functional Area Cfg Web

Where Documented *Siebel Tools Reference*

Service Name

Parent Object Type Business Component

Description Name of business service that is used by a Virtual business component.

Functional Area

Service Parameters

Parent Object Type Business Component

Description Optional. Names a text string of parameters parsed typically by the Pre_Invoke method and used by the Virtual business component. For example, DLLName = VirtualBusCompODBC.dll.

Makes additional business components accessible to the State Model business component Multi-Value Group applet.

1. In Tools, go to the business component you want to add and navigate to the business component user properties object. Add a record called State Model with a value of Y.
2. Recompile the .srf file.
3. Open the Siebel application and navigate to Workflow Administration > State Models. Add a record to the State Model list applet. All of the business components you added in Tools appear in the business component Name Multi-Value Group.
4. When you select the appropriate one and move to Field Name, the MVG applet that pops up will include all of the fields for the previously selected business component.

The State Name values are pulled into State Model depending on what values are listed in the List Of Values in the Application Administration List of Values Screen. Administration of the List of Values is completely separate from the State Model, however.

NOTE: Only values that appear in the LOV can be used in the State Name field. For Account, all the LOV names are ACCOUNT_STATUS with different display values associated with each.

After adding a business component, you should be able to create the State Model rules that Workflow will enforce.

Functional Area Siebel Business Process Designer

Set Primary Sales Rep As Owner

This user property finds the Primary Sales Rep assigned to the logged-in user, and specifies that all newly created Activities will be assigned to this Primary Sales Rep.

Syntax Values:

- Y Causes all new Activities to be assigned to the Primary Sales Rep of the current user.
- N (or blank) Feature is disabled.

Usage The Set Primary Sales Rep As Owner user property is applicable only when the business component name is Action (Web) and it is used for the Professional Portal LS application.

You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.

This user property is supported for use in CSSBCFINSActivity, but not in its subclasses.

Parent Object Type Business Component

Functional Area CSSBCFINSActivity

Set User As Contact

This user property associates the logged-in user to a newly created Activity and sets the user as the Primary Contact.

Syntax

Values:

- Y Sets the current user as Primary Contact for new Activities.
- N (or blank) Feature is disabled.

Usage

The Set User As Contact user property is applicable only when the business component name is Action (Web), and it is used for the Professional Portal LS application.

You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.

This user property is supported for use in CSSBCFINSActivity, but not in its subclasses.

Parent Object Type

Business Component

Functional Area

CSSBCFINSActivity

Share Home Phone Flag Field

Parent Object Type Business Component

Description Specifies whether or not to hide an employee's home phone number from the Contact List and Employee List views.

Values:

- Y Hides the employee's home phone number.

Employees and Administrators can check the Share Home Phone Flag field to show their home numbers and uncheck to hide.

The functionality only hides the value of the Home Phone # field if the Share Home Phone Flag field is set to N and the Employee Flag field is Y.

Usage You can inactivate and modify the values for this user property. You can also create new instances of this user property as needed.

Functional Area User

Show Required *n*

The Show Required user property allows you to specify a control on the applet to be required. The control specified in the value of this user property is validated as an applet-level required field.

Syntax	The value of this user property is the name of a control on the applet.
Usage	When using the Show Required user property, the corresponding business component must use the CSSBCUser class. Additionally, the control specified in the value of this user property must be present on the applet user interface; otherwise the user has no way of entering a value for the required control. For example, creating a user property called Show Required 1 with the value EmailAddress causes the EmailAddress control on the Applet to be required. You can create additional instances of this user property as needed. If you have more than one instance of this user property for an applet, they are executed sequentially by number (for example, Show Required 1, then Show Required 2, and so on). You can also inactivate this user property.
Parent Object Type	Applet
Functional Area	CSSSWEFrameUserRegistration

Skip Existing Forecast Series Date

This user property restricts a user's ability to pick a date in the Forecast Date pop-up window on the Forecast views.

Syntax Values:

- Y If a forecast exists for the current user for the current date, the date is not available for the user to pick.
- N No restriction on picking the Forecast Date.

Usage You can inactivate or modify the values for this user property. However, you cannot create new instances of this user property.

Parent Object Type Business Component

Functional Area Forecast

SleepTime

Parent Object Type	Business Service
Description	Sets the time out interval, in seconds, on receive requests. The default value is 20 seconds. **NOTE:** Setting the SleepTime user property to a low value or zero can have serious negative performance consequences. Setting a low SleepTime value can cause SAP to flood its system buffers and to retry sending IDOCs to Siebel, because the connector is not “receiving” them.
Functional Area	eBusiness Application Integration Business Services

Sort Search Optimization

Parent Object Type Field

Description An optimization that helps some databases (mostly Oracle) choose the correct index when there are multiple candidate indexes with the leading edge. The search is a no-op, since for nonnull fields, searching for greater than ASCII 0 will return everything.

If this is a required field, the business component is in Sales Rep or Organization Visibility mode, and the Visibility field (position or organization) is in the same table as preceding field, optimize by adding a dummy search specification (for example, NAME > = '\1') on the first column of the business component sort specification. Optimization can be disabled on the business component level by setting Sort Search Optimization to FALSE.

NOTE: Sort Search Optimization is always disabled for DB2.

Functional Area

State Model

Parent Object Type Business Component

Description Makes additional Business Components accessible to the State Model business component Multi-Value Group applet.

To add business components to the State Model business component multi-value group applet

1. In Tools, go to the business component you want to add (for example, Account) and navigate to the Business Component User Prop object.
2. Add a record called "State Model" with a value of "Y" (do not include the quotation marks).
3. Recompile the .srf file.
4. Re-enter the Siebel application and navigate to Site Map > Workflow Administration > State Models > State Models.
5. Add a record to the State Model List Applet.
6. All of the Business Components you added in Tools should appear in the business component Name Multi-Value Group.
7. When you select the appropriate one and navigate to Field Name, the MVG applet that pops up will include all of the fields for the previously selected Business Component.
   The State Name values populate the State Model depending on the values listed in the List of Values (Site Map > Application Administration > List of Values).

NOTE: Administration of the List of Values is separate from the State Model. However, only values that appear in the LOV can be used in the State Name field. For Account, all the LOV names are ACCOUNT_STATUS with different display values associated with each.

Once you have added business components to the MVG Applet, you should be able to create the State Model rules that Siebel Workflow will enforce.

Functional Area Siebel Workflow

Where Documented *Applications Administration Guide*

Text Length Override

Parent Object Type Field

Description Use the field's Text Length property to define the maximum field length instead of the database column size. Use only for Text type fields.
Replaces Field Length property in older versions of Siebel applications.

Functional Area

TypeRetailNew

This user property allows you to specify the value used in the Opportunity Type field that indicates that a given opportunity is a new retail opportunity.

Syntax The value of the TypeRetailNew user property is a text string.

Usage This user property is used by the CSSBCFINOppty business component class when calculating the Actual Number of new retail opportunities. For example, defining the TypeRetailNew user property with the value Retail New causes CSSBCFINOppty to include only opportunities with the Opportunity Type of Retail New when calculating the Actual Number of new retail opportunities.

If the TypeRetailNew user property is not defined, the default value of New is used.

You can inactivate or change the value of this user property. You can also create new instances of this user property as needed.

Parent Object Type Business Component

Functional Area eAutomotive

TypeRetailUsed

This user property allows you to specify the value used in the Opportunity Type field that indicates that a given opportunity is a used retail opportunity.

Syntax The value of the TypeRetailUsed user property is a text string.

Usage This user property is used by the CSSBCFINOppty business component class when calculating the Actual Number of new retail opportunities. For example, defining the TypeRetailUsed user property with the value Retail Used causes CSSBCFINOppty to include only opportunities with the Opportunity Type of Retail Used when calculating the Actual Number of used retail opportunities.

If the TypeRetailUsed user property is not defined, the default value of Used is used.

You can inactivate or change the value of this user property. You can also create new instances of this user property as needed.

Parent Object Type Business Component

Functional Area eAutomotive

Update Parent BC

This user property specifies the name of the parent business component to update at the end of an Account hierarchy update.

Syntax	The value of the Update Parent BC user property must be the name of an active business component.
Usage	The business component specified in the Update Parent BC user property will be updated at the end of the hierarchy update. You can inactivate or change the value of this user property. However, you cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Account

Update Planned Field On Set: StartDate, StartTime

This user property causes the Planned field to be updated when the Start Date field is updated.

Syntax

Values:

- Y Causes the Planned field to be updated when the Start Date field is updated.
- N (or any value other than Y) Automatic update functionality is disabled.

Usage

You can inactivate or modify the values for this user property. However, you cannot create new instances of this user property.

Parent Object Type

Business Component

Functional Area

CSSBCPharmaSpecializedAct

Update Status To Synchronized

This user property causes the status of certain Activities to be set to Synchronized during handheld synchronization. The types of Activities that will be updated are specified using the Update Status To Synchronized Types user property.

Syntax

Values:

- Y Causes the status of certain Activities to be changed to Synchronized.
- N (or blank) Feature is disabled.

Usage

The Update Status To Synchronized user property is used during handheld synchronization. For Siebel Industry Applications, Activities with the Status field set to Synchronized are read-only in the Activities view.

You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.

This user property is supported for use in CSSBCFINSActivity, but not in its subclasses.

Parent Object Type

Business Component

Functional Area

CSSBCFINSActivity

Update Status To Synchronized Types

This user property defines which types of Activities will have their Status field changed to Synchronized during handheld sync.

Syntax

Values:

- *,type1,type2,* The Status of *type1* and *type2* Activities will be changed to Synchronized during handheld synchronization.
- N (or blank) No Activities are changed.

For example, if you set the Update Status To Synchronized user property to Y, and set the Update Status To Synchronized Types to

```
,Account Call,Professional Call,Attendee Call,
```

the Status field for Account Call, Professional Call, and Attendee Call will be updated to Synchronized during handheld synchronization.

Usage

The list of types must be prefixed and suffixed by a comma, and each type must be separated by a comma.

The Update Status To Synchronized user property must be set to Y.

For Siebel Industry Applications, Activities with the Status field set to Synchronized are read-only in the Activities view.

You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.

This user property is supported for use in CSSBCFINSActivity, but not in its subclasses.

Parent Object Type

Business Component

Functional Area

CSSBCFINSActivity

Validate Parent Account

This user property indicates whether or not to validate the Parent Account Id during Account hierarchy changes.

Syntax Values:

- Y Validate the Parent Account Id.
- N Do not validate.

Usage It is not recommended that you change the value of this user property from its default.

You cannot inactivate or create new instances of this user property.

Parent Object Type Business Component

Functional Area Account

WebGotoPlayerErrorPage

Parent Object Type	Applet
Description	This indicates the Web page to display when an error occurs while playing a SmartScript in Siebel eMarketing. The error page has a control to return to the eSmartScript page that caused the error.
Functional Area	Web applets

WebGotoView

Parent Object Type	Applet
Description	This indicates the view to go to if either the FINISH or the CANCEL event is invoked (when the corresponding button is pressed) in the Siebel SmartScript player applet while playing a SmartScript in Siebel eMarketing.
Functional Area	Web applets

WorkFlow Behaviour

This user property specifies whether or not the application is a Siebel Financial Services application. It is used to distinguish Siebel Financial Services applications from other types of applications for the Workflow Process Object Manager.

Syntax

Values:

- Y (or any value) Specifies that this is a Siebel Financial Services application.
- blank Specifies that this is not a Siebel Financial Services application.

Usage

You can inactivate and modify the values for this user property. However, you cannot create new instances of this user property.

This user property is supported for use in CSSBCFINSActivity, but not in its subclasses.

Parent Object Type

Business Component

Functional Area

CSSBCFINSActivity

Tags 5

Double-click the Tags book in the Contents tab to see the list of tags.

swe:all-applets, swe:all-controls, swe:list-control

Purpose

Specialized tags used with the XML interface to SWE in "No Template" mode. For internal use by Siebel Systems only.

swe:applet

Purpose
Refers to a Siebel applet to be shown within the view.

Usage
```
<swe:applet id="1" var=”Parent”/>
```

Attributes
id. References applets using the View Web Template Item list.

var. Allows setting of a variable that can be used in the applet template using the `<swe:if-var>` tag to conditionally show HTML. For example, you can use this to show the Applet Title in the applet template only when the applet is a parent (top) applet.

Restrictions
Can be used only in View Web Templates.

swe:calendar

Purpose

Specialized SWE tags used with calendar applets only. For internal use by Siebel Systems.

The calendar tags are:

```
swe:calendar

swe:calendarActivity

swe:calendarActivityLoop

swe:CurrentDayHeader

swe:calendarInterval

swe:calendarIntervalLoop

swe:calendarMultiDayActivity

swe:calendarNotCurrentlyDayHeader

swe:printHr

swe:printTr
```

swe:control

Purpose
Provides a placeholder in an Applet Web Template for a Control object, or a List Column object.

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

Attributes
Id. Maps an applet child object, Control, or List Column to this placeholder tag.

Property. This is optional. If present, this attribute indicates the property of the control or list item that should be rendered on the Web page. If `property` is not specified, this indicates that the tag will show the body only if the control ID is mapped. This attribute should only be used in singleton tags.

The following values can be used for the property attribute:

Value	Description
FormattedHtml	Displays the data for a control or list item in HTML.
DisplayName	Shows the caption property of the control or list item.
ListHeader	Shows a column header for list columns that includes links for sorting the column. This property value should be used only with `swe:control` tags that are mapped to list columns in a Base template for List Applets.
RequiredIndicator	Shows an icon if the control is Required. (For example, the user needs to enter a value for the control before the record can be committed to the database.) The icon to be used is defined in the configuration file for the application under the SWE section, using the parameter RequiredIndicator.

NOTE: When the property attribute is not specified, the property can be displayed within the body of the `<swe:control>` tag using the `<swe:this>` tag.

Restrictions

Can be used in Applet Web Templates of type Base, Edit, New, or Query.

`swe:control` cannot be nested; for example, the following usage is *not* supported:

```
<swe:control id = "1"> <!-- the "parent" control
<swe:this property = "displayname"/>
<swe:this property = "formattedHtml"/>
<swe:control id = "2" property="formattedHtml">
     <!-- A child control gets context or
        <!-- inherits  properties from its "parent"
</swe:control>         <!-- End of parent -->
```

swe:dir

Purpose

Used to create templates that can work with bidirectional languages. This tag gets converted to `dir="rtl"` when running in right-to-left languages. Does not generate anything in left-to-right languages, as this is the default for the browser.

Usage

```
<HTML dir="swe:dir">
```

Restrictions

Can be used within any HTML tag that takes the `dir` attribute.

swe:error

When a server side error occurs on submitting a form, SWE will show the same page again with the error message displayed within the page. For errors that occur outside of a form submission, SWE will continue to use the application's Error Page.

This error message display is mainly developed for showing error messages within a form. It is also used to show an error message in an error page to replace the to be depreciated pageitem.errorMsg way of showing error messages.

To display the error message within a form, place the following tags inside <swe:form> tag:

```
<swe:error>

   <swe:this property = “FormattedHtml”/>

</swe:error>
```

The error messages will be shown in plain text, but each error message will take a new paragraph. It is the responsibility of the enclosing HTML tags to modify the font and style of the error message. Sometimes, the error message may not be visible; this is because the font uses the same color as the background.

If the application developer does not use error tags in the swt files, the code will automatically generate an error node (a CSSSWEErrorSWX instance). This automatically generated error node will be inserted as the first child of the enclosing page/form node.

Syntax

The syntax of the <swe:error> tag is as follows:

```
swe:error
```

Usage

```
<swe:error property="FormattedHtml" />
```

or

```
<swe:error>

   <swe:this property="FormattedHtml" />
```

```
<swe:error/>
```

This tag should be used within all <swe:form> tags.

NOTE: The use of this tag inside a <swe:form> tag in an applet template is used only in the standard interactivity applications. In high interactivity applications, these errors are shown using a pop-up window.

You should also use this tag instead of the <swe:pageitem> tag mapped to the "_SWEErrMsg" item in the application's Error Page. The use of the "_SWEErrMsg" item is deprecated.

An example of the use of this tag is:

```
<swe:form>

   <swe:error>

      <b> <font color="red"> <swe:this property="FormattedHtml"/> </
      font> </b>

   </swe:error>

      .....

</swe:form>
```

When the form is being rendered when there are no errors, the contents of the <swe:error> tag will be skipped.

swe:for-each

Purpose

Iterates the specified number of times. Allows you to reduce the size of template files where the same HTML and Siebel tags are used with controls or page items with different values for the ID parameter.

Usage

```
<swe:for-each count="x" iteratorName="yyyy" startValue="z"/>
```

Attributes

Count. Specifies the number of times the tag should iterate its contents.

startValue. The value that should be assigned to the iterator at the start of the iteration. The tag will start the iteration by assigning this value to the iterator, and will increment it by one for each iteration.

iteratorName. The name of the iterator. This name can be used to get the value of the iterator during the iteration using the syntax `swe:iteratorName`.

You must replace the iteratorName with the actual name of the iterator. For example, if you set the value of the iteratorName to "CurrentID", then you can get the value of the iterator using the syntax `swe:CurrentID`.

Restrictions

None.

swe:for-each-child, swe:child-applet

These two tags, in combination, support a new catalog-like layout for views with master-detail applets. Records from the master applet and the detail applet can be shown interwoven with each other, instead of the traditional layout where the records in the master applet are shown with the records in the detail applet below it.

To create this layout, the master and detail applets are configured as list applets. We will refer to the master applet as a root level applet. It is possible to show more than one set of master-detail relationships within a view (that is, there could be more than one root level applet). To define the relationship between the applets the new Position attribute of the View Web Template Item object type is used. The position attribute works similarly to the Position attribute of the Tree Node object type. The root level applets will have position values like 1, 2, and so on. For the applet with position 1, its immediate child applets will be assigned position values 1.1, 1.2, and so on. It is possible to define a third level applets with position 1.1.1, 1.1.2, and so on. (for example, these are the child applets of the applet with position 1.1).

In the View Web Template Item object definition, only the root level applets are mapped to `<swe:applet>` tags in the view template. The other applets in the view defined in the View Web Template Item object are not assigned an ID value. The layout of these nonroot applets are not specified in the view template, but instead in the applet template of the root level applets. The < swe:for-each-child > and < swe:child-applet > tags are used to specify this layout.

NOTE: Siebel Systems currently supports only applets in the base mode in this layout.

Example

Suppose you have a master-detail view that includes the "Category Items List Applet" as the master applet and the "Sub Category Items List Applet" as the detail applet. The View Web Template Item of the view that contains this applet will have the following values:

New Identifier	Applet	Applet Mode	Position
101	Category Items List Applet	Base	1
	Sub Category Items List Applet	Base	1.1

The base template for the Category Items List Applet will have the following table definition:

```
<table>

<swe:for-each-row>

<tr>

   <td>

   <swe:control id ="5001" /> </td><!-- field value like "Small
   Business" -->

   <td>

   <swe:for-each-child>

      <swe:child-applet> <!-- Show the child applet -->

      </swe:for-each-child>

   </td>

</tr>

</swe:for-each-row>

</table>
```

The base template for the Sub Category Items List Applet will have the following table definition:

```
<table><tr>

<swe:for-each-row>

<td>

<swe:control id="5001"/>  </td><!-- field value like "Desktop" -->

</swe:for-each-row>

</tr></table>
```

<swe:for-each-child>

Purpose

This tag will iterate over each of the child applets defined for this applet (based on the Position value in the View Web Template object of the view to which the applet belongs). This tag can be used only in the base template of an applet. If the applet does not have any child applets, this tag will be skipped.

Usage

```
<swe:for-each-child>  ..... </swe:for-each-child>
```

<swe:child-applet>

Purpose

This tag is used to place the child applet within the parent applet. The base template of the child applet will be used to render the child applet at the point where this tag is placed.

Usage

```
<swe:child-applet/>
```

NOTE: Set the "HTML Number of Rows" property of the Sub Category Items List Applet to the number of values you want to show under each category value. To allow drilling down from the category and sub-category values, configure the appropriate drilldown objects.

The catalog style layout will force a view to be shown in standard interactivity mode and so these tags should not be used in high interactivity applications.

swe:for-each-node, swe:for-each-indent, swe:indent-img, swe:node, swe:applet-tree-list

These SWE tags are used for implementing explorer Views, that is, Views containing a tree applet and a corresponding list applet for viewing the details of the nodes under a selected node in the tree.

The explorer-style (or tree) applet presents hierarchically structured information in an expandable tree control. The tree control is displayed in a frame on the left side of the applet content area. Detailed information for a selected tree node is displayed in the details applet in a frame to the right. The separate vertical frames allow the contents of the tree applet to be scrolled independently from the details applet. This is important because tree structures can typically grow very large in length and width.

A tree control can have repository tree nodes and field values as elements in the tree. We will use the term “tree item” for a tree element regardless of its type. We will call a repository tree node a “tree node.”

In order to display a tree the logic iterates over each item of the tree in a top-down, depth-first fashion and displays one item at time.

Each tree item is indented to place the text at the correct indent level relative to the root, and to display the expand/collapse mark, the text of the item, and the hyperlinks. The indentation is accomplished using a series of GIF images, or just white spaces (when in text-only mode). The expand/collapse mark and the item are displayed using images (or just text, in text-only mode). The list applet associated with the currently selected tree node is displayed as part of the view.

The following new tags have been implemented to accomplish the above: swe:for-each-node, swe:for-each-indent, swe:indent-img, swe:node, and swe:applet-tree-list.

swe:for-each-node

Purpose

Iterates over each visible item in the tree control in a top-down, “depth-first” fashion. This tag is used to display tree nodes and field values. If “count” is not specified, the tag iterates over all nodes in the tree.

swe:for-each-indent

Purpose

Iterates over each level of a tree item. Used for creating indentation when displaying tree items.

Attributes

None.

swe:indent-img

Purpose

Provides a placeholder for a GIF image corresponding to a tree item's current indentation level. At each level, SWE will determine which GIF file to use in the `<img>` tag to output. The GIF image can be either a blank space or a vertical bar.

Attributes

None.

swe:node

Purpose

Provides a placeholder for an item in the tree. A tree item can be a repository tree node or a field value. The display name will be printed if the tree item is a tree node. Otherwise, the field value will be generated. The expand/collapse mark, the item's icon, and the links are also parts of a tree item. Depending on the configuration file settings, the expand/collapse mark is shown as either a GIF image or text. The expand/collapse mark will only be shown for tree items with child items. There are two links associated with each item. There is a link for the + /- mark to expand or collapse the item and the link for the item image for selecting the item (or for going to next or previous workset). The item selection allows the user to access the list applet associated with the tree node. This tag should use `<swe:this>` as a child tag.

Attributes

state. Possible values are “Active” and “Inactive.” “Active” state is used to show a selected node. Otherwise, an “Inactive” state.

type. Set to “DisplayName” or “FieldValue.” Outputs the repository tree node's Display Name if “DisplayName.” Otherwise, outputs field values.

swe:applet-tree-list

Purpose

Provides a placeholder for a list applet to be displayed as the result of selecting or expanding a tree item. The list applet to be shown will depend on the type of the item that is currently selected.

Attributes

None.

swe:for-each-row

Purpose

Iterates over each record that is displayed in a list applet. This tag is used to create columns of data for showing list applets.

Usage

```
<swe:for-each-row count="x"/>
```

Attributes

Count. The maximum number of records to iterate. If the actual number of records in the list applet is less than the value specified in the count, then the tag will iterate only for the actual number of records.

Restrictions

Can be used only in Base templates for List Applets.

swe:form

Purpose
Creates an HTML form to capture user input.

Usage
```
<swe:form name="xxx" htmlAttr="yyy"> ... </swe:form>
```

Attributes
htmlAttr. This is optional. If specified, the value should be valid attributes to the HTML form tag other than method, name, and action. These attributes will be used as is with the HTML form tag that is generated.

Name. This is optional. If specified, creates the HTML form using the specified name. If not specified, uses an internally generated name.

Restrictions
None.

swe:form-applet-layout

Purpose
Serves as a single container for all controls in the main body of a form applet.

Usage
```
<swe:form-applet-layout>

</swe:form-applet-layout>
```

Attributes
None.

Restrictions
You cannot use the `<swe:idgroup>` tag in this template.

swe:frame, swe:frameset

Rather than using the HTML <frame> and <frameset> tags, Siebel applications use the <swe:frameset> (swe:frameset) and <swe:frame> (swe:frame) tags. This is so that SWE is aware of the frame names, and can control refresh and the targeting of URLs. SWE frames and framesets can be used in the following situations:

- **Container page templates**. A container page template is used to create the frame definition document for the application.

  Note the following implementation details of <swe:frame> and <swe:frameset> tags in container pages:

  - The contents of a frame using the <swe:include> tag does not have to be defined, though it is recommended. The contents can be placed directly into the body of the <swe:frame> tag.
  - The contents of the <swe:frame> have to be complete HTML documents, that is, they should contain the HTML document structure tags like <html>, <head>, <body>, and so on. This includes the view templates also.
  - The contents of the <swe:frame> tag when the type is "View" should contain only the <swe:current-view/> tag.

- **View templates**. Frames can be used in a View template to create a frame definition document to show the Applets in the View. SWE will refresh these frames only when one or more of the Applets contained in a frame has new data.

  NOTE: You can use frames in a View template only if frames are also used in the container page and there is a separate frame in the container page for the View.

  Note the following implementation details of frameset definitions in View templates:

  - When placing Applets into frames you need to make sure that at least one <swe:applet> tag within a frame gets mapped to an Applet in the repository. Otherwise empty frames will occur.
  - When a <swe:frame> block contains a <swe:applet> tag, its type attribute should be set to Applet.

swe:frameset

Purpose

This tag is analogous to the HTML frameset tag and is used to define the set of frames contained in the document. This tag is rendered by SWE as an HTML `<frameset>` tag. The body of this tag can only contain the `<swe:frame>` tags described below.

Usage

```
<swe:frameset htmlAttr="xxx"> ... </swe:frameset>
```

Attribute

htmlAttr. This attribute can be used to specify the attributes for the HTML `<frameset>` tag.

swe:frame

Purpose

This tag is used to mark the beginning and end of the contents to be placed into a frame. SWE renders this tag as an HTML `<frame>` tag, with its src attribute set to a SWE URL that will retrieve the contents of the frame. This tag should be placed within the body of the `<swe:frameset>` tag.

Usage

```
<swe:frame type="xxx" name="yyy"> ....  </swe:frame>
```

Attributes

Type. The type attribute is used to indicate the nature of the contents of the frame. SWE uses this information to decide when to refresh this frame. SWE currently supports the following values for this attribute (more will be supported later).

Value	Description
Screenbar	In a container page template, specifies that the frame contains the primary tab bar.
Viewbar	In a container page template, specifies that the frame contains the application menus, Visibility picklist and Search picklist.
View	In a container page template, specifies that the frame contains the current view, that is, the content area.

Value	Description
Page	In a container page template, specifies that the frame contains a Web page. These frames will not be refreshed after initially loading.
Applet	In a View template, specifies that the frame contains an applet.
Content	Content frames are used to create framesets that contain the view frames. This frame is needed to support the display of multiple views in features such as Search Center. Usually, the Content frame contains a frameset that has one frame in it, which will be the View frame. There are other Content frames that contain framesets that have two or more frames, the View frame and frames to show other alternate views like the Search View.
AltView	Used to designate subframes to show one or more alternate views in the content frame, such as Search Center, in addition to the one in the View frame.

Name. This attribute can be used only when the type of the frame is Page. In this case you can use this attribute to optionally specify a name for the frame. For other frame types, SWE will generate standard names for the frames.

SWE supports nested framesets. In this case the `<swe:frame>` tag will contain a `<swe:frameset>` tag, and the Type attribute of the outer `<swe:frame>` tag is set to Page.

swe:gantt

Purpose
Specialized SWE tags used with Gantt charts only. For internal use by Siebel Systems.

The Gantt tags are:

```
swe:ganttChart
swe:ganttChartMajorAxisLegend
swe:ganttChartMinorAxisLegend
swe:ganttChartXObjectExtendedOT
swe:ganttChartXObjectLoop
swe:ganttChartXObjectMutiple
swe:ganttChartXObjectNone
swe:ganttChartXObjectOT
swe:ganttChartXObjectOff
swe:ganttChartXObjectOn
swe:ganttChartYObjectLoop
```

swe:idgroup

SWE supports having separate Siebel object to SWE tag mappings for various browsers. To support this feature we have introduced the concept of a namespace. A namespace is defined by using a new SWE tag called `<swe:idgroup>`.

Syntax

```
swe:idgroup
```

Usage

```
<swe:idgroup name="xxx">
```

Attributes

Name. The namespace ID.

For example, consider the following requirement. You want to show an applet with an DHTML based menu in an IE 5.0 browser, and with regular controls in place of the tasks performed by the menu with other browsers. All other controls in the applet are the same for all the browsers. You can have browser-specific mappings by enclosing those mappings within a `<swe:idgroup>` tag as shown below:

```
<swe:switch>

   <swe:case condition="Web Engine User Agent, IsMemberVirtualUA,
   'VirtualAgent:IE5'">

      <swe:idgroup name="IE5">

         <swe:menu> <!-- a new tag that we will be supporting -->

      </swe:idgroup>

   </swe:case>

   <swe:default>

      <swe:idgroup name="NonIE5">

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

         <swe:control id="2" ..>

      </swe:idgroup>

   </swe:default>
```

```
</swe:switch>
        <swe:control id="3" ..>
        <swe:control id="4" ..>
```

In this case, when applet controls are mapped to `<swe:control>` tags with IDs 1 and 2, they are marked with the namespace "NonIE5". There is a new attribute of "Applet Web Template Item" object called "Namespace" that stores the namespace value. When the mapping is done using the visual Web Layout editor in Siebel Tools, this attribute gets filled in automatically. The mappings to `<swe:control>` tags with IDs 3 and 4 are outside the namespace and hence this attribute will be NULL for these mappings.

The namespace can be applied to other Siebel object-to-SWE tag mapping, such as applets and page items.

swe:if, swe:switch, swe:case, swe:default

The SWE framework supports the following conditional tags: swe:if, swe:switch, swe:case, and swe:default.

swe:if

Purpose

Provides a simple conditional branching capability.

Usage

```
<swe:if condition="xxx,yyy,aaa:bbb,ccc:ddd,..."> ... </swe:if>
```

Attributes

Condition. The condition to check for. In the usage example above:

- `xxx` is the business service name
- `yyy` is the invoke method
- `aaa` is the first argument and its value is bbb
- `ccc` is the second argument and its value is ddd

If the condition evaluates to TRUE, the body of the `<swe:if>` tag is processed. If the condition evaluates to FALSE, the body of the tag is skipped.

You can use the `<swe:if>` tag with any business service, such as a custom business service that checks an HTTP header. However, the output arguments for the business service must contain a property that includes the method name and a value of 1 or 0 to indicate whether it is true or false.

The common conditions supported by the preconfigured business service *Web Engine State Properties* are:

- IsEvenRow
- IsOddRow
- IsCurrentRow
- IsErrorRow

- IsRowMultipleOf
- IsRowPositionOf
- IsLastDisplayedRow
- IsHighInteractive
- IsHighInteractiveApplet
- IsLowInteractive
- Invalid

NOTE: This tag does not provide an "else" capability like the if tags in programming languages. To get that behavior use the tags <swe:switch>, <swe:case> and <swe:default> described below.

---

swe:switch, swe:case, and swe:default

Purpose

These three tags are used together to provide a conditional branching capability similar to the switch, case, and default statements in C/C++ languages. The <swe:switch> is a container tag for the <swe:case> and <swe:default> tags. Anything other that <swe:case> and <swe:default> within the body of the <swe:switch> tag will be ignored. The condition to check is specified as an attribute of the <swe:case> tag. The <swe:case> tags are checked starting from the first <swe:case> tag. If any of the <swe:case> tags satisfies the condition, the other <swe:case> tags and the <swe:default> tags are skipped. If none of the <swe:case> tags satisfy their condition, the body of the <swe:default> tag is processed. There should only be one <swe:default> tag within the body of a <swe:switch> tag.

Usage

```
<swe:switch>

   <swe:case condition="xxx">

      ...

   </swe:case>
```

```
   <swe:case condition="yyy">
      ...
   </swe:case>
   <swe:default>
      ...
   </swe:default>
</swe:switch>
```

Attributes

Condition. Supported only in the <swe:case> tag. If the condition evaluates to TRUE, the body of the <swe:case> tag is processed. Any subsequent <swe:case> tags within the <swe:switch> tag are skipped without checking their associated conditions. If the condition evaluates to FALSE, the body of the tag is skipped.

NOTE: The format for the condition is the same as the format described for the <swe:if> tag.

---

swe:include

Purpose
Used to include another template or HTML file within the current template file.

Usage
```
<swe:include file="xxx.swt"/>
```

Attribute
File. Required. The name of the file to be included. This file must reside in the same folder as the other template files used by the application and have the extension .swt, even if it is an HTML file.

Restrictions
None.

swe:layout

About Layout Control

To enable user layout control, Siebel ERM includes a SWE tag <swe:layout> and supporting view and applet control attributes and objects in Siebel Tools.

Siebel Web Template Files (SWT) are located in the WEBTEMPL directory of your Siebel Server installation directory.

Purpose

To reorder and hide applets. The swe:layout tag is used to conditionally determine the HTML content that will be displayed, based on the current view layout mode and layout preferences.

Usage

```
<swe:layout viewDisplayMode="xxx" appletDisplayMode="xxx"
appletDisplaySize="xxx" />
```

Attributes

viewDisplayMode. This is optional. It can have the value "Layout" or "Show." If viewDisplayMode is "Layout," the tag will be shown only if the user is in "Edit View Layout" mode. If viewDisplayMode is "Show," the tag will be shown only if the user is not in "Edit View Layout" mode. If viewDisplayMode is not specified, the tag will be shown whether the user is in "Edit View Layout" mode or not.

appletDisplayMode. This is optional. It can have the value "Show" or "Hide." If appletDisplayMode is "Show," the tag will be shown only if the applet is visible. If appletDisplayMode is "Hide," the tag will be shown only if the applet is hidden. If appletDisplayMode is not specified, the tag will be shown regardless of applet visibility.

appletDisplaySize. This is optional. It can have the value of "Min" or "Max." If appletDisplaySize is "Min," the tag will be shown only if the applet is minimized. If appletDisplaySize is "Max," the tag will be shown only if the applet is maximized. If appletDisplaySize is not specified, the tag will be shown regardless of applet display size.

Restrictions

When using the <swe:layout> tag, at least one of the attributes *must* be specified.

swe:pageitem

Purpose
Provides a placeholder in a Web Page for a Web Page Item.

Usage
```
<swe:pageitem id="1" property="FormattedHTML" />
```

Attributes
Id. References page items using the Web Page Item list.

Property. This is optional. If present, this attribute indicates the property of the Web page item that should be rendered on the Web page. If property is not specified, this indicates that the tag will show the body only if the Web page item ID is mapped. This attribute should only be used in singleton tags.

The following values can be used for this attribute:

Value	Description
FormattedHtml	Shows the data for the Web page item in HTML.
DisplayName	Shows the caption property of the Web page item defined in the repository. When the property attribute is not specified, the property can be displayed within the body of the `swe:pageitem` tag using the `swe:this` tag.

Restrictions
This tag can be used only in Web Pages.

swe:pdqbar

Purpose

Outputs the list of predefined queries for a view, and executes the selected query. It can be used within the viewbar and view. A `<swe:pageitem>` in the viewbar or a `<swe:control>` in the view can be used in this tag to show a label.

Usage

```
<swe:pdqbar>

...HTML code...

<swe:pageitem id="##"/>

...HTML code...

<swe:this property="FormattedHtml" />

</swe:pdqbar>
```

swe:screenbar, swe:for-each-screen, swe:screenlink, swe:screenname

swe:screenbar

Purpose

Defines the section of the template that renders the screenbar. This tag is used to mark the beginning and end of the screenbar section of the template.

Usage

```
<swe:screenbar>
   ... HTML ...
   <swe:for-each-screen>
      ... HTML ...
      <swe:screenlink property property="xxx" state="yyy
         ... HTML ...
         <swe:screenname/>
            ... HTML ...
      </swe:screenlink>
      ... HTML ...
   </swe:for-each-screen>
   ... HTML ...
</swe:screenbar>
```

swe:for-each-screen

Purpose

Creates a screen bar by iterating over the screens defined in the Page Tab property of the application in the repository.

NOTE: The order in which the screens are iterated is defined by the sequence value of the screens defined in the Page Tab.

Usage

```
<swe:for-each-screen>

   ... HTML ...

   <swe:screenlink property property="xxx" state="yyy”>

      ... HTML ....

            <swe:screenname/>

      ... HTML ...

         </swe:screenlink>

      ... HTML ...

</swe:for-each-screen>
```

Attributes

None.

Restrictions

None.

swe:screenlink

Purpose

Outputs a link to navigate to the screen.

Attributes

State. This is optional. Can have values “Active” or “Inactive.” If state is “Active” this tag will be used only if the current view name being rendered is the currently active view. If state is “Inactive” this tag will be used only if the current view name being rendered is not the currently active view. If not specified, the tag will be shown for all views.

Property. This is optional. Can have only one value, “FormattedHtml,” which will output the HTML for creating a link to navigate to the screen. If this attribute is not specified, then no output will be generated.

htmlAttr. This is optional. Can be used to add additional HTML attributes to the generated HTML tag.

NOTE: The `swe:screenlink` tag can be used without specifying the property attribute, but with a value for the state attribute to conditionally show different HTML for active and inactive views. When the property attribute is not specified, the property can be displayed within the body of the `swe:screenlink` tag using the `swe:this` tag.

---

swe:screenname

Purpose

Outputs the name of the screen.

swe:screenoptionlink

Purpose

Particularly used for generating Phone.com-style option links so the UI will display numbers for list items. It should only be used for WML apps.

swe:scripts

Purpose
Provides a hint for the location to place any JavaScript that may be generated for the page.

Usage
```
<swe:scripts/>
```

Attributes
None.

Restrictions
Siebel Web Engine includes some JavaScript code in every page that is sent to the browser. The requirement is for these scripts to occur after all the Siebel content in the page, but within the HTML <body> tag.

The <swe:scripts> tag is used to indicate to Siebel Web Engine exactly where you wish this code to be placed. By default, if this tag is not specified, Siebel Web Engine places the JavaScript code immediately after the last tag on the page. In some cases this might occur within an HTML formatting tag like <td>... </td>, and sometimes this might cause minor formatting problems in browsers when rendering the page.

NOTE: The <swe:scripts> tag should be the last Siebel tag in the template, and should be contained within the HTML <body> tag.

As mentioned, the Siebel Web Engine always checks for the correct placement of this tag, and will reposition it if needed, and remove any superfluous <swe:scripts> tag.

swe:select-row

Multiselect list applets provide a way to select multiple items for a transaction. The check boxes in the left column are used to select the items.

SWE supports selection of multiple records in list applets for invoking methods that act on these selected records. This is similar to pressing CTRL and selecting records on the dedicated client. The selection of rows is done using check boxes that are placed on each row.

This is different from positioning the current record using the "PositionOnRow" control. You can have both the "PositionOnRow" control and "Multiple Row Selection" on the same list applet. When you initially navigate to a list applet the record on which the business component is positioned will be automatically selected. Users can unselect this using the check box if desired.

Unlike "PositionOnRow," when you select rows using the check box there is no server round trip. The selected records are marked as selected on the business component only when a method is invoked on the applet. You can select records across multiple pages (that is, you can navigate using the "Next" and "Previous" controls and select records from different working sets).

By default, multi-record selection is not enabled for list applets. To enable this feature on list applets where this needs to be supported, set the new attribute of the "List" object in Siebel Tools called "HTML Multi Row Select" to "TRUE."

To render the check boxes to select multiple rows in list applet templates, a new tag, `<swe:select-row>`, has been introduced. Using this tag you can create a generic list applet template that can be used with list applets that support multirecord selection and those that do not. In the list header, use the `<swe:select-row>` tag to conditionally put in a `<td>` for the header for the row selection check box column, and in the list body use the `<swe:select-row>` tag along with the `<swe:this>` tag to conditionally put in a `<td>` that will contain the check box.

NOTE: You need to place your list applet controls and list columns within a `<swe:form>` tag when you enable the multi-select feature, as any invoke method on the applet will require the form which will contain the row selection check boxes to be submitted.

The HTML Client framework will mark the records as selected in the buscomp when a method is invoked on the applet. This happens in CSSSWEFrame::PrepareToInvokeMethod before the CSSSWEFrame::DoInvokeMethod is called. The sequence of events when the framework gets a command to invoke a method on an applet are:

1. The parent business component of the applet (if there are any) are positioned on the correct records.
2. The applet's business component is positioned on the currently active record (if needed).
3. If multiple records are marked, they are selected in the buscomp by calling CSSBusComp::SelectRowById.
4. The method is invoked on the applet by calling CSSSWEFrame::DoInvokeMethod.

Currently none of the methods in the base CSSSWEFrame support multiple records. This may change in the future. If any specialized applet needs to support this feature, the DoInvokeMethod should be specialized. In this method you can call CSSSWEFrame::IsMultiRecSelected to check if multiple records are selected. If TRUE, you can call CSSBusComp::EnumAllSelections to get all the records that are currently selected in the buscomp.

Controls that do not support invoking methods when multiple records are selected will not be disabled as in the dedicated client when the user selects multiple records. This is because there is no server call when selecting multiple records. Instead, when the control is activated a message will be shown to the user that the action cannot be performed when multiple records are selected.

Syntax

To render the check boxes to select multiple rows in list applet templates a new tag, `swe:select-row`, has been introduced. The syntax of this tag is as follows:

```
<swe:select-row property="FormattedHtml" />
```

Attributes

Property. When the property attribute is set to "FormattedHtml" in either the <swe:select-row> or <swe:this> tag, the checkbox will be rendered if the applet is enabled for multi-record selection in Siebel Tools. When <swe:select-row> tag is used without the property attribute, it acts as a conditional tag to show its body if the applet is enabled for multi-record selection.

Example

```
<swe:select-row property="FormattedHtml" />
```

or

```
<swe:select-row>
   <swe:this property="FormattedHtml" />
</swe:select-row>
```

swe:subviewbar, swe:for-each-subview

Purpose

`swe:subview`

A subcategory View picklist is implemented using a `<swe:subviewbar>` tag. The `<swe:subviewbar>` tag can alternately be configured to display a second horizontal tab bar beneath the detail View bar, but the picklist is the preferable approach.

The default behavior of the `<swe:viewbar>` tag, if the `<swe:subviewbar>` tag is not present, is to display the default View for that category when the user selects the category name in the detail View bar. The default View is the View with the lowest sequence number in that category that is visible to the user.

By including the `<swe:subviewbar>` tag, this behavior is augmented with a picklist or subcategory tab bar that conditionally appears when the currently active View belongs to a category. This tag expands to list all the Views that belong to the selected category. If the currently active View does not belong to a category, then this tag does not render anything on the page. Thus, if the user chooses a category name from the detail View bar (which means the default View within that category is now the active View), the subcategory picklist or tab bar is rendered in that default View if this tag is present. If the user chooses a noncategory View from the detail View bar, the subviewbar tag does not render anything on the resulting page.

`swe:for-each-subview`

This tag is used when the type attribute of `<swe:subviewbar>` is not set to Select. This tag iterates over each of the subviews. This is similar to the behavior of `<swe:for-each-view>`.

Syntax

The `<swe:subviewbar>` tag syntax is described below:

```
<swe:subviewbar type="xxx" property="zzz">
```

Attributes

Type. This can have one value: “Select.” If the type is set to Select the subviewbar will be rendered as a HTML select control (picklist) showing the set of available Views in the selected category.

Property. This attribute is to be used only when the type is set to Select (it will not have any effect in other cases). This attribute can have a value of "FormattedHtml", in which case the HTML Select control is rendered. If this attribute is not specified, then this tag acts as a conditional tag to show its contents if there are sub-views.

When the Type attribute is not set to Select, `<swe:viewlink>` and `<swe:viewname>` tags are used within the body of the `<swe:subviewbar>` tag. The behavior of these tags is similar to their use inside of a `<swe:viewbar>` tag.

Examples

- As an HTML select control (subcategory picklist): Refer to CCViewDetail.swt and CCSubViewbar_Drop.swt:

```
<swe:form>

   <swe:subviewbar type="Select">

      <tr class="tier4On">

         <td>   </td>

         <td valign="top"><swe:pageitem id="2"
         property="DisplayName" /></td>

         <td ><swe:this property="FormattedHtml" /></td>

         <td width="100%"> </td>

      </tr>

   </swe:subviewbar>

</swe:form>
```

- As tabs or links in a subcategory tab bar (refer to CCSubviewbar_tabs.swt):

```
<td class='tier4Off'><img src="images/nav/left_cap.gif"
align="absmiddle" width="6" height="19" border="0" alt=""/></td>

<swe:subviewbar>

   <swe:viewlink state="Active" property="FormattedHtml" >

      <td><img src="images/nav/tabon_arrw.gif" align="absmiddle"
      width="6" height="19" border="0"  alt=""/></td>
```

```
            <td class='tier4OnLabel' background="images/nav/
            tabon_back.gif"><nobr> <swe:viewname/> </nobr></td>

            <td><img src="images/nav/tabon_rightcap.gif"
            align="absmiddle" width="6" height="19" border="0" alt="" /></
            td>

         </swe:viewlink>

         <swe:viewlink state="Inactive" property="FormattedHtml" >

            <td class='tier4Off'><nobr> <swe:viewname/> img
            src="images/nav/tab_rightcap.gif" align="absmiddle" width="6"
            height="19" border="0" alt=""/></nobr></td>

         </swe:viewlink>

      </swe:subviewbar>

      <td width="100%" class="tier4Back"> </td>
```

swe:this

Purpose

Refers to the object inside which it is placed. Used to display a property of a parent tag if it is nested within another tag, or if used outside of a tag, to refer to the property of an Applet, View, or Application based on the type of the template in which the tag is used.

Usage

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

   <td> <swe:this property="DisplayName"/>:  </td>

   <td> <swe:this property="FormattedHtml"/> </td>

</swe:control>
```

Attributes

Property. This tag can be used inside any SWE tag that supports the property attribute.

If the `swe:this` tag is used in an Applet Web Template, then it refers to the Applet. In this case the property attribute can have the following values:

Value	Description
Title	Shows the title of the Applet.
RowCounter	Shows a row counter. The format used to show the row counters can be customized using the List of Values SWE_ROW_COUNTER_MAP.

If the `swe:this` tag is used in a View Web Template, then it refers to the View. In this case the property attribute can have the following value:

Value	Description
Title	Shows the title of the View.

If the swe:this tag is used in a Web Page, then it refers to the Application. In this case the property attribute can have the following value:

Value	Description
Title	Shows the title of the Application.

Restrictions

The swe:this tag is a singleton tag except when used to refer to the swe:viewlink or swe:screenlink tags.

swe:this.Id

Purpose

Used to create scrollable tab bars for screens and views. This tag is used to generate a unique ID for the HTML <TD> tag that contains each tab.

Usage

```
<swe:screenlink>

   <td id="swe:this.Id" ...>

   ...

   </td>

</swe:screenlink>
```

Restrictions

Should only be used in the HTML <TD> tag that contains a screen, view, or subview tab. This <TD> tag should be created within a <swe:screenlink> or <swe:viewlink> tag.

swe:this.TableId

Purpose

Used to create scrollable tab bars for screens and views. This tag is used to generate a unique ID for the HTML <TABLE> tag that contains the tabs.

Usage

```
<swe:screenbar>

   ...

   <table ID="swe:this.TableId" ...>

      <swe:for-each-screen>

      ...

      </swe:for-each-screen>

   </table>

   ...

</swe:screenbar>
```

Restrictions

Should only be used in the HTML <TABLE> tag that contains the screen, view, or subview tabs. This <TABLE> tag should be created within a <swe:screenbar>, <swe:viewbar>, or <swe:subviewbar> tag.

swe:threadbar, swe:for-each-thread, swe:threadlink, swe:stepseparator

The thread bar is used to track user navigation among the views. A thread bar in HTML text format has been implemented. An example of the thread bar is as follows:

Home > Consumer:PCs > PCs:Laptops > Laptops:Pentium III

where “Home”, “Consumer:PCs”, and so on, are the thread buttons. The thread buttons are displayed in “title: value” format, and either title or value can be omitted when appropriate. The thread button may contain a hyperlink, which will lead the user to a previous page. The thread buttons are separated by separators. In the preceding example, the right-angle bracket (>) is the separator.

A thread button may have a hyperlink that will lead the user to a previous page. The hyperlink requires a new SWE Command: GotoBookmarkView. The hyperlink for each thread button should contain at least the following parameters:

```
SWECmd=GotoBookmarkView&SWEBMCount=2SWECount =3
```

The SWEBMCount = 2 indicates that bookmark #2 will be used to create the view. SWECount = 3 is the bookmark ID for the current view. With the definition of the SWE tags and thread link format, a thread button for account AK Parker will be translated into HTML format as:

```
<a href = “www.siebel.com/
start.swe?SWECmd=GotoBookmarkView&SWEBMCount=2& SWECount=3>
Account: AK Parker </a>
```

A new bookmark will be created when the user clicks the thread button and brings back a bookmarked view. The bookmark ID for the new view will be the current SWE count (the count passed to the server in the request) increased by 1.

Bookmark deletion policy is not modified with the above bookmark ID assignment policy. By default, the system will keep the most recently created 20 bookmarks and delete previous ones. If the SWE count in the user request is less than the SWE count on the server side, all the bookmarks with a SWE count larger than that in the user request will be deleted.

The HTML threadbar is based on the same configuration used in Siebel Tools as was used previously to display the dedicated client threadbar. The behavior of the thread bar also remains unchanged, and is summarized below:

- When a new screen is requested, a new thread will be created to replace the current thread.
- When a view button is clicked, the last thread step will be replaced by that of the new view requested.
- When the user follows a drill-down link, a new step will be appended on the thread bar for the view requested.
- When a thread button is clicked, all the thread buttons to the right of it will be deleted.
- Some views may not have a thread applet or thread field defined. Showing these views will not cause the thread button to be updated.

When a thread button is clicked, the thread will be truncated up to the step view indicated by SWEBMCount.

Syntax

The following three new SWE tags are defined to create an HTML thread bar: swe:for-each-thread, swe:threadlink, and swe:stepseperator. The usage of these SWE tags is very similar to that of the screen bar and view bar tags.

swe:for-each-thread

Purpose

Iterates over each of the thread steps to show its contents.

swe:threadlink

Purpose

Indicates the definition of a thread button on the thread bar.

Usage

```
<swe:threadlink property="xxx" title="yyy">...</swe:threadlink>
```

Attributes

FormattedHtml. Indicates that HTML hyperlink should be included.

Title. Indicates that the title/value pair of the thread button should be displayed.

swe:stepseperator

Purpose

Specifies the symbol used to separate thread buttons. Include at the beginning or the end of the `<swe:threadbar>` block.

Usage

`<swe:stepseperator>` *`separator_symbol`* `</swe:stepseperator>`

Attributes

None.

NOTE: The `swe:threadlink` and `swe:stepseperator` tags should only be used within the `<swe:threadbar>` tag.

Example

To use a thread bar, insert thread bar definitions into an appropriate SWT file by using the tags defined above. An example is given below:

```
<swe:threadbar>

   ... HTML ...

   <swe:for-each-thread>

      ... HTML...

      <swe:threadlink property="FormattedHtml">

         <span class="threadbar"><nobr><swe:this property="Title"/
         ></nobr></span>

      </swe:threadlink>

      ... HTML ...

      <swe:stepseparator>

         <span class="threadbardiv"> &gt; </span>
```

```
      </swe:stepseparator>
      ... HTML ...
   </swe:for-each-thread>
   ... HTML ...
</swe:threadbar>
```

This will create a thread bar as shown below:

Home > Consumer:PCs > PCs:Laptops

For applications without frames, put the definition in a container page such as CCPageContainer.swt; for applications with frames, insert it in the "Viewbar" frame swt file or the "View" frame swt file.

swe:togglebar, swe:for-each-toggle, swe:togglelink, swe:togglename

Purpose

SWE supports showing toggle applets. Links to navigate between the toggle applets can be rendered either as a drop-down select control or as links or tabs.

The toggle selection control can be rendered in any applet template using the new tag `<swe:togglebar>`. This tag works similarly to the `<swe:viewbar>` and `<swe:for-each-screen>` tags.

swe:togglebar

Syntax

```
swe:togglebar
```

Usage

```
<swe:togglebar type="xxx"  property="zzz">
```

Attributes

Type. This can have one value: "Select." If the type is set to "Select" the togglebar will be rendered as a HTML Select control showing the set of applets that can be toggled to. The applet titles are used as values in the select control.

Property. This attribute is to be used only when the type is set to "Select" (will not have any effect in other cases). This attribute can have a value of "FormattedHtml", in which case the HTML Select control is rendered. If this attribute is not specified, this tag acts as a conditional tag to show its contents if there are toggle applets defined. The `<swe:this>` tag will be used within the body of this tag in this case to render the select control.

If the applet does not have toggle applets defined, this tag and its contents are skipped.

When the type attribute is not set to "Select,"`<swe:for-each-toggle>`, `<swe:togglelink>`, and `<swe:togglename>` tags can be used within the body of the `<swe:togglebar>` tag to create toggle links or tabs similar to the use of `<swe:for-each-view>`, `<swe:viewlink>`, and `<swe:viewname>` tags.

swe:for-each-toggle

Purpose

Iterates over the number of toggle applets to show its contents.

Attributes

None.

swe:togglelink

Usage

```
<swe:togglelink state="xxx" property="yyy">
```

Attributes

State. This is optional. Can have value "Active" or "Inactive." If state is "Active" this tag will be used only if the current applet title being rendered is the currently active applet. If state is "Inactive" this tag will be used only if the current applet title being rendered is not the currently active applet. If not specified, the tag will be shown for all applets.

Property. This is optional. Can have only one value, "FormattedHtml," which will output the HTML for creating a link to toggle to the applet. If this attribute is not specified, then no output will be generated.

swe:togglename

Purpose

Outputs the title of the applet.

Usage

```
<swe:togglename/>
```

Examples

To show the toggle applets as a select control:

```
<swe:togglebar type="Select" >

<table>

   <tr>

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

```
      <td> <swe:this property="FormattedHtml"/> </td>
   </tr>
</table>
</swe:togglebar>
```

where the control is used to create a label like “Show:” before the select control.

To show the toggle applets as tabs or links:

```
<swe:togglebar>
<table>
   <tr>
      <td><swe:togglelink property="FormattedHtml">
      <swe:togglename> </swe:togglelink> </td>
   </tr>
</table>
</swe:togglebar>
```

swe:toolbar, swe:toolbaritem

Toolbars and menus provide the means for the user to initiate various actions. Toolbars appear in their own frame near the top of the application in the browser window.

Clicking on a toolbar icon or menu item is translated into a call to an invoke method, which may reside in a service on the browser or server, or in classes in the browser application or server infrastructure (applet or business component classes, SWE frame manager, or model). The toolbar icon or menu item is configured to target a method name, a method handler (from which it may be automatically retargeted if not found), and, optionally, a service.

Application-level items (which include both toolbar icons and application-level menus) are implemented through the use of Command object definitions in Siebel Tools, which are then mapped to Toolbar Item or Menu Item object definitions. Applet-level menus do not use Command object definitions, but the set of properties used for targeting the method are essentially the same as are found in the Command object type.

In SWE templates, the `<swe:toolbar>` tag specifies a named toolbar (where the name corresponds to the Name property in the Toolbar object definition in the repository), and the `<swe:toolbaritem>` tag between the toolbar start and end tags recursively retrieves all of the toolbar items for that toolbar from the repository.

Two types of toolbars are supported: HTML and Java applet. HTML toolbars reside in the topmost frame in the application template, which is set aside for this purpose. An additional frame beneath this one is specified for Java toolbars in Call Center and similar applications using CTI. If no Java toolbar is used, this frame is omitted.

For an HTML toolbar, in the SWT file, add the following:

```
<swe:toolbar name=xxx>    // where xxx is the name of toolbar in the
repository.

// any HTML stuff here...

<swe:toolbaritem>

// any HTML stuff here...
```

```
</swe:toolbar>
```

NOTE: For combobox items, the command has to be targeted to a service.

For a Java toolbar, add the following to the SWT file:

```
<swe:toolbar name="xxx" javaapplet="true" />
```

The Java applet will invoke the ShellUIInit method on the command target service when it tries to initialize. It will invoke ShellUIExit when it exits. There is a set of communication protocols defined for the communication between the Java Applet and the service.

The toolbar will be implemented as a Java Applet (including all the toolbar controls and the threads interacting with the server).

swe:toolbar

Usage

```
<swe:toolbar name="xxx" javaapplet="true/false" />
```

Attributes

Name. Name of the toolbar, as specified in the repository Toolbar object definition.

Javaapplet. Specify as TRUE to implement a Java toolbar. Specify as FALSE or omit to implement an HTML toolbar.

swe:toolbaritem

Usage

```
<swe:toolbaritem>
```

Attributes

None.

swe:*training*

Purpose

Specialized SWE tags used with the eTraining Test feature. These tags cannot be used anywhere else. A description of these tags follows:

- `swe:answer`. Displays an answer.
- `swe:answerList`. Listing of answers for the test.
- `swe:questionList`. To receive all the questions in the test.
- `swe:questionPoints`. Displays the maximum number of points for the test question.
- `swe:questionPointsReceived`. Points received by the user.
- `swe:questionSequence`. Test question sequence.
- `swe:questionText`. Text for the test questions.
- `swe:test`. Displays the name of the test.
- `swe:userAnswer`. User's answers to the test.

swe:view, swe:current-view

The SWE framework supports showing multiple views simultaneously on a page. The multiple views consist of a “Main” view and one or more “Alternate” views. The main view is the view that is selected using the viewbar (Level 2 or 3) for a given screen. There will always be only one main view. Alternate views are other views that can be shown along with the main view; for example, the Search View that shows applets that can be used for find/search operations.

The multiple views shown on a page can be placed into separate HTML frames or can share the same frame. Moreover, multiple views can be shown on the main browser window or in pop-up windows.

The examples in this document describe creating multiple view layouts when SWE frames are used. The process is similar when frames are not used. In such cases HTML tables can be used in the place of frames and framesets to position the views.

To support multiple views, the structure of framesets and frames used in the application needs to be modified. In addition to frame sets and frames, you must also modify an a layer called the “Content Container.” You can think of this as the container page for the Content area.

The frame of type “View,” which used to be in the Application's Container page, should be replaced with a frame of type “Content.” This frame defines the area where one or more views can be loaded. Initially this frame will contain a frameset that will have the “View” type frame.

The new structure of the container template should be something like that given below:

```
<swe:frameset htmlAttr="rows='80,50,50,*' border='0'
frameborder='No'">

   <swe:frame type="Page" htmlAttr="marginheight='0'
   marginwidth='0' noresize scrolling='No'">

      <swe:include file="CCBanner.swt"/>

   </swe:frame>

   <swe:frame type="Screenbar" htmlAttr="marginheight='0'
   marginwidth='0' noresize scrolling='No'">
```

```
      <swe:include file="CCScreenbar.swt"/>

   </swe:frame>

   <swe:frame type="Viewbar" htmlAttr="marginheight='0'
   marginwidth='0' noresize scrolling='No'">

      <swe:include file="CCViewbar.swt"/>

   </swe:frame>

   <swe:frame type="Content" htmlAttr="marginheight='0'
   marginwidth='0' noresize scrolling='Yes'">

      <swe:include file="CCMainView.swt"/>

   </swe:frame>

</swe:frameset>
```

The file CCMainView.swt will define a frameset that contains the main view.

```
<swe:frameset htmlAttr="cols='100%' border='0' frameborder='No'">

   <swe:frame type="View" htmlAttr=" noresize scrolling='Yes'">

      <swe:current-view/>

   </swe:frame>

</swe:frameset>
```

After making this change, the application should behave as before. An additional layering of frames in the content area was introduced. The previous application container page template that had the “View” frame without the outer “Content” frame will not generate any errors, but will not allow showing multiple views in the application

To show additional views in the content area, we load a different “Content Container” page in the “Content” frame. This can be done by invoking the method “LoadContentContainer” from a control or page item. The “Content Container” to be loaded should be passed in using the User Property “Container.”

NOTE: This should be set to the Web Template Name of the content container page and not to the SWT filename.

For example, to show the search view along with the main view, create a content container page, say “CCSMainAndSearchView.swt”, and load it using the “LoadContentContainer” method. CCSMainAndSearchView.swt will contain the tags to load the main view and search view into two frames as shown:

```
<swe:frameset htmlAttr="cols='100%' border='0' frameborder='No'">

   <swe:frame type="View" htmlAttr="noresize scrolling='Yes'">

      <swe:current-view/>

   </swe:frame>

   <swe:frame type="AltView" name="Search" htmlAttr="noresize
   scrolling='Yes'">

      <swe:view name="Search View" id="Search" />

   </swe:frame>

</swe:frameset>
```

The main view is still referred to by the `<swe:current-view>` tag. Alternate views are referred to using the new `<swe:view>` tag.

swe:view

Syntax

```
<swe:view name="xxx" id="yyy">
```

Attributes

Name. Name of the Alternate View.

Id. An Id for the location (or zone) occupied by this view. This ID will be used to replace this view with another view in its place.

swe:current-view

The location or zone occupied by an alternate view is identified by the View ID, which is "Search" in the above example. This is necessary because, just as with the main view, we want to be able to navigate to other views. In other words, we have multiple view zones now. In the above example, SWE will navigate to the Search view automatically when the Search View Zone is shown the first time. After that, the specialized frame code in the Search view can do a BuildViewAsync() or provide controls of GotoView invokemethod for the users to navigate to other views. The main view has a NULL view ID.

When calling BuildViewAsync(), set the pViewId parameter to the desired View ID. If you are calling from a frame, you can use the frame's view ID, which is set in the data member m_cszViewId of the frame. This will cause a navigation to another view within the same view zone. You can also cause the main view to navigate to another view using BuildViewAsync(). Just make sure you set the pViewId parameter to NULL.

To get the desired view, you can call:

```
CSSSWEFrameMgr::GetView (const SSchar* pViewId = NULL);
```

In other words, you can call:

```
m_pFrameMgr->GetView() to get the main view.

m_pFrameMgr->GetView ("Search") to get the Search view.
```

The CSSSWEFrame contains a new data member now. It is m_cszViewId, which is the view to which it belongs.

swe:viewbar, swe:for-each-view, swe:viewlink, swe:viewname

swe:viewbar

Purpose

The `<swe:viewbar>` tag implements the picklist used for second-level navigation and the detail view list used for third-level navigation.

The Visibility picklist appears in the "viewbar" frame (see the CCPageContainer.swt and CCFrameViewbar.swt templates). The Visibility picklist is implemented as a `<swe:viewbar>` tag with a Type setting of Select and a Mode setting of Context, as shown:

```
<swe:form>
   <td nowrap>
      <swe:viewbar type="Select" mode="Context">
         <swe:this property="FormattedHtml"/>
      </swe:viewbar>
   </td>
</swe:form>
```

The detail View bar is also implemented by means of a `<swe:viewbar>` tag, but with different attribute settings. Specifically, the Type attribute is omitted, and the Mode attribute has a value of NonContext instead of Context. This creates a horizontal View bar consisting of tabs populated with the display names of all the noncontext Views, instead of a picklist control populated with the display names of the context Views. The template logic for rendering the detail View bar is as follows (see CCViewbar_Tabs.swt):

```
<swe:viewbar>
   ... HTML ...
      <swe:for-each-view>
   ... HTML ...
```

```
      <swe:viewlink state="Active">
         ... HTML ....
            <swe:this property="FormattedHtml">
            ... HTML ....
               <swe:viewname/>
            ... HTML ...
            </swe:this>
         ... HTML ...
      </swe:viewlink>
      <swe:viewlink state="InActive">
         ... HTML ....
            <swe:this property="FormattedHtml">
            ... HTML ....
               <swe:viewname/>
            ... HTML ...
            </swe:this>
         ... HTML ...
   </swe:viewlink>
   ... HTML ..
      </swe:for-each-view>
   ... HTML ..
</swe:viewbar>
```

Notice that the detail View bar implementation of the <swe:viewbar> tag requires the use of the child tags <swe:for-each-view>, <swe:viewlink>, and <swe:viewname>. The Visibility picklist implementation omits these child tags.

Syntax

The syntax of the `<swe:viewbar>` tag appears below:

```
<swe:viewbar type="xxx" mode="yyy" property="zzz">
```

Attributes

Type. This can have one value, which is "Select." If the type is set to "Select" the viewbar will be rendered as an HTML select control showing the set of available views (context, noncontext or both, depending on the Mode setting). The user is navigated to the selected view as soon as the user makes a choice in this control.

Mode. The mode can have two values: Context and NonContext. If the value is Context only the context-based views will be shown. If the value is NonContext only the noncontext views will be shown. If this attribute is not specified, all views are shown.

Property. This attribute is to be used only when the type is set to Select. This attribute can have a value of FormattedHtml, in which case the HTML select control is rendered. If this attribute is not specified, then the behavior defaults to the 6.0 behavior, where this tag acts as a conditional tag to show its contents if there are Views to show.

swe:for-each-view

Purpose

Iterates over the views to be shown in the view bar.

Attributes

None.

swe:viewlink

Purpose

Outputs a link to navigate to the view.

Attributes

State. This is optional. Can have value "Active" or "Inactive." If the state is "Active," this tag will be used only if the current view name being rendered is the currently active view. If the state is "Inactive," this tag will be used only if the current view name being rendered is not the currently active view. If not specified, the tag will be shown for all views.

Property. This is optional. Can have only one value, "FormattedHtml," which will output the HTML for creating a link to navigate to the view. If this attribute is not specified, then no output will be generated.

htmlAttr. This is optional. Can be used to add additional HTML attributes to the generated HTML tag.

NOTE: The `swe:viewlink` tag can be used without specifying the property attribute, but with a value for the state attribute to conditionally show different HTML for active and inactive views. When the property attribute is not specified, the property can be displayed within the body of the `swe:viewlink` tag using the `swe:this` tag.

---

swe:viewname

Purpose

Outputs the name of the view.

swe:xsl-stylesheet

Purpose

Specifies the name of the XSLT stylesheet to perform the XSLT on the XML output document. The style sheet must reside in the application's webtempl directory.There is only one `<swe:xsl-stylesheet>` tag per view. If more than one `<swe:xsl-stylesheet>` tag is specified in the view, the last tag found is used.

Usage

```
<swe:xsl-stylesheet name= ”table.xsl” mode= “process”/>
```

Attributes

Name. Specifies the name of the stylesheet.

Mode. XML Interface Reference: Manipulating Siebel XML with XSL Stylesheets and XSLT. You can set the mode to either “process” or “embed.” When set to “process,” SWE performs XSLT processing on the XML output and sends the transformed document as the response back to the client. When set to “embed,” SWE inserts an XML processing instruction in the beginning of the XML document for external XSLT processing.

Siebel Templates for Employee Applications 6

The links below let you access topics that describe the user interface (UI) templates for Siebel Web Client employee applications. Graphical illustrations of the templates are included.

Overview of UI Elements

Table 13 gives an overview of user interface elements in employee applications.

Table 13. Description of UI Elements

UI Elements	Description
Application-Level Menus	Menus with applicable commands carried over from the Windows client.
Branding Area	The area in which the Siebel Systems, Inc. logo will appear.
Control Banner	Area that contains a menu and buttons for a form or list.
Primary Applet	First form or list on page that displays the primary record or record set.
Detail Applet	Displays different sets of data based on the individual primary record. Data is automatically updated when primary record changes.
First-Level Navigation—Primary Tabs	Set of screen-level navigation controls.
Second-Level Navigation—Context Filter	Drop-down list that filters record set based on My, My Team's, and All. It can also display alternative views such as the explorer tree.
Third-Level Navigation—Detail Tabs	Set of applet-level navigation controls that allow the user to see different sets of detail data based on the selected primary record.
Fourth-Level Navigation—Drop-Down List	A single drop-down list used to change the view of a detail form or list. Most commonly used for seeing different levels of charts.
Record Navigation	Area that displays position within the record set as well as controls to move forward/backward within the record set.
Toolbars	The standard application toolbar is implemented using HTML. It displays application-level commands, such as History drop-down and Dashboard show icon.

Table 13. Description of UI Elements

UI Elements	Description
Search	Integrated Search and Global Find launch button.
Favorites	List of saved and predefined queries for the view.

Applet Visual Reference

Applet Form 1-Col (Base/Edit/New)

CCAppletForm1Col_B_E_N.swt

This template also supports the child and grandchild styles. See Figure 1 for an example.

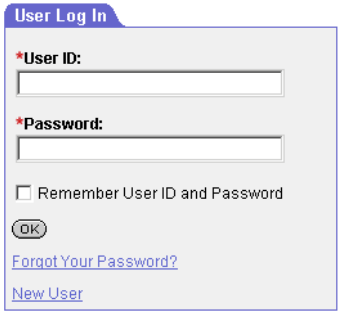
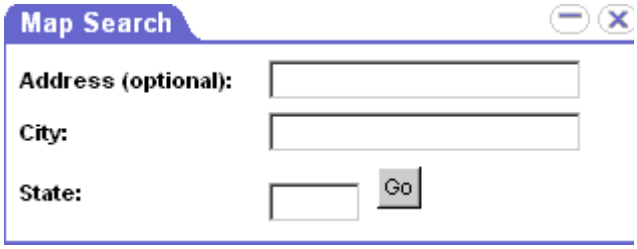


Figure 1. CCAppletForm1Col_B_E_N.swt

Applet Form 1-Col Light (Base/Edit/New)

CCAppletForm1ColLight_B_E_N.swt

Use on portal pages where a light, single-column form treatment is desired, as follows:



Applet Form 4-Col (Base), (Edit/New), and Applet List (Edit/New/Query)

The following three templates are grouped together:

Applet Form 4-Col (Base)

CCAppletForm4Col_B.swt

Applet Form 4-Col (Edit/New)

CCAppletForm4Col_E_N.swt

Applet List (Edit/New/Query)

CCAppletList_E_N_Q.swt

These three applet templates create the same four-column form layout. These templates also support the child and grandchild styles. (Parent and child styles are shown in Figure 2 and Figure 3.)

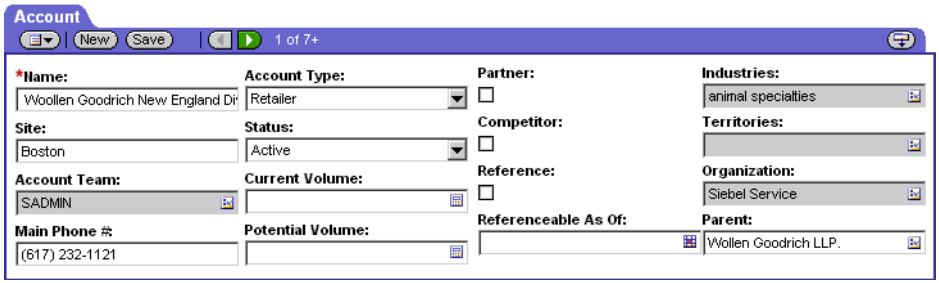


Figure 2. Parent Style

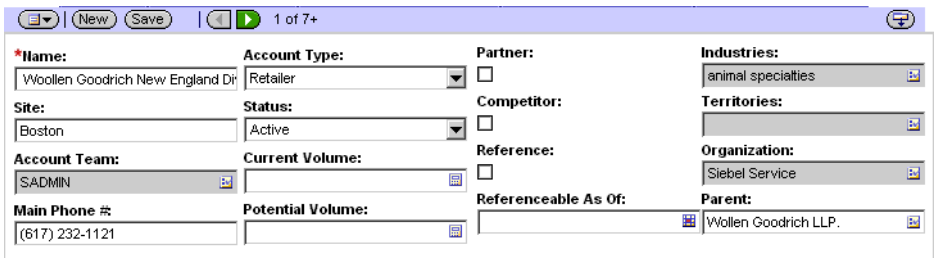


Figure 3. Child Style

Applet List (Base/Edit-List)

CCAppletList_B_EL.swt

Use for read-only and editable lists. The list shown in Figure 4 is depicted in single-row editable format. This template supports the parent, child, and grandchild styles.

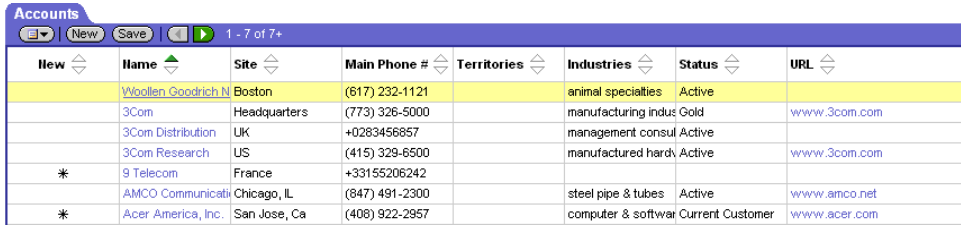


Figure 4. Applet List (Base/Edit-List)

Applet List Totals (Base/Edit-List)

CCAppletListTotals_B_EL.swt

Use for read-only and editable lists that show column totals in the last row, such as the list shown in Figure 5. When using totals list be sure to apply a "Totals:" label to the placeholder in the first column. This template supports the parent, child, and grandchild styles.

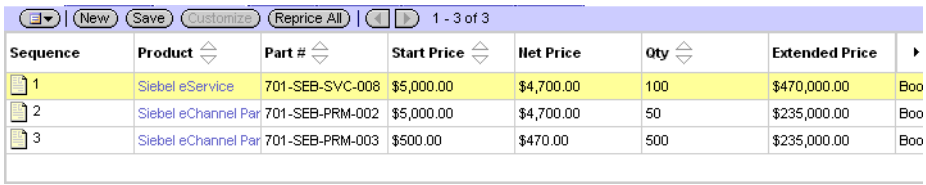


Figure 5. Applet List Totals (Base/Edit-List)

List Portal (Graphical) Applet

CCAppletListPortalGraphical.swt

An example of the List Portal (Graphical) applet is shown in Figure 6.

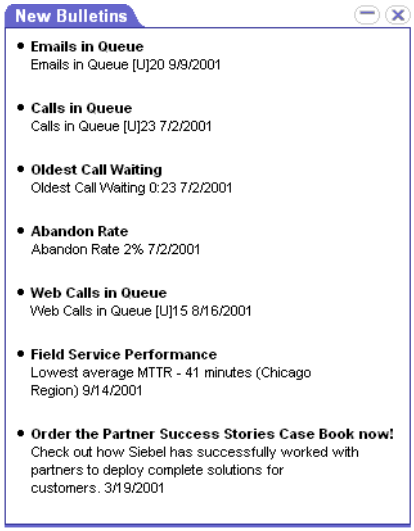
Company Name	Company Site	Partner Type	Partner Tier
Asyrex Technologies	HQ	Consulting Partner	Premier
Total Software Solutions, Inc.	HQ	VAR	Premier
Der Rechenschieber	Munich	Reseller Partner	Global Strategic
CustomerInfo Technologies	Chicago HQ	System Integrator	Premier
Expert Solutions	Dallas	System Integrator	Premier

Figure 6. List Portal (Graphical)

Applet List Messages

CCAppletListMessage.swt

An example of the List Messages Applet is shown below.



Popup List, Popup Query, Popup Form

The following three templates are grouped together:

Popup List
CCPopupList.swt

Popup Query
CCPopupQuery.swt

Popup Form
CCPopupForm.swt

Popup List and Popup Query applets are shown in Figure 7 and Figure 8.

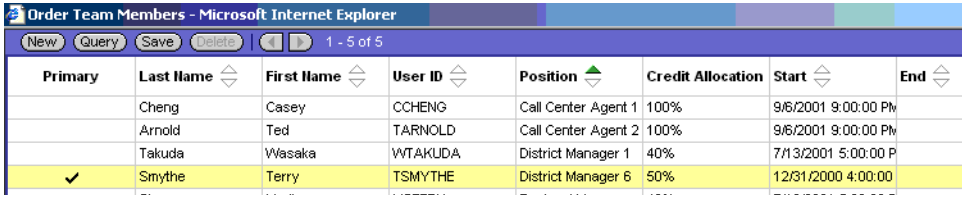


Figure 7. Popup List

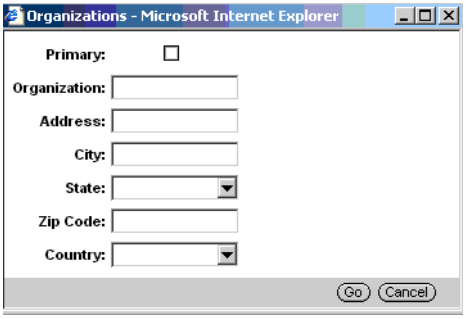


Figure 8. Popup Query

Calendar Monthly, Calendar Weekly, Calendar Daily

The following three templates are grouped together:

Calendar Monthly
CCAppletCalendarMonthly.swt

Calendar Weekly
CCAppletCalendarWeekly.swt

Calendar Daily
CCAppletCalendarDaily.swt

Each of the calendar templates supports the parent, child, and grandchild styles. (Parent style is shown in Figure 9, Figure 10, and Figure 11.)

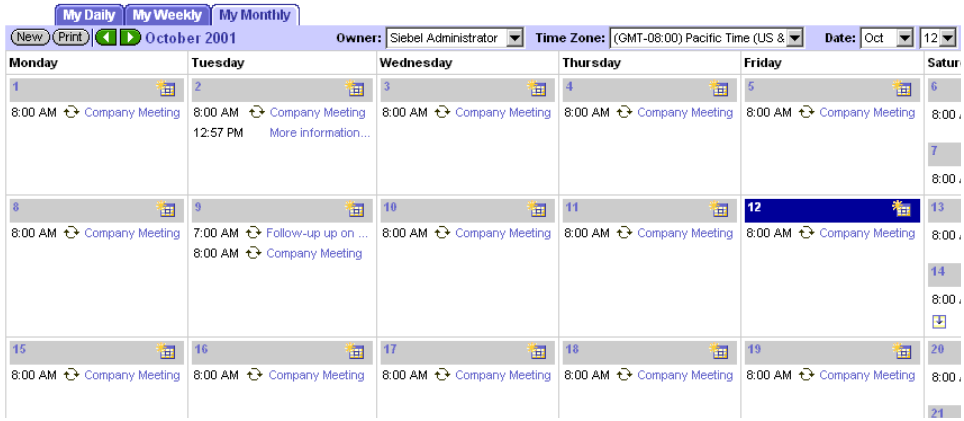


Figure 9. Monthly Calendar Applet

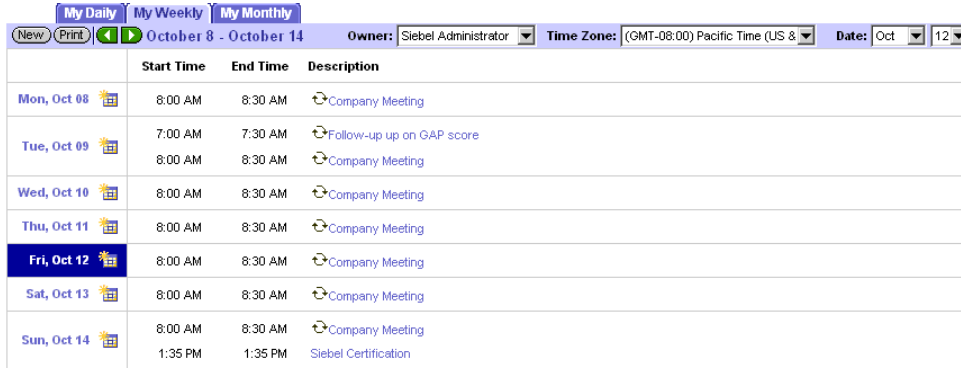


Figure 10. Weekly Calendar Applet

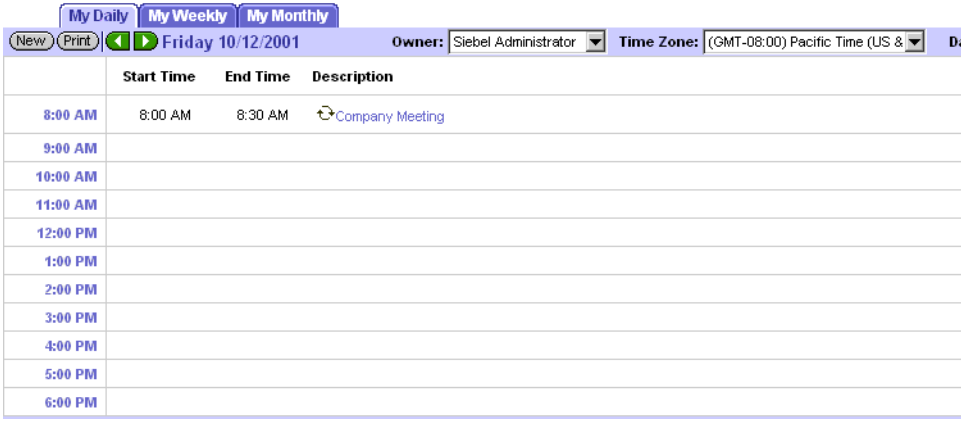


Figure 11. Daily Calendar Applet

Applet Gantt Chart

CCAppletGanttChart.swt

The Gantt chart template supports the parent, child, and grandchild styles, as shown in Figure 12.

Figure 12. Applet Gantt Chart

Applet Chart

CCAppletGanttChart.swt

This template supports the parent, child, and grandchild styles. (Child style is shown in Figure 13.)

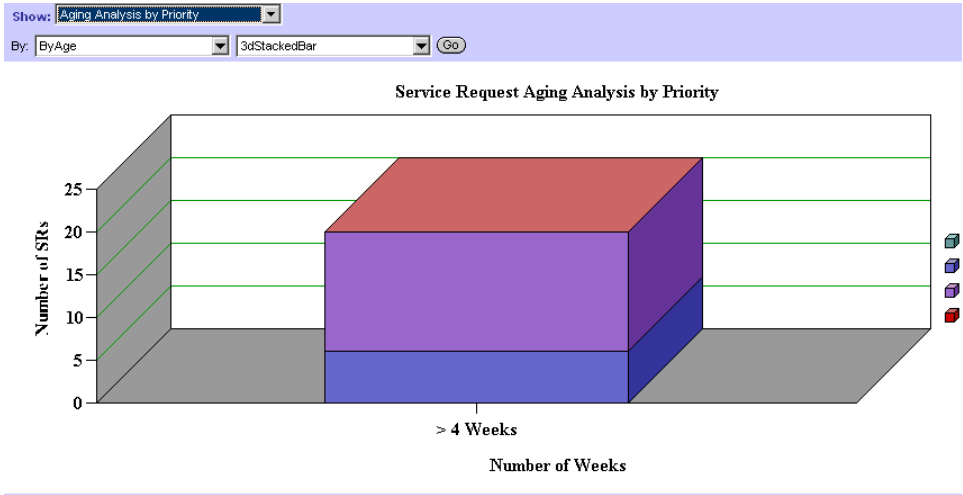


Figure 13. Child Style

Form Layouts

If you are accustomed to designing forms on a tightly defined grid, you may not be aware of the limits that Web form development imposes. The four-column form layout tries to assist you in overcoming some of these limitations.

The four-column form template defines a set of layout regions. A region can hold one or more label/field pairs. Controls are dimensioned in Tools and then placed on regions of the form. The four-column form contains regions of different horizontal proportion; it is possible to accommodate controls that span one, two, and four columns.

Regions are also grouped. Grouping helps guarantee that regions consume only the minimum vertical space required to render them. When controls are not mapped to a region, the region collapses, and the next mapped region moves up the form to take its place.

By combining horizontally proportioned regions with grouped regions, you can achieve a wide variety of form designs while maintaining only one form template.

Figure 14 displays the master template for layout regions.

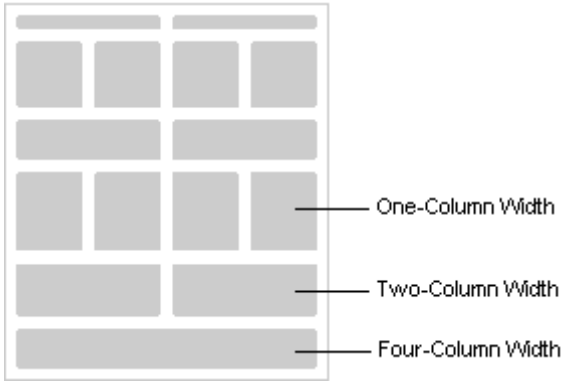


Figure 14. Master Template

Figure 15 and Figure 16 are two examples of layouts that can be derived from the master. The Xs indicate regions that do not contain mapped controls.

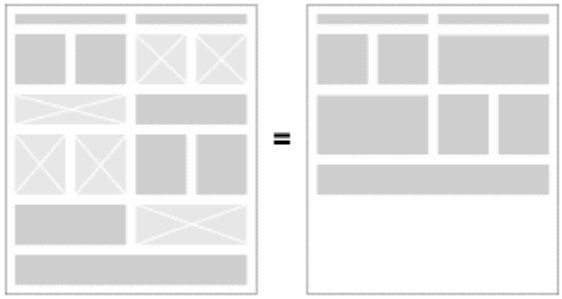


Figure 15. Layout Derived from the Master Template

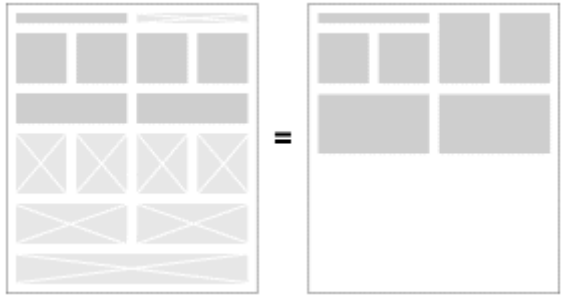


Figure 16. Layout Derived from the Master Template

Controls IDs Per Region

Figure 17 shows the ID ranges for controls within each region. Your strategy for mapping controls to a region should be to place them in the region's topmost free ID.

Figure 17. Control IDs for Each Region

Form Issues

Vertical Alignment of Fields in a Four-Column Form

Each column of fields is laid out independently of the others. This approach maximizes form layout options and minimizes gaps between fields within the same column. A drawback to this approach is that vertical alignment of fields across columns cannot be guaranteed. If you want to maximize the potential for vertical alignment, then place taller fields, such as text area fields, at the bottom of forms.

Mapping a FormSection

A form section is not a control; it is a label that helps to group related fields. Form sections are implemented as a custom control called *FormSection*. In Tools, you will find it in the custom control drop-down within the Web Applet Layout Editor. Map the control onto a label, and fill in its Caption property. The FormSection label will expand to fit the region in which you place it. To set it apart, the label appears against the FormSection color defined in CSS.

NOTE: The control might not appear to expand to fit within the layout editor, but it will be rendered that way in production.

Mapping a MiniButton

You should never map a native HTML button to a list or form. The Siebel eBusiness UI standard is to use the custom control called *MiniButton*. MiniButton is distinct in that it has rounded sides and takes up less of the screen than the average HTML button. It can be found in the custom control drop-down within the Web Applet Layout Editor. Set the MiniButton properties as you would a regular button.

Displaying the Button Divider Between Buttons

To save time during configuration, the button divider is automatically added after the menu button and before the previous record button when those buttons are mapped to your applet.

Displaying the Record Navigation Buttons

There are two record navigation controls implemented as custom controls: *RecNavPrv* and *RecNavNxt*. All record navigation should map these controls because they take up less space and are found within the interface. All record navigation that has carried the “Previous” or “Next” text labels should be migrated to this new standard.

Applet Template Descriptions

Applet List (Base/Edit List)

SWT Filename: CCAppletList_B_EL.swt

This is the standard list template for lists in base or edit-list modes, as shown in Figure 18. A list can typically display between 7-10 visible columns. It is possible to map more visible columns, but this is not recommended as they may appear off screen. The template supports mapping up to 40 fields, but this is done so that you may mark the majority as available but hidden. Fields marked as such will not appear by default in the list but will appear in the columns displayed dialog.

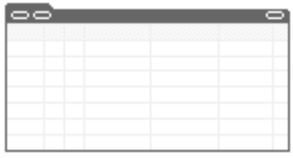


Figure 18. Applet List (Base/Edit List)

Includes Tree

```
CCAppletList_B_EL.swt
   CCApplet_NamedSpacer.swt
   CCTitle_Named.swt
      CCTitle.swt
   CCListButtonsTop.swt
      CCButtons.swt
      CCRecordNav.swt
      CCTogglebar_drop.swt
```

```
    CCListButtonsTopRight.swt
  CCListHeader.swt
  CCListBody.swt
  CCBottomApplet_NamedList.swt
```

Table 14 lists the mappable items for this template.

Table 14. Mappable Items

ID	Description
2	...
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
121	First
122	Previous
123	Next
124	Last
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
142-143	Control
144	Selected Row

Table 14. Mappable Items

ID	Description
145	Control
146	Save
147	Pick Control
150-151	Control
160-164	Control
501-540	Field
598	Save
599	Save
1100	Outside Applet Help Text
1500	Required Legend

Applet List Inverted

SWT Filename: CCAppletListInverted.swt

This is a specialized list applet most commonly used to create comparison lists. See Figure 19 for an example. The list's x and y axes are flipped so that column headers run down the left side of the list. The optimal number of records shown in this type of list is three to five at a time. The list supports record navigation so it is possible to page through larger record sets.

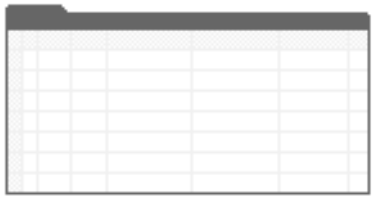


Figure 19. Applet List Inverted

Includes Tree

```
CCAppletListInverted.swt
   CCApplet_NamedSpacer.swt
   CCTitle_Named.swt
      CCTitle.swt
   CCListButtonsTop.swt
      CCButtons.swt
      CCRecordNav.swt
      CCTogglebar_drop.swt
      CCListButtonsTopRight.swt
   CCListBodyInverted.swt
   CCBottomApplet_NamedList.swt
```

Table 15 lists the mappable items for this template.

Table 15. Mappable Items

ID	Description
2	...
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
121	First
122	Previous
123	Next
124	Last
131	New

Table 15. Mappable Items

ID	Description
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
142-143	...
144	...
145	...
146	...
147	...
150-151	Control
160-164	Control
499	Record Title Row
501-520	Control
599	Save
1100	Outside Applet Help Text
1500	Required Legend

Applet List Message

SWT Filename: CCAppletListMessage.swt

Applet List Message, shown in Figure 20, is frequently used on home pages to emphasize breaking news or timely information. Each record displays a round bullet and provides a placeholder for a link and short descriptive text.

Figure 20. Applet List Message

The template supports a mappable title (90/184) and layout controls.

Includes Tree

```
CCAppletListMessage.swt
   CCApplet_Spacer.swt
   CCLayoutTitlePortal.swt
      CCApplet_Spacer.swt
      CCLayoutButtons.swt
      CCBottomApplet.swt
   CCTitle_Portal.swt
      CCLayoutButtons.swt
   CCListButtonsTopNoRecNav.swt
      CCButtons.swt
      CCListButtonsTopRight.swt
   CCListBodyBullet.swt
   CCBottomApplet.swt
```

Table 16 lists the mappable items for this template.

Table 16. Mappable Items

ID	Description
2	Back
90	Title
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
150-151	Control
157	Label
160-164	Control
184	Drilldown Title
203	MinimizeApplet
204	MaximizeApplet
207	MoveAppletUp
208	MoveAppletDown
211	ShowApplet

Table 16. Mappable Items

ID	Description
212	HideApplet
501	Field
502-511	Field
555	Label
599	Save
1100	Outside Applet Help Text

Applet List Portal

SWT Filename: CCAppletListPortal.swt

This is the standard list template, shown in Figure 21. It is to be used on portal pages. This list presents a title, layout controls, and an optional line of buttons beneath the title. ID90/184 supports a mappable title suitable for drilldown to a related view.

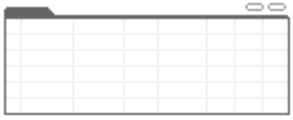


Figure 21. Applet List Portal

Includes Tree

```
CCAppletListPortal.swt
   CCApplet_Spacer.swt
   CCLayoutTitlePortal.swt
      CCApplet_Spacer.swt
      CCLayoutButtons.swt
```

```
   CCBottomApplet.swt
CCTitle_Portal.swt
   CCLayoutButtons.swt
CCListButtonsTopNoRecNav.swt
   CCButtons.swt
   CCListButtonsTopRight.swt
CCListHeaderNoSort.swt
CCListBodyNoRowHilite.swt
CCBottomApplet.swt
```

Table 17 lists the mappable items for this template.

Table 17. Mappable Items

ID	Description
2	Back
90	Title
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control

Table 17. Mappable Items

ID	Description
142-143	Control
144	Selected Row
145	Control
146	Save
147	Pick Control
150-151	Control
157	Label
160-164	Control
184	Drilldown Title
203	MinimizeApplet
204	MaximizeApplet
207	MoveAppletUp
208	MoveAppletDown
211	ShowApplet
212	HideApplet
501-540	Field
555	Label
599	Save
1100	Outside Applet Help Text

Applet List Portal (Graphical)

SWT Filename: CCAppletListPortalGraphical.swt

This list template, shown in Figure 22, can be used on portal pages. This list presents a title, layout controls, and an optional line of buttons beneath the title. ID 90/184 supports a mappable title suitable for drilldown to a related view. This is a specialized list template in that it can display a graphical applet title treatment. Map the applet image to ID 89. Map the applet title to ID 90.

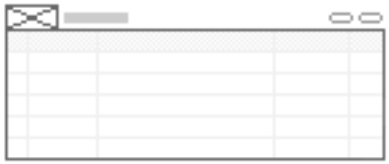


Figure 22. Applet List Portal (Graphical)

Includes Tree

```
CCAppletListPortalGraphical.swt
   CCApplet_Spacer.swt
   CCLayoutTitlePortal.swt
      CCApplet_Spacer.swt
      CCLayoutButtons.swt
      CCBottomApplet.swt
   CCTitle_PortalGraphical.swt
      CCLayoutButtons.swt
   CCListButtonsTopNoRecNav.swt
      CCButtons.swt
      CCListButtonsTopRight.swt
   CCListHeaderNoSort.swt
   CCListBodyNoRowHilite.swt
   CCBottomApplet.swt
```

Table 18 lists the mappable items for this template.

Table 18. Mappable Items

ID	Description
2	Back
89	Image
90	Title
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
142-143	Control
144	Selected Row
145	Control
146	Save
147	Pick Control
150-151	Control
157	Label
160-164	Control

Table 18. Mappable Items

ID	Description
184	Drilldown Title
203	MinimizeApplet
204	MaximizeApplet
207	MoveAppletUp
208	MoveAppletDown
211	ShowApplet
212	HideApplet
501-540	Field
555	Label
599	Save
1100	Outside Applet Help Text

Applet List Search Results

SWT Filename: CCAppletListSearchResults.swt

This applet, shown in Figure 23, defines the search results list found in the Search Center pane. To conserve vertical real estate the applet title is embedded in the button bar, to the right of the menu button.

Figure 23. Applet List Search Results

Includes Tree

```
CCAppletListSearchResults.swt
   CCButtons.swt
   CCRecordNav.swt
   CCListHeader.swt
   CCListBodySearchResults.swt
```

Table 19 lists the mappable items for this template.

Table 19. Mappable Items

ID	Description
2	Back
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
121	First
122	Previous
123	Next
124	Last
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control

Table 19. Mappable Items

ID	Description
142-143	Control
144	Select
145	Control
146	Control
147	...
501-540	...
598	...
599	Save

Applet List Totals (Base/Edit List)

SWT Filename: CCAppletListTotals_B_EL.swt

This is the standard list template for lists in base or edit-list modes. See Figure 24 for an example. A list can typically display between 7-10 visible columns. It is possible to map more visible columns, but this is not recommended as they may appear off screen. The template supports mapping up to 40 fields, but this is done so that you may mark the majority as available but hidden. Fields marked as such will not appear by default in the list but will appear in the columns displayed dialog.

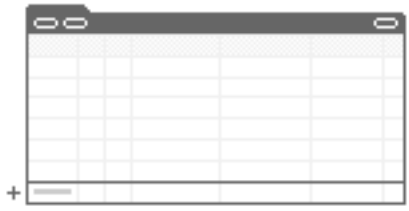


Figure 24. Applet List Totals

Includes Tree

```
CCAppletListTotals.swt
   CCApplet_NamedSpacer.swt
   CCTitle_Named.swt
      CCTitle.swt
   CCListButtonsTop.swt
      CCButtons.swt
      CCRecordNav.swt
      CCTogglebar_drop.swt
      CCListButtonsTopRight.swt
   CCListHeaderTotals.swt
   CCListBodyTotals.swt
   CCBottomApplet_NamedList.swt
```

Table 20 lists the mappable items for this template.

Table 20. Mappable Items

ID	Description
2	...
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
121	First
122	Previous
123	Next
124	Last

Table 20. Mappable Items

ID	Description
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
142-143	Control
144	Selected Row
145	Control
146	Save
147	Pick Control
150-151	Control
160-164	Control
199	Totals Label
501-520	...
599	Save
1100	Outside Applet Help Text
1500	Required Legend

Popup List

SWT Filename: CCPopupList.swt

This template defines the list treatment used in the Base or Edit List pop-up modes. See Figure 25 for an example.

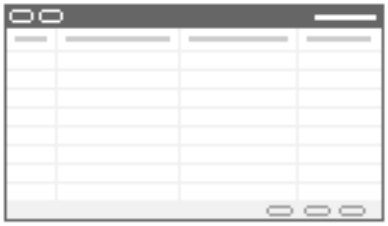


Figure 25. Popup List

Includes Tree

```
CCPopupList.swt
   CCStylesChoice.swt
   CCListButtonsTop.swt
      CCButtons.swt
      CCRecordNav.swt
      CCTogglebar_drop.swt
      CCListButtonsTopRight.swt
   CCListHeader.swt
   CCListBody.swt
   CCButtons_Popup.swt
```

Table 21 lists the mappable items for this template.

Table 21. Mappable Items

ID	Description
2	...
106	Query

Table 21. Mappable Items

ID	Description
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
121	First
122	Previous
123	Next
124	Last
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
142-143	Control
144	Selected Row
145	Control
146	Save
147	Pick Control
150-151	Control
152	OK
153	Cancel
154-158	Control
160-164	Control

Table 21. Mappable Items

ID	Description
501-540	Field
598	Save
599	Save

Applet Form 1 Column Light (Base/Edit/New)

SWT Filename: CCAppletForm1ColLight_B_E_N.swt.

The template is a lightweight one-column form template where the label appears above the field value, as shown in Figure 26. Fields support required field indicators. The template is designed for use in narrow columns such as on home page views. It supports the standard applications styles: Parent, Child, and Grandchild. Buttons appear at the bottom of this applet. Applet title is derived from the applet object's title property.

Figure 26. Applet Form 1 Column Light (Base/Edit/New)

Includes Tree

```
CCAppletForm1ColLight_B_E_N.swt

   CCApplet_NamedSpacer.swt

   CCTitle_Named.swt

      CCTitle.swt

   CCForm1ColBodyLight.swt
```

```
    dCCFormButtonsBottom.swt
    dCCButtons_Form.swt
CCBottomApplet_NamedForm.swt
```

Table 22 lists the mappable items for this template.

Table 22. Mappable Items

ID	Description
2	Back
91	Inside Applet Help Text
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-110	Control
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
151-155	Control
156	Control
157-158	Control
1301-1330	Required; Label; Field
1500	Required Legend

Applet Form 4 Column (Base)

SWT Filename: CCAppletForm4Col_B.swt

This is the standard four-column form template for forms in base mode, as shown in Figure 27. Fields can be mapped to up to four columns. Labels appear above field values. Validation errors appear at the top of the form. Instructional text can be added to ID 91; it spans all four columns. The form defines a large number of control placeholders, some spanning one column, some spanning two columns, and some spanning all four columns. The standard applet styles are supported.

Figure 27. Applet Form 4 Column (Base)

Includes Tree

```
CCAppletForm4Col_B.swt
   CCApplet_NamedSpacer.swt
   CCTitle_Named.swt
      CCTitle.swt
   CCFormButtonsTop.swt
      CCButtons.swt
      CCRecordNav.swt
      CCTogglebar_drop.swt
      CCFormButtonsTopRight.swt
   CCForm4ColBody.swt
```

```
CCBottomApplet_NamedForm.swt
```

Table 23 lists the mappable items for this template.

Table 23. Mappable Items

ID	Description
2	...
91	Inside Applet Help Text
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
121	First
122	Previous
123	Next
124	Last
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
150-152	Control
160-164	Control
599	Save
1001-1009	FormSection

Table 23. Mappable Items

ID	Description
1020	FormSection
1296-1300	Required; Label: 2-Column Wide Field
1301-1310	Required; Label; Field
1311-1315	Required; Label: 2-Column Wide Field
1316-1330	Required; Label; Field
1331-1335	Required; Label: 2-Column Wide Field
1336-1340	Required; 4-Column Wide Label; 4-Column Wide Field
1360-1374	Required; Label; Field
1500	Required Legend
1801-1810	Required; Label; Field
1816-1830	Required; Label; Field
1860-1874	Required; Label; Field
2001-2009	FormSection
2296-2300	Required; Label: 2-Column Wide Field
2301-2310	Required; Label; Field
2316-2330	Required; Label; Field
2331-2335	Required; Label: 2-Column Wide Field
2360-2374	Required; Label; Field
2801-2810	Required; Label; Field
2811-2815	Required; Label: 2-Column Wide Field
2816-2830	Required; Label; Field
2860-2874	Required; Label; Field

Applet Form 4 Column (Edit/New)

SWT Filename: CCAppletForm4Col_E_N.swt

This is the standard four-column form template for forms shown in edit or new mode, as shown in Figure 28. Fields can be mapped up to four columns. Labels appear above field values. Validation errors appear at the top of the form. Instructional text can be added to ID 91; it spans all four columns. The form defines a large number of control placeholders, some spanning one column, some spanning two columns, and some spanning all four columns. The standard applet styles are supported.

Figure 28. Applet Form 4 Column (Edit/New)

Includes Tree

```
CCAppletForm4Col_E_N.swt
   CCApplet_NamedSpacer.swt
   CCTitle_Named.swt
      CCTitle.swt
   CCFormButtonsTop.swt
      CCButtons.swt
      CCRecordNav.swt
      CCTogglebar_drop.swt
      CCFormButtonsTopRight.swt
   CCForm4ColBody.swt
```

```
CCBottomApplet_NamedForm.swt
```

Table 24 lists the mappable items for this template.

Table 24. Mappable Items

ID	Description
2	...
91	Inside Applet Help Text
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
121	First
122	Previous
123	Next
124	Last
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
150-152	Control
160-164	Control
599	Save
1001-1009	FormSection

Table 24. Mappable Items

ID	Description
1020	FormSection
1296-1300	Required; Label: 2-Column Wide Field
1301-1310	Required; Label; Field
1311-1315	Required; Label: 2-Column Wide Field
1316-1330	Required; Label; Field
1331-1335	Required; Label: 2-Column Wide Field
1336-1340	Required; 4-Column Wide Label; 4-Column Wide Field
1360-1374	Required; Label; Field
1500	Required Legend
1801-1810	Required; Label; Field
1816-1830	Required; Label; Field
1860-1874	Required; Label; Field
2001-2009	FormSection
2296-2300	Required; Label: 2-Column Wide Field
2301-2310	Required; Label; Field
2316-2330	Required; Label; Field
2331-2335	Required; Label: 2-Column Wide Field
2360-2374	Required; Label; Field
2801-2810	Required; Label; Field
2811-2815	Required; Label: 2-Column Wide Field
2816-2830	Required; Label; Field
2860-2874	Required; Label; Field

Applet Form 4-Col (No Record Nav)

SWT Filename: CCAppletForm4Col_NoRecNav.swt

This is the standard four-column form template for forms in edit or new mode, as shown in Figure 29. Fields can be mapped up to four columns. Labels appear above field values. Validation errors appear at the top of the form. Instructional text can be added to ID 91; it spans all four columns.

Figure 29. Applet Form 4-Col (No Record Nav)

The form defines a large number of control placeholders, some spanning one column, some spanning two columns, and some spanning all four columns. The standard applet styles are supported.

Unique to this four column form template is the absence of record navigation.

Includes Tree

```
CCAppletForm4Col_NoRecNav.swt

   CCApplet_NamedSpacer.swt

   CCTitle_Named.swt

      CCTitle.swt

   CCFormButtonsTopNoRecNav.swt

      CCButtons.swt

      CCRecordNav.swt

      CCTogglebar_drop.swt
```

```
    CCFormButtonsTopRight.swt
  CCForm4ColBody.swt
  CCBottomAppletPntr.swt
```

Table 25 lists the mappable items for this template.

Table 25. Mappable Items

ID	Description
2	...
91	Inside Applet Help Text
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
150-152	Control
160-164	Control
599	Save
1001-1009	FormSection
1020	FormSection
1296-1300	Required; Label: 2-Column Wide Field

Table 25. Mappable Items

ID	Description
1301-1310	Required; Label; Field
1311-1315	Required; Label: 2-Column Wide Field
1316-1330	Required; Label; Field
1331-1335	Required; Label: 2-Column Wide Field
1336-1340	Required; 4-Column Wide Label; 4-Column Wide Field
1360-1374	Required; Label; Field
1500	Required Legend
1801-1810	Required; Label; Field
1816-1830	Required; Label; Field
1860-1874	Required; Label; Field
2001-2009	FormSection
2296-2300	Required; Label: 2-Column Wide Field
2301-2310	Required; Label; Field
2316-2330	Required; Label; Field
2331-2335	Required; Label: 2-Column Wide Field
2360-2374	Required; Label; Field
2801-2810	Required; Label; Field
2811-2815	Required; Label: 2-Column Wide Field
2816-2830	Required; Label; Field
2860-2874	Required; Label; Field

Applet List Edit (Edit/New/Query)

SWT Filename: CCAppletList_E_N_Q.swt

This is the standard four-column form template for forms shown in edit, new and query mode. See Figure 30 for an example. Fields can be mapped to up to four columns. Labels appear above field values. Validation errors appear at the top of the form. Instructional text can be added to ID 91; it spans all four columns.

Figure 30. Applet List Edit (Edit/New/Query)

The form defines a large number of control placeholders, some spanning one column, some spanning two columns and some spanning all four columns. The standard applet styles are supported. It is possible to use other four-column form templates in place of this one. This template remains in the release set so that you may alter its layout and affect change on forms in one mode without affecting form layouts that appear in other modes.

Includes Tree

```
CCAppletList_E_N_Q.swt

   CCApplet_NamedSpacer.swt

   CCTitle_Named.swt

      CCTitle.swt

   CCFormButtonsTop.swt

      CCButtons.swt

      CCRecordNav.swt

      CCTogglebar_drop.swt

      CCFormButtonsTopRight.swt

   CCList4ColBody.swt

   CCBottomApplet_NamedForm.swt
```

Table 26 lists the mappable items for this template.

Table 26. Mappable Items

ID	Description
2	…
91	Inside Applet Help Text
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
121	First
122	Previous
123	Next
124	Last
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
150-152	Control
160-164	Control
599	Save
1001-1009	FormSection
1020	FormSection

Table 26. Mappable Items

ID	Description
1296-1300	Required; Label: 2-Column Wide Field
1301-1310	Required; Label; Field
1311-1315	Required; Label: 2-Column Wide Field
1316-1330	Required; Label; Field
1331-1335	Required; Label: 2-Column Wide Field
1336-1340	Required; 4-Column Wide Label; 4-Column Wide Field
1360-1374	Required; Label; Field
1500	Required Legend
1801-1810	Required; Label; Field
1816-1830	Required; Label; Field
1860-1874	Required; Label; Field
2001-2009	FormSection
2296-2300	Required; Label: 2-Column Wide Field
2301-2310	Required; Label; Field
2311-2315	Required; Label: 2-Column Wide Field
2316-2330	Required; Label; Field
2331-2335	Required; Label: 2-Column Wide Field
2360-2374	Required; Label; Field
2801-2810	Required; Label; Field
2816-2830	Required; Label; Field
2860-2874	Required; Label; Field

Applet Wizard

SWT Filename: CCAppletFormWizard.swt

This applet is used by SmartScript applets and application wizards. See Figure 31 for an example. Buttons appear at the bottom of the form, making sure that users move through a procedure before advancing to the next screen. Map the applet title to ID 90. This is done so that each step in your wizard can have its own title. Map outside applet text, such as the name of the running script, to ID 1100.

Figure 31. Applet Wizard

Includes Tree

```
CCAppletFormWizard.swt

   CCTitle_Mapped.swt

   CCForm1ColBody.swt

   CCButtons.swt
```

Table 27 lists the mappable items for this template.

Table 27. Mappable Items

ID	Description
2	Back
90	Title
91	Inside Applet Help Text
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control

Table 27. Mappable Items

ID	Description
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
184	Title
599	Save
1100	Outside Applet Help Text
1301-1350	Required; Label; Field
1500	Required Legend
2301-2350	Required; Label; Field

Error Page

SWT Filename: CCError.swt

This is the standard template for displaying system errors.

Includes Tree

```
CCError.swt
   CCHTML.swt
      CCStylesChoice.swt
   CCBottomApplet.swt
   CCHTMLFooter.swt
```

Table 28 lists the mappable items for this template.

Table 28. Mappable Items

ID	Description
1	…
2	…
4	…
102	…

Popup Form

SWT Filename: CCPopupForm.swt

This standard pop-up template defines the one-column form treatment used in the base or edit pop-up modes. See Figure 32 for an example.

Figure 32. Popup Form

Includes Tree

```
CCPopupForm.swt

   CCStylesChoice.swt

   CCButtons.swt

   CCButtons_Popup.swt
```

Table 29 lists the mappable items for this template.

Table 29. Mappable Items

ID	Description
2	Back
91	Inside Applet Help Text
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
152	OK
153	Cancel
154-158	Control
599	Save
1001-1009	FormSection
1090-1099	Required; Label; field
1100	Label With Rule
1101-1110	Required; Label; Field
1111-1115	Required; Label; Field

Table 29. Mappable Items

ID	Description
1150	Label With Rule
1151-1160	Required; Label; Field
1500	Required Legend
2001-2002	FormSection
2101-2110	Required; Label; Field
2111-2115	Required; Label; Field

SmartScript Player Applet (Player Only)

SWT Filename: CCSmartScriptPlayerApplet.swt

This is the standard applet definition for SmartScript applets. See Figure 33 for an example.

Figure 33. SmartScript Player Applet (Player Only)

NOTE: To encourage form completion, these applets display buttons at the bottom of the form.

Includes Tree

`CCSmartScriptPlayerApplet.swt`

Table 30 lists the mappable items for this template.

Table 30. Mappable Items

ID	Description
1	Finish Script
2	Cancel Script
3	Previous Section
4	Next Section
5	Save Script
6	Save Answers
1500	Required Label

Applet Tree 2

SWT Filename: CCAppletTree2.swt

This a standard tree template which displays applet tab and border treatment. See Figure 34 for an example.

Figure 34. Applet Tree 2

Includes Tree

```
CCAppletTree2.swt

   CCApplet_NamedSpacer.swt

   CCTitle_Named.swt
```

```
    CCTitle.swt
CCListButtonsTopNoRecNav.swt
CCButtons.swt
CCListButtonsTopRight.swt
```

Table 31 lists the mappable items for this template.

Table 31. Mappable Items

ID	Description
2	Back
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
150	Control
151	Control
160-164	Control
599	Save
1100	Outside Applet Help Text
1500	Required Legend

Applet Tree Marketing

SWT Filename: CCAppletTreeMarketing.swt

This is a specialized tree applet with tab and border treatment. The template incudes support for a toggle bar. See Figure 35 for an example.

Figure 35. Applet Tree Marketing

Includes Tree

`CCAppletTreeMarketing.swt`

`CCTitle.swt`

Table 32 lists the mappable items for this template.

Table 32. Mappable Items

ID	Description
2	...
101	Label
132	Control
133	Control
142-143	Control
201	Field
1500	Required Legend

Smart Script Player Applet (Tree Only)

SWT Filename: CCAppletTree.swt

This is a minimal tree applet without applet title or border. See Figure 36 for an example.

Figure 36. Smart Script Player Applet (Tree Only)

Includes Tree

`CCAppletTree.swt`

Table 33 lists the mappable items for this template.

Table 33. Mappable Items

ID	Description
	No mappable items

Applet Calendar Daily (Portal)

Swt Filename: CCAppletCalendarDailyPortal.swt

This template creates a condensed calendar suitable for use on home pages. It supports a graphical header mapped to ID 89. See Figure 37 for an example.

Figure 37. Applet Calendar Daily (Portal)

Includes Tree

```
CCAppletCalendarDailyPortal.swt
   CCLayoutTitlePortal.swt
      CCApplet_Spacer.swt
      CCLayoutButtons.swt
      CCBottomApplet.swt
   CCCalendarAppletTitleGraphical.swt
   CCApplet_Spacer.swt
   CCLayoutButtons.swt
```

Table 34 lists the mappable items for this template.

Table 34. Mappable Items

ID	Description
89	Image
90	Title
101	Label

Table 34. Mappable Items

ID	Description
102	Field
103	…
104	…
105	…
106	…
107	TimeZoneLabel
108	TimeZone
130	…
131-132	…
133	…
142	…
157	Label
158	GoToWeeklyView
159	GoToMonthlyView
184	DrillDown Title
203	MinimizeApplet
204	MaximizeApplet
207	MoveAppletUp
208	MoveAppletDown
211	ShowApplet
212	HideApplet
555	Label
999	GoToToday

eCalendar Daily Applet

SWT Filename: CCAppletCalendarDaily.swt

This is a standard daily calendar template. This template supports standard applet styles: Parent, Child, and Grandchild. See Figure 38 for an example.

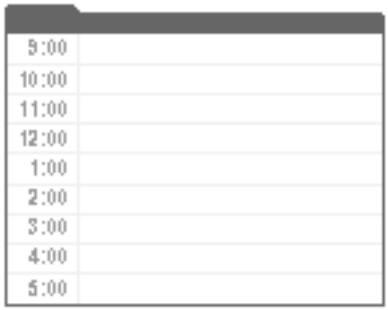


Figure 38. eCalendar Daily Applet

Includes Tree

```
CCAppletCalendarDaily.swt
   CCApplet_NamedSpacer.swt
   CCTitle_Named.swt
      CCTitle.swt
   CCBottomApplet.swt
```

Table 35 lists the mappable items for this template.

Table 35. Mappable Items

ID	Description
101	Label
102	Field
103	…

Table 35. Mappable Items

ID	Description
104	…
105	…
106	…
107	Label
108	Field
130	…
131-132	…
133	…
142	…
145	…
301-303	…
998	…
1500	Required Legend

eCalendar Monthly Applet

SWT Filename: CCAppletCalendarMonthly.swt

This is a standard monthly calendar template. This template supports the standard applet styles. Applet title should be mapped to ID 90. See Figure 39 for an example.

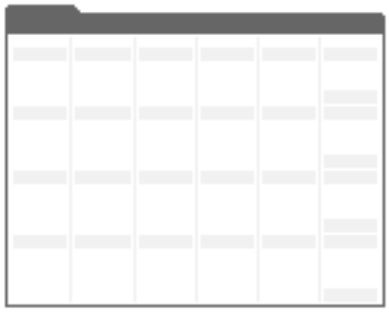


Figure 39. eCalendar Monthly Applet

Includes Tree

```
CCAppletCalendarMontly.swt
   CCApplet_NamedSpacer.swt
   CCTitle_Named.swt
      CCTitle.swt
   CCCalendarMonthly_weekday.swt
   CCCalendarMonthly_weekend.swt
   CCBottomApplet.swt
```

Table 36 lists the mappable items for this template.

Table 36. Mappable Items

ID	Description
101	Label
102	Field
103	...

Table 36. Mappable Items

ID	Description
104	…
105	…
106	…
107	Label
108	Field
130	…
131-132	…
133	…
141-142	…
142	…
145	…
301-306	…
998	…
1500	Required Legend

eCalendar Weekly Applet

SWT Filename: CCAppletCalendarWeekly.swt

This is the standard weekly calendar template. See Figure 40 for an example. Days of the week run down the side of the applet. Daily activities appear embedded beside them. This template supports the standard applet styles. Applet title should be mapped to ID 90.

Figure 40. eCalendar Weekly Applet

Includes Tree

```
CCAppletCalendarWeekly.swt
   CCApplet_NamedSpacer.swt
   CCTitle_Named.swt
      CCTitle.swt
   CCBottomApplet.swt
```

Table 37 lists the mappable items for this template.

Table 37. Mappable Items

ID	Description
101	Label
102	Field
103	…

Table 37. Mappable Items

ID	Description
104	…
105	…
106	…
107	Label
108	Field
130	…
131-132	…
133	…
141-142	…
142	…
145	…
301-303	…
998	…
1500	Required Legend

Service Calendar Applet

SWT Filename: CCAppletCalendarService.swt

This is the standard service calendar template. This template supports the standard applet styles. Ids 301-307 are used to map days of the weeks, which appear as column headers. See Figure 41 for an example.

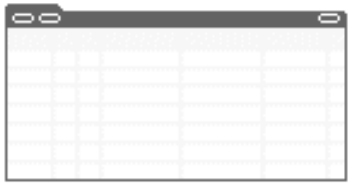


Figure 41. Service Calendar Applet

Includes Tree

```
CCAppletCalendarService.swt
   CCApplet_NamedSpacer.swt
   CCTitle_Named.swt
      CCTitle.swt
   CCBottomApplet.swt
```

Table 38 lists the mappable items for this template.

Table 38. Mappable Items

ID	Description
301-306	...
307	...
1500	Required Legend

Applet Chart

SWT Filename: CCAppletChart.swt

This is a standard chart template. This template supports the standard applet styles. See Figure 42 for an example.

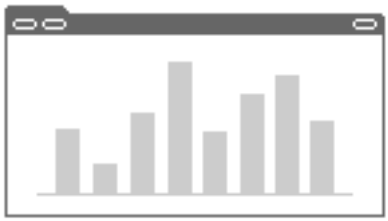


Figure 42. Applet Chart

Chart controls can be mapped to ID ranges 501-521.The chart itself is mapped to ID 599.

This applet can participate in a toggle applet relationship. The other toggle applets will appear in a drop-down list. The drop-down label is mapped to ID 2.

Includes Tree

```
CCAppletChart.swt
   CCApplet_NamedSpacer.swt
   CCTitle_Named.swt
      CCTitle.swt
   CCChartBasic.swt
      CCTogglebar_drop2.swt
   CCBottomApplet_NamedForm.swt
```

Table 39 lists the mappable items for this template.

Table 39. Mappable Items

ID	Description
2	…
501-520	…

Table 39. Mappable Items

ID	Description
551-555	...
599	Chart
1500	Required Legend

View Layouts

In the view diagrams shown in the following section, the gray areas represent applet regions where one or more applets can be placed (for example, see "View Template Descriptions" on page 448). Applets rendered on the Web expand horizontally to fit the column to which they have been assigned. The amount of displayed data determines how much vertical space an applet consumes.

In the view templates, each applet placeholder declares a style. The style declaration is evaluated at runtime and determines which color-scheme the applet will display and whether its title will display.

NOTE: In Tools, style declarations are not evaluated. Therefore, color schemes and applet titles may display inaccurately.

View Issues

- Mapping tree applet maps onto views other than view tree

  The tree applet and the associated target applet are not invoked in templates in the way standard applets are; therefore, you cannot map them like standard applets. Use View Tree or View Tree 2.

- Increasing the number of applets that show within a region

  Most regions use a `<swe:for-each>` loop to define how many applets can be mapped to a region. The `<swe:for each>` tag includes a count argument. By increasing the count argument, you can increase the number of applets that can show within a region.

- Sub-frame views

  View 25 – 75 (Framed) demonstrates sub-frame views. In this example, some applets appear in the left frame, and other applets appear in the right frame.

- Applet displays differently depending on where applet is placed in a view

  In the view templates, each applet placeholder declares a style. The style declaration is evaluated at runtime and determines which color-scheme the applet will display and whether its title will display.

  By applying styles at the view template level and coding the applet templates to evaluate styles, it is possible to reuse one repository applet object in many situations. This approach promotes code reuse and reduces the number of applet copies that must be maintained for a given application.

  Currently there are three applet styles: Parent, Child, and Grandchild. Review the applet visual reference to determine what each of these styles looks like.

View Template Descriptions

View 1 Over 2 Over 1

SWT Filename: CCView_1Over2Over1.swt

This is a standard view template. Applets in the first region consume the full window width. Applets in the second and third regions each consume 50 percent of the window width. Applets in the last region consume the full window width. See Figure 43 for an example.

Figure 43. 1 Over 2 Over 1

Includes Tree

```
CCView_1Over2Over1.swt
   CCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   CCHTMLFooter.swt
```

Table 40 lists the mappable items for this template.

Table 40. Mappable Items

ID	Description
101	Salutation Applet
102-106	Applet
201	Mini-Applet
202-206	Applet
302-306	Applet
402-406	Applet
502-506	Applet
602-606	Applet

View 25 - 50 – 25

SWT Filename: CCView_25_50_25.swt

This is a general-purpose view. Applets in the first column consume 25 percent of the window width. Applets in the second column consume 50 percent of the window width. Applets in the third column consume 25 percent of the window width. See Figure 44 for an example.

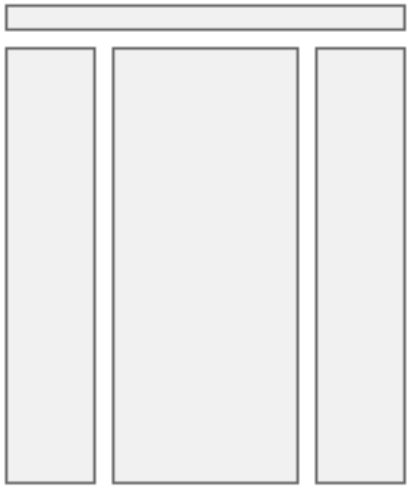


Figure 44. View 25 - 50 - 25

Includes Tree

```
CCView_25_50_25.swt
   CCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   CCHTMLFooter.swt
```

Table 41 lists the mappable items for this template.

Table 41. Mappable Items

ID	Description
101	Salutation Applet
102-111	Applet
201	Mini-Applet
202-211	Applet
302-311	Applet

View 25 – 75

SWT Filename: CCView_25_75.swt

This is standard view template. Applets in the first column consume 25 percent of the window width. Applets placed in the second column consume 75 percent of the window width. See Figure 45 for an example.

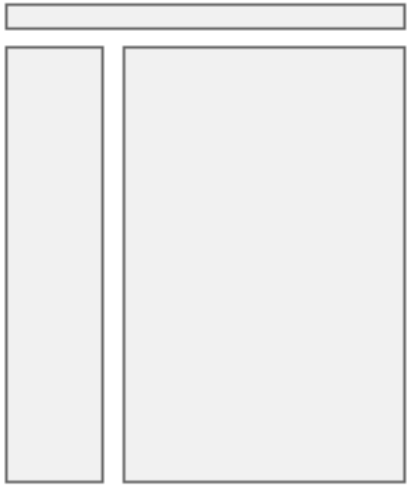


Figure 45. View 25 - 75

Includes Tree

```
CCView_25_75.swt
   CCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
      CCHTMLFooter.swt
```

Table 42 lists the mappable items for this template.

Table 42. Mappable Items

ID	Description
101	Salutation Applet
102-106	Applet
108-112	Applet
201	Mini-Applet
202-211	Applet

View 25 – 75 Framed

SWT Filename: CCView_25_75_Framed.swt

This is a standard view template. Applets placed in the first column consume 25 percent of the window width. Applets in the second column consume 75 percent of the window width. The first and second columns reside in separate frames. See Figure 46 for an example.

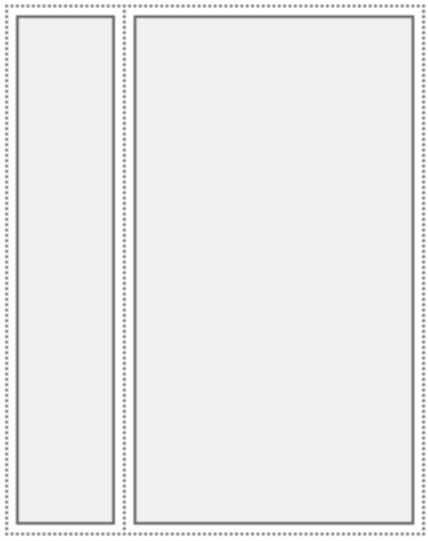


Figure 46. View 25 - 75 (Framed)

Includes Tree

```
CCView_25_75.swt
   CCHTMLHeader.swt
      CCStylesChoice.swt
   CCHTMLFooter.swt
   CCThreadbar.swt
```

Table 43 lists the mappable items for this template.

Table 43. Mappable Items

ID	Description
101	Salutation Applet
102-106	Applet
108-112	Applet
201	Mini-Applet
202-211	Applet

View 25 75 Framed 2

SWT Filename: CCView_25_75_Framed2.sw

This is a standard view template. Applets in the first column consume 25 percent of the window width. Applets in the second column consume 75 percent of the window width. The first column is broken into two frames with support for one applet in each frame. The second column is in its own frame. See Figure 47 for an example.

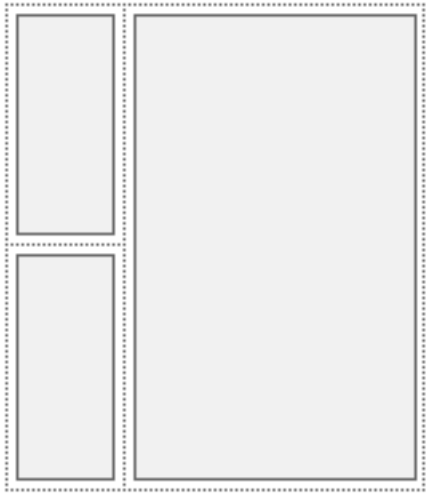


Figure 47. View 25 75 Framed 2

Includes Tree

```
CCView_25_75Framed2.swt
   CCHTMLHeader.swt
      CCStylesChoice.swt
   CCHTMLFooter.swt
   CCThreadbar.swt
```

Table 44 lists the mappable items for this template.

Table 44. Mappable Items

ID	Description
102-103	Applet
201	Mini-Applet
202-211	Applet

View 50 – 50

SWT Filename: CCView_50_50.swt

This is a standard view template. Applets in the first column consume 50 percent of the window width. Applets in the second column consume 50 percent of the window width. See Figure 48 for an example.

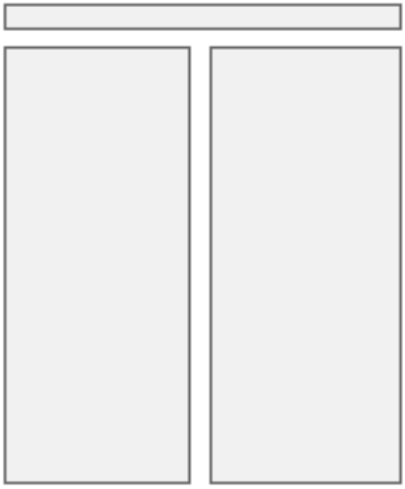


Figure 48. View 50 - 50

Includes Tree

```
CCView_50_50.swt
   CCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   CCHTMLFooter.swt
```

Table 45 lists the mappable items for this template.

Table 45. Mappable Items

ID	Description
101	Salutation Applet
102-106	Applet
201	Mini-Applet
202-206	Applet

View 66 – 33

SWT Filename: CCView_66_33.swt

This is a standard view template. Applets in the first column consume 66 percent of the window width. Applets in the second column consume 33 percent of the window width. See Figure 49 for an example.

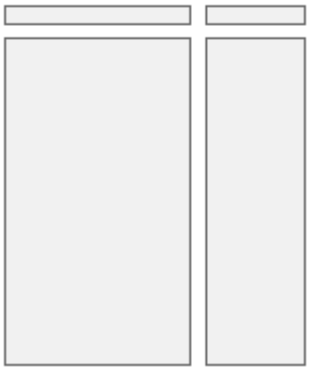


Figure 49. View 66 - 33

Includes Tree

```
CCView_66_33.swt
   CCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   CCHTMLFooter.swt
```

Table 46 lists the mappable items for this template.

Table 46. Mappable Items

ID	Description
101	Salutation Applet
102-121	Applet

Table 46. Mappable Items

ID	Description
201-220	Applet
901	Layout Controls

View Admin 1

SWT Filename: CCViewAdmin1.swt

This template displays subviews as tabs across the top of the view. See Figure 50 for an example. It is useful for admin views that need to display nonrelated views that are not easily categorized.

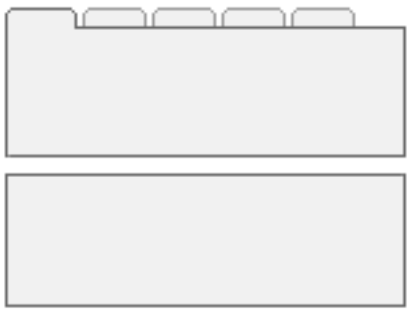


Figure 50. View Admin 1

Includes Tree

`CCViewAdmin1.swt`

```
CCHTMLHeader.swt
   CCStylesChoice.swt
CCThreadbar.swt
CCSubViewbar_Tabs.swt
   CCApplet_Spacer.swt
CCHTMLFooter.swt
```

Table 47 lists the mappable items for this template.

Table 47. Mappable Items

ID	Description
5	Child Applet With Pointer
6	Child Applet
7-9	Grandchild Applet
13-15	Grandchild Applet
201	Mini-Applet

View Admin 1 (Grandchild Indented)

SWT Filename: CCViewAdmin1_GrndchldIndnt.swt

This is similar to View Admin 1 except that the second and subsequent applets appear indented. This is useful for demonstrating a hierarchical relationship and for admin views that need to display nonrelated views that are not easily categorized. See Figure 51 for an example.

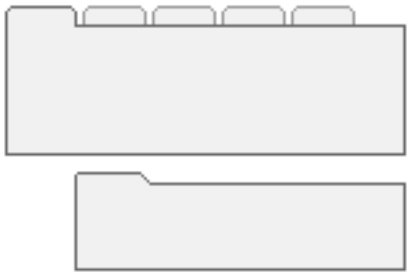


Figure 51. View Admin 1 (Grandchild Indented)

Includes Tree

```
CCViewAdmin1_GrndchldIndnt.swt

   CCHTMLHeader.swt

      CCStylesChoice.swt

   CCThreadbar.swt

   CCSubViewbar_Tabs.swt

      CCApplet_Spacer.swt

   CCHTMLFooter.swt
```

Table 48 lists the mappable items for this template.

Table 48. Mappable Items

ID	Description
5	Child Applet With Pointer
6	Child Applet
7	Grandchild Applet With Pointer

Table 48. Mappable Items

ID	Description
8-9	Child or Grandchild Applet
10-12	Grandchild Applet
13-15	Grandchild Applet
201	Mini-Applet

View Basic

SWT Filename: CCViewBasic.swt

This is a standard view template. All applets consume the full window width and appear stacked on top of each other. See Figure 52 as an example.

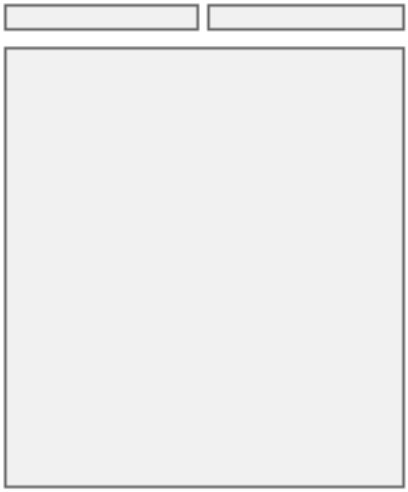


Figure 52. View Basic

Includes Tree

```
CCViewBasic.swt
   CCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   CCHTMLFooter.swt
```

Table 49 lists the mappable items for this template.

Table 49. Mappable Items

ID	Description
1-20	Applet
101	Salutation Applet
201	Mini-Applet
901	Layout Controls

View Catalog Admin

SWT Filename: CCViewCatalog.swt

This is a specialized view template to be used in catalog views only. See Figure 53 for an example.

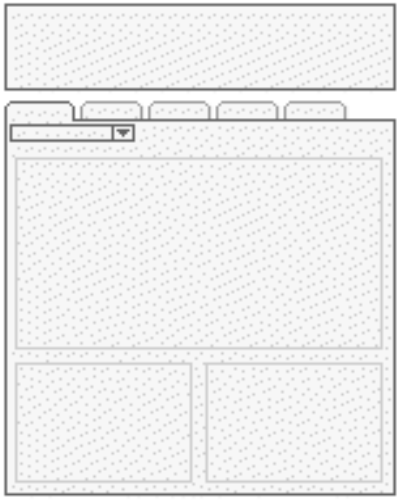


Figure 53. View Catalog Admin

Includes Tree

```
CCViewCatalog.swt
   CCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   CCViewbar_Tabs.swt
   CCHTMLFooter.swt
```

Table 50 lists the mappable items for this template.

Table 50. Mappable Items

ID	Description
1	Parent Applet
2	Grandchild Applet
3	Grandchild Applet
4-5	Child Applet
6-7	Grandchild Applet
8-9	Grandchild Applet
201	Mini-Applet

View Detail

SWT Filename: CCViewDetail.swt and CCViewDetail_ParentPntr.swt

This is a standard view template. It shows parent applet, noncontext views presented as tabs, categorized subviews presented in a drop-down, child applet, and multiple grandchild applets. See Figure 54 for an example.

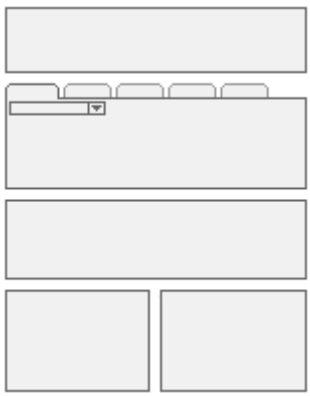


Figure 54. View Detail

Includes Tree

```
CCViewDetail.swt
   CCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   CCViewbar_Tabs.swt
   CCSubViewbar_Drop.swt
   CCHTMLFooter.swt
```

Table 51 lists the mappable items for this template.

Table 51. Mappable Items

ID	Description
1	Parent Applet
2	...
3-5	Child or Grandchild Applet
6-7	Grandchild Applet
8-9	Grandchild Applet
201	Mini-Applet

View Detail (Grandchild Indented)

SWT Filename: CCViewDetail_GrandchldIndnt.swt

This is a standard view template. It shows parent applet, noncontext views presented as tabs, a child applet, categorized subviews presented as a drop-down list, and multiple grandchild applets. See Figure 55 for an example.

Figure 55. View Detail (Grandchild with Indent)

NOTE: Grandchild applets appear indented to convey hierarchy.

Includes Tree

```
CCViewDetail_GrndchldIndnt.swt

   CCHTMLHeader.swt

      CCStylesChoice.swt

   CCThreadbar.swt

   CCViewbar_Tabs.swt

   CCSubViewbar_Drop.swt

   CCHTMLFooter.swt
```

Table 52 lists the mappable items for this template.

Table 52. Mappable Items

ID	Description
1	Parent Applet
2	...
3-5	Child or Grandchild Applet
6-7	Grandchild Applet
8-9	Grandchild Applet
201	Mini-Applet

View Detail 2

SWT Filename: CCViewDetail2.swt

This is a standard view template. It shows a parent applet, noncontext views presented as tabs, a child applet, categorized subviews presented as tabs, and multiple grandchild applets. See Figure 56 for an example.

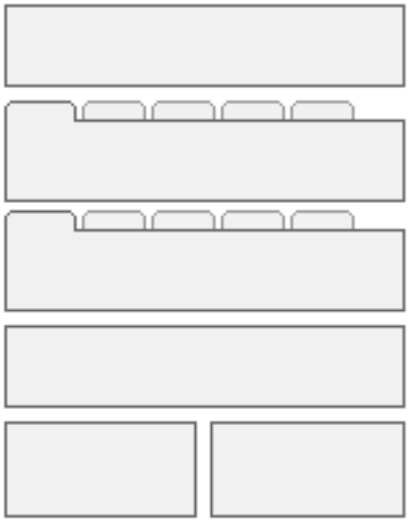


Figure 56. View Detail 2

Includes Tree

```
CCViewDetail2.swt
   CCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   CCViewbar_Tabs.swt
   CCSubViewbar_Drop.swt
      CCApplet_Spacer.swt
   CCHTMLFooter.swt
```

Table 53 lists the mappable items for this template.

Table 53. Mappable Items

ID	Description
1	Parent Applet
2	Child Applet
3	Grandchild Applet
4-5	Grandchild Applet
6-7	Grandchild Applet
8-9	Grandchild Applet
201	Mini-Applet

View Detail 2 (Grandfather Indent)

SWT Filename: CCViewDetail2_GrndchldIndnt.swt

This is a standard view template. It shows parent applet, noncontext views presented as tabs, a child applet, categorized subviews presented as tabs, and multiple grandchild applets. See Figure 57 for an example.

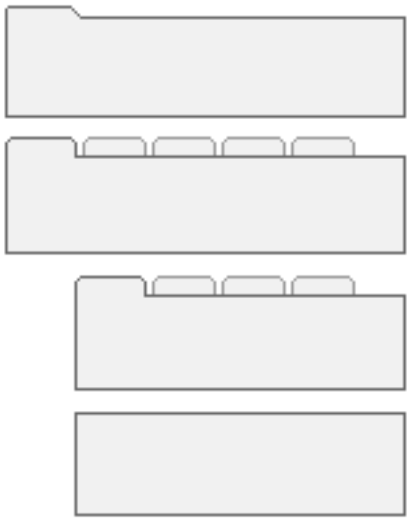


Figure 57. View Detail 2 (Grandfather Indent)

NOTE: Grandchild applets appear indented to convey hierarchy.

Includes Tree

```
CCViewDetail2_GrndchldIndnt.swt
   CCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   CCViewbar_Tabs.swt
   CCSubViewbar_Tabs.swt
      CCApplet_Spacer.swt
```

```
CCHTMLFooter.swt
```

Table 54 lists the mappable items for this template.

Table 54. Mappable Items

ID	Description
1	Parent Applet
2	Child Applet
3	Grandchild Applet
4-5	Child or Grandchild Applet
6-7	Grandchild Applet
8-9	Grandchild Applet
201	Mini-Applet

View Detail 3

SWT Filename: CCViewDetail3.swt

This is a specialized view template. It shows all views as tabs. The child applet appears beneath these tabs. Categorized subviews are presented in a drop-down list, and multiple grandchild applets appear beneath the child applet. It is useful for admin views that display a collection of views irrespective of grouping and visibility rules. See Figure 58 for an example.

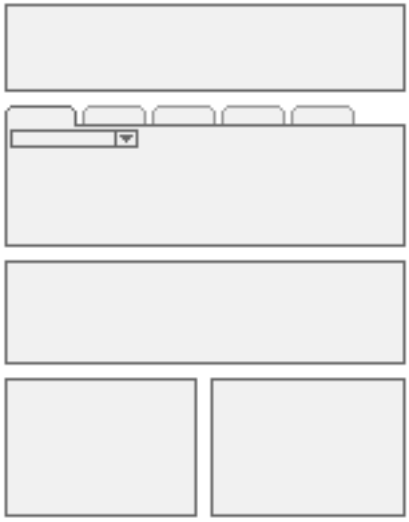


Figure 58. View Detail 3

Includes Tree

```
CCViewDetail3.swt
   CCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   CCViewbarAll_Tabs.swt
   CCSubviewbar_Drop.swt
   CCHTMLFooter.swt
```

Table 55 lists the mappable items for this template.

Table 55. Mappable Items

ID	Description
1	Parent Applet
2	...
3-5	Grandchild Applet
6-7	Grandchild Applet
8-9	Child or Grandchild Applet
201	Mini-Applet

View Detail 3 (Grandchild Indented)

SWT Filename: CCViewDetail3_GrndchldIndnt.swt

It shows all views as tabs. The parent applet appears beneath these tabs. The child applet appears beneath the categorized view tabs. Any grandchild applets appear beneath the child applet. See Figure 59 for an example.

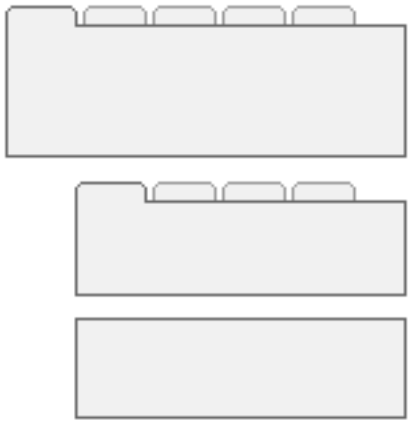


Figure 59. View Detail 3 (Grandchild Indented)

NOTE: Child and grandchild applets appear indented to convey hierarchy.

Includes Tree

```
CCViewDetail3_GrndchldIndnt.swt

   CCHTMLHeader.swt

      CCStylesChoice.swt

   CCThreadbar.swt

   CCViewbarAll_Tabs.swt

   CCSubViewbar_Drop.swt

   CCHTMLFooter.swt
```

Table 56 lists the mappable items for this template.

Table 56. Mappable Items

ID	Description
1	Parent Applet
2	...
3-5	Grandchild Applet
6-7	Child or Grandchild Applet
8-9	Child or Grandchild Applet
201	Mini-Applet

View Detail 3 Multi Child

SWT Filename: CCViewDetail3MultiChild.swt

This is a specialized view template. It shows all views as tabs. The child applet appears beneath these tabs. Categorized subviews are presented in a drop-down list, and multiple grandchild applets appear within a bounding box. It is useful for admin views that need to display a collection of views irrespective of grouping and visibility rules. See Figure 60 for an example of the View Detail 3 Multi Child template.

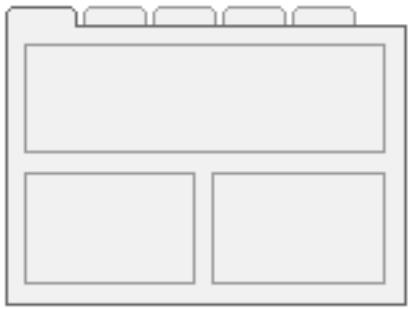


Figure 60. View Detail 3 Multi Child

Includes Tree

```
CCViewDetail3MultiChild.swt
   CCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   CCViewbarAll_Tabs.swt
   CCSubViewbar_Drop.swt
   CCHTMLFooter.swt
```

Table 57 lists the mappable items for this template.

Table 57. Mappable Items

ID	Description
1	Parent Applet
2	...
3-5	Grandchild Applet
6-7	Child or Grandchild Applet
8-9	Child or Grandchild Applet
201	Mini-Applet

View Detail Multi-Child

SWT Filename: CCViewDetailMultiChld.swt

This is a standard view template. It shows a parent, noncontext views presented as tabs, categorized subviews presented in a drop-down, a child applet, and multiple grandchild applets. See Figure 61 for an example.

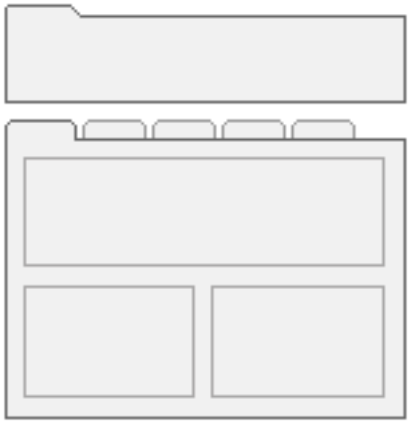


Figure 61. View Detail Multi-Child

NOTE: The child and grandchild applets appear inside a bounding box.

Includes Tree

```
CCViewDetailMultiChild.swt

   CCHTMLHeader.swt

      CCStylesChoice.swt

   CCThreadbar.swt

   CCViewbar_Tabs.swt

   CCSubViewbar_Drop.swt

   CCHTMLFooter.swt
```

Table 58 lists the mappable items for this template.

Table 58. Mappable Items

ID	Description
1	Parent Applet
2	...
3-5	Grandchild Applet
6-7	Child or Grandchild Applet
8-9	Child or Grandchild Applet
10	Child or Grandchild Applet
11	Child or Grandchild Applet
201	Mini-Applet

View Search

SWT Filename: CCView_Search.swt

This is a specialized view template used for search center. Applets consume the full window width. See Figure 62 for an example.

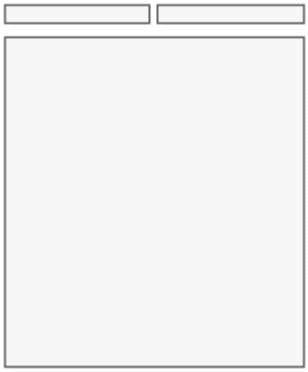


Figure 62. View Search

Includes Tree

```
CCView_Search.swt

   CCStylesChoice.swt

   CCHTMLFooter.swt
```

Table 59 lists the mappable items for this template.

Table 59. Mappable Items

ID	Description
1-5	Applet
11-13	Applet

View Tree

SWT Filename: CCViewTree.swt

This view template supports a two-column format where the first column is 25 percent of window width and contains the tree applet. The second column is 75 percent of window width and contains the tree's target list applet. See Figure 63 for an example.

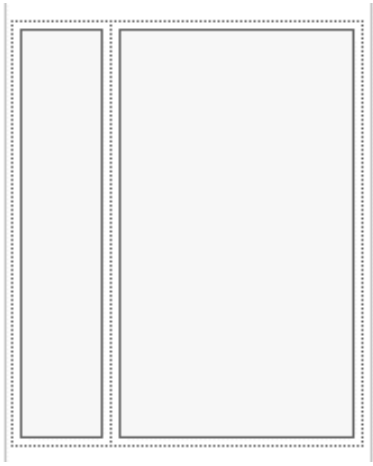


Figure 63. View Tree

The view also supports noncontext views as tabs.

Includes Tree

```
CCViewTree.swt
   CCHTMLHeader.swt
      CCStylesChoice.swt
   CCHTMLFooter.swt
   CCThreadbar.swt
   CCViewbar_Tabs.swt
   CCSubViewbar_Drop.swt
```

Table 60 lists the mappable items for this template.

Table 60. Mappable Items

ID	Description
1	Applet
2	...
3	Applet
4	Bottom Applet
201	Mini-Applet

View Tree 2

SWT Filename: CCViewTree2.swt

This view template supports a two-column format where the first column is 25 percent of the window width and contains the tree applet. The second column is 75 percent of the window width and contains the tree's target list applet. See Figure 64 for an example.

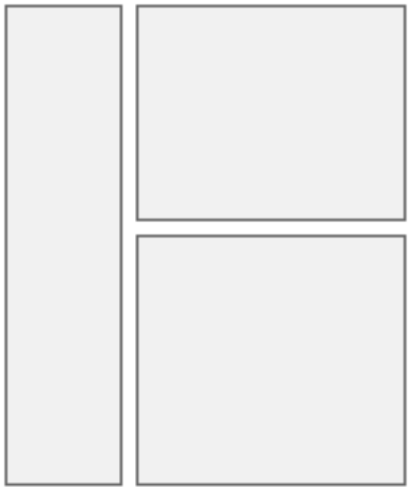


Figure 64. View Tree 2

The view also supports noncontext views as tabs.

Includes Tree

```
CCViewTree2.swt

   CCHTMLHeader.swt

      CCStylesChoice.swt

   CCHTMLFooter.swt

   CCThreadbar.swt
```

Table 61 lists the mappable items for this template.

Table 61. Mappable Items

ID	Description
1	Applet
3	Applet
201	Mini-Applet

Page Container Templates

Employee applications require frames. The employee container templates create frames for the banner, screen bar, view bar, and content. Some applications may also display frames for a toolbar and a message bar. To understand how the frames are organized and what conditions control their display, see the following template called CCPageContainer.swt.

Page Container

SWT Filename: CCPageContainer.swt

The page container template defines the setup for all frames within the application. Standard frames include banner, screenbar, viewbar, and content. Optional frames include toolbar, message bar, and dashboard. Display of optional frames is evaluated at runtime by calling standard UI methods stored in the repository. The page container supports the mapping of page items. These items actually display within frames that the page container references, but Tools expects them to be mapped to the page container.

Includes Tree

```
CCPageContainer.swt
   CCStylesChoice.swt
   CCFrameToolbar.swt
      CCStylesChoice.swt
      CCFrameBanner.swt
      CCScreenbar_Tabs.swt
   CCFrameViewbar.swt
      CCStylesChoice.swt
   CCFrameworkContent_Logic.swt
```

```
   CCFrameContent_VSD.swt
   CCFrameContent_VS.swt
   CCFrameContent_VD.swt
   CCFrameContent_V.swt
CCFrameworkMsgbar.swt
   CCStylesChoice.swt
CCFrameScreenbar.swt
CCStylesChoice.swt
CCFrameBanner.swt
CCScreenbar_Tabs.swt
```

Table 62 lists the mappable items for this template.

Table 62. Mappable Items

ID	Description
1	Show Label
11-15	...
21	...
22	...
23	...
33-34	Control
35	...
36-37	Control
38	Control

CC Container Page Logic

SWT Filename: CCFrameContent_Logic.swt

This is the Logical frameset manager. It is responsible for testing preferences and passing in the correct logical frameset.

Includes Tree

```
CCFramesetContent_Logic.swt

   CCFrameContent_VSD.swt

   CCFrameContent_VS.swt

   CCFrameContent_VD.swt

   CCFrameContent_V.swt
```

Specialized Applet Templates

Applet Advanced Search

SWT Filename: CCAppletSearchAdvanced.swt

Includes Tree:
`CCAppletSearchAdvanced.swt`

```
CCFormSearch.swt
```

Table 63. Mappable Items

ID	Description
91	Inside Applet Help Text
132	Menu
133	New
141-142	Control
143	Control
1101-1130	Label; Field

Applet Dashboard

SWT Filename: CCAppletDashboard.swt

This the standard dashboard applet. The dashboard applet is a lightweight form used to show customer context in call center applications. It supports three rows of up to four columns of information. Labels appear above fields. ID 211 can be used to map a button that will close the dashboard. See Figure 65 for an example of the Applet Dashboard template.

Figure 65. Applet Dashboard

Includes Tree

```
CCAppletDashboard.swt
```

Table 64 lists the mappable items for this template.

Table 64. Mappable Items

ID	Description
211	Hide
1200	Label
1201	Label
1202	Label
1300	Field
1301	Field
1302	Field
1700	Label
1701	Label
1702	Label
1800	Field
1801	Field
1802	Field
2200	Label
2201	Label
2202	Label
2300	Field
2301	Field
2302	Field
2700	Label
2701	Label
2702	Label

Table 64. Mappable Items

ID	Description
2800	Field
2801	Field
2802	Field
2901	Button
2902	Button

Applet Find

SWT Filename: CCAppletSearchFind.swt

The bottom of the search applet is used to show fields available for search through applet query.

Includes Tree

```
CCAppletSearchFind.swt

   CCFormSearch.swt
```

Table 65 lists the mappable items for this template.

Table 65. Mappable Items

ID	Description
91	Inside Applet Help Text
132	Menu
133	New
141-142	Control
143	Control
1101-1130	Label; Field

Applet Form Grid Layout

SWT filename: CCAppletFormGridLayout.swt

This is the applet Web template that supports grid-based layout of controls on form applets. It uses `<swe:form-applet-layout>` tag as a placeholder for all controls on a form applet, providing you with the ability to modify the layout of the controls using the Web Layout Editor in Siebel Tools.

Includes Tree

```
CCAppletFormGridLayout.swt

CCApplet_NamedSpacer.swt

   CCTitle_Named.swt

      CCFormButtonsTop.swt
```

Table 66 lists the controls that can be mapped to the header and footer regions of the Applet Form Grid Layout template.

Table 66. Mappable Items

ID	Description
2	...
91	Inside Applet Help Text
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
121	First
122	Previous
123	Next
124	Last
131	New

Table 66. Mappable Items

ID	Description
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
150-152	Control
160-164	Control
599	Save

Applet Form Search Top

SWT Filename: CCAppletFormSearchTop.swt

This applet is used in Search Center. It defines the top portion of the Search Center, where the user defines the search from the drop-down box choices. The Search Center title should be mapped to ID 90. The control to hide the Search Center frame should be mapped to ID 141.

Includes Tree

```
CCAppletFormSearchTop.swt

   CCFormSearch.swt
```

Table 67 lists the mappable items for this template.

Table 67. Mappable Items

ID	Description
90	...
91	Inside Applet Help Text

Table 67. Mappable Items

ID	Description
141	Hide Icon
1101-1130	Label; Field

Applet Items Displayed

SWT Filename: CCAppletItemsDisplayed.swt

This is a specialized template used to show the columns displayed dialog, which is available on most lists through the applet menu. It includes specialized code, which prevents it from being used outside this context.

Includes Tree

```
CCAppletItemsDisplayed.swt

   CCFormSearch.swt
```

Table 68 lists the mappable items for this template.

Table 68. Mappable Items

ID	Description
1	Save
2	Reset
3	Cancel
10	Available Items Label
11	Available Items Combobox
12	Available Items Hidden Field
20	Move Item to Selected
21	Move All to Selected
22	Move Item to Available
23	Move All to Available

Table 68. Mappable Items

ID	Description
30	Selected Items Label
31	Selected Items Combobox
32	Selected Items Hidden Field
40	Move Item to Top
41	Move Item Up
42	Move Item Down
43	Move Item to Bottom
91	Inside Applet Help Text
100	...
1100	Outside Applet Help Text

Applet Popup Form Grid Layout

SWT filename: CCAppletPopupFormGridLayout.swt

This is the applet Web template that supports grid-based layout of controls for popup applets. It uses `<swe:form-applet-layout>` tag as a placeholder for all controls on a Popup form applet, providing you with the ability to modify the layout of the controls using the Web Layout Editor in Siebel Tools.

Includes Tree

```
CCAppletPopupFormGridLayout.swt

   CCApplet_NamedSpacer.swt

   CCButtons.swt

      CCButtons_Popup.swt
```

The controls listed in Table 69 can be mapped to the header and footer regions of the Applet Popup Form Grid Layout template.

Table 69. Mappable Items

ID	Description
2	Back
91	Inside Applet Help Text
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
152	OK
153	Cancel
154-158	Control
599	Save

Applet Salutation

SWT Filename: CCAppletSalutation.swt

This is the standard salutation applet for use on home pages where a personalized greeting is desired. See Figure 66 for an example.

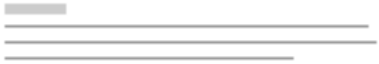


Figure 66. Applet Salutation

Includes Tree

```
CCAppletSalutation.swt

   CCApplet_Spacer.swt
```

Table 70 lists the mappable items for this template.

Table 70. Mappable Items

ID	Description
1	Salutation

Applet Salutation (Graphical)

SWT Filename: CCAppletSalutationGraphical.swt

This is a specialized salutation applet used on home pages where a personalized greeting is desired. The applet has a placeholder for an image (ID 89). See Figure 67 for an example.

Figure 67. Applet Salutation (Graphical)

Includes Tree

```
CCAppletSalutationGraphical.swt

   CCApplet_Spacer.swt
```

Table 71 lists the mappable items for this template.

Table 71. Mappable Items

ID	Description
1	Salutation
89	Image

Applet Screen Links

SWT Filename: CCAppletScreenLinks.swt

This template can be used to manually map controls such as GoToView links to create a table of contents for underlying information.

Includes Tree

```
CCAppletScreenLinks.swt
   CCApplet_Spacer.swt
   CCTitle.swt
   CCFormButtonsTop.swt
      CCButtons.swt
      CCRecordNav.swt
      CCTogglebar_drop.swt
      CCFormButtonsTopRight.swt
   CCScreenLinks.swt
   CCBottomApplet.swt
```

Table 72 lists the mappable items for this template.

Table 72. Mappable Items

ID	Description
2	...
91	Inside Applet Help Text
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
121	First
122	Previous
123	Next
124	Last
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
150-152	Control
160-164	Control
599	Save
1100	Group Label
1101-1120	Link

Table 72. Mappable Items

ID	Description
1200	Group Label
1201-1220	Link
1300	Group Label
1301-1320	Link
1400	Group Label
1401-1420	Link
1500	Required Legend
2100	Group Label
2101-2120	Link
2200	Group Label
2201-2220	Link
2300	Group Label
2301-2320	Link
2400	Group Label
2401-2420	Link

Applet Send Mail

SWT Filename: CCAppletSendMail.sw

This is a specialized applet for creating the send mail pop-up list.

Includes Tree

```
CCAppletSendMail.swt

   CCPopupButtonsBottom.swt
```

Table 73 lists the mappable items for this template.

Table 73. Mappable Items

ID	Description
152	OK
153	Cancel
154-158	Control
300-301	Icon
1200	From: Label
1201	To: Label
1202	Cc: Label
1203	Bcc Field
1204	Subject: Label
1205	Body: Label
1206	Optional Label
1207	Attachments: Label
1300	From Field
1301	To field
1302	CC Field
1303	Bcc Field
1304	Subject Field
1305	Templates Field
1306	Body Field
1307	Attachments Field
1400	Optional Control

Table 73. Mappable Items

ID	Description
1401	AB Control
1404	Control

Applet Send Mail Pick

SWT Filename: CCAppletSendEmailPick.swt

This is a specialized applet used in the selection of email recipients.

Includes Tree

```
CCAppletSendEmailPick.swt

   CCStylesChoice.swt

   CCListBody.swt

   CCPopupButtonsBottom.swt
```

Table 74 lists the mappable items for this template.

Table 74. Mappable Items

ID	Description
142-143	Control
144	Selected Row
145	Control
146	Save
147	Pick Control
152	OK
153	Cancel
154-158	Control
501-540	Field

Table 74. Mappable Items

ID	Description
598	Save
1200	Label

eActivityGanttChart Applet

SWT Filename: CCApplet.swt

This is one of several Gantt chart templates. It includes specialized Gantt code, which prevents this template from being used outside the context of Gantt applets.

Includes Tree

```
CCAppletActivityGanttChart.swt
   CCApplet_NamedSpacer.swt
   CCGanttAppletTitle.swt
   CCBottomApplet.swt
   CCBottomApplet_NamedList.swt
```

Table 75 lists the mappable items for this template.

Table 75. Mappable Items

ID	Description
301	...
302	...
303-304	Control
306-314	Label; Control
405	Control
2101-2130	...

EGantt Chart Applet

SWT Filename: CCAppletGanttChart.swt

This is the standard Gantt chart template upon which other Gantt chart templates are based. The template supports drag-and-drop in IE5x. The template accepts optional controls mapped to ranges 306-314.

Includes Tree

```
CCAppletGanttChart.swt
   CCGanttAppletTitle.swt
   CCBottomApplet.swt
   CCBottomAppletPntr.swt
```

Table 76 lists the mappable items for this template.

Table 76. Mappable Items

ID	Description
301	...
302	...
303-304	Control
306-314	Label; Control
405	Control
2101-2130	...

eGanttChart Applet (Portal)

SWT Filename: CCAppletGanttChartPortal.swt

This is similar to the standard Gantt chart templates; this template is used on portal pages where layout controls may be needed. The layout controls use the standard mappings found in the range 203-212. The template also supports a mappable title, useful for creating a drilldown to a related view.

Includes Tree

```
CCAppletGanttChartPortal.swt
   CCLayoutTitlePortal.swt
      CCApplet_Spacer.swt
      CCLayoutButtons.swt
      CCBottomApplet.swt
   CCApplet_Spacer.swt
   CCTitle_Portal.swt
      CCLayoutButtons.swt
   CCBottomApplet.swt
```

Table 77 lists the mappable items for this template.

Table 77. Mappable Items

ID	Description
90	Title
157	Label
184	DrillDown Title
203	MinimizeApplet
204	MaximizeApplet
207	MoveAppletUp
208	MoveAppletDown
211	ShowApplet
212	HideApplet
301	...
302	...
303-304	Control

Table 77. Mappable Items

ID	Description
306-314	Label; Control
405	Control
555	Label
2101-2130	...

Search Applet

SWT Filename: CCAppletSearchBasic.swt

This is the search applet bottom for basic search in the search center frame.

Includes Tree

```
CCAppletSearchBasic.swt

   CCFormSearch.swt
```

Table 78 lists the mappable items for this template.

Table 78. Mappable Items

ID	Description
91	Inside Applet Help Text
132	Menu
133	New
141-142	Control
143	Control
1101-1130	Label; Field

Site Map

SWT Filename: CCSiteMap.swt

This template creates a table of contents for all screens and views in the application.

Includes Tree

```
CCSiteMap.swt

   CCStylesChoice.swt
```

Spell Checker Popup Applet

SWT Filename: CCAppletSpellCheck.swt

This is a specialized salutation applet used for creating the spell check pop-up list.

Includes Tree

```
CCAppletSpellCheck.swt
```

Table 79 lists the mappable items for this template.

Table 79. Mappable Items

ID	Description
91	Inside Applet Help Text
132-133	Control
134	Div
135-136	Control
137	Div
138-139	Control
140	Div
141-142	Control
143	Div
144-145	Control
146	Div
152-153	Control
154	Div

Table 79. Mappable Items

ID	Description
155	Control
156	Control
1201	Replacement Word Label
1211	Suggested Word Label
1221	Dictionary Label
1300	Text Segment

Specialized Views

View Dashboard

SWT Filename: CCViewDashboard.swt

This is a specialized view for the Dashboard applet. There is support for one applet, and it consumes the full window width.

Includes Tree

```
CCViewDashboard.swt

   CCHTMLHeader.swt

      CCStyleChoice.swt

   CCHTMLFooter.swt
```

View SME Segment Detail

SWT Filename: CCSegmentDetailView.swt

Figure 68. View SME Segment Detail

This is a specialized view used in tree and expression builder.

Includes Tree:

```
CCSegmentDetailView.swt
   CCHTMLHeader.swt
      CCStylesChoice.swt
   CCViewbar_Tabs.swt
   CCHTMLFooter.swt
```

Table 80. Mappable Items

ID	Description
1	Parent Applet
2	Tree Applet
3	Java Applet

Siebel Templates for Customer Applications 7

Most of the Customer Applications (that is, Siebel Interactive Selling, Siebel eMarketing, Siebel eService, and Siebel eTraining) will use a set of approximately 25 Customer Application templates along with a limited subset of eBusiness templates for messages and greetings.

There are two types of Siebel Partner Relationship Manager: Partner Portal and Partner Manager. Partner Manager can use High Interactivity, and all templates shown through it adapt to the employee application look and feel. Partner Portal uses Standard Interactivity and adapts to the customer application color scheme.

This links below let you access topics that provide examples of the applet and view templates used in the Customer Applications.

Overview of UI Elements

Table 81 gives an overview of user interface elements in customer applications.

Table 81. UI Elements in Customer Applications

Element	Description
Header	The header is composed of the following elements: the banner, the screenbar, and the viewbar. The framed area at the top of the page that remains visible as the content area is scrolled.
Banner	The top frame of the header that is used for site branding and navigation. The banner contains the company's logo on the left side, and the global navigation hyperlinks in the lower right corner.
Global Navigation Hyperlinks	Hyperlinks that appear in the lower right corner of the banner frame and provide functions outside the domain of the primary tabs. In customer and partner applications, the global navigation hyperlinks include *Shopping Cart, My Account, Help, Contact Us,* and *Log In/Out*.
Screenbar	The area that displays the primary tabs, which provide access to key areas of the applications.
Viewbar	The bottom frame of the header that is used to display the simple search applet in most Customer Applications. In eChannel, the viewbar is used to display the second-level navigation and favorites drop-down list controls.
Simple Search Applet	A small applet in the right side of the viewbar used to perform simple searches. The simple search applet also contains an icon that links to the Search Center page, where more advanced searches can be conducted.
Content Area	The largest frame of the page that contains a view template and one or more applet templates. The content area may be scrolled vertically without affecting the position of the header frames.

Table 81. UI Elements in Customer Applications

Element	Description
Salutation and Personalized Greeting	Area at the top of the content area for displaying text messages. The message area usually displays a personalized greeting on an application's home page and a threadbar on pages deeper within an application's hierarchical structure.
Threadbar	A set of hyperlinks in the message area that illustrates the path the user has taken through the hierarchical structure of the application and allows for easy navigation back to previously visited pages.

Applet Template Visual Reference

This section provides examples of applets using each of the Customer Application templates.

- "Go To View List" on page 528

List Brief/Bullet

dCCAppletListBriefBullet.swt

For an example of the List Brief/Bullet applet, see Figure 69.

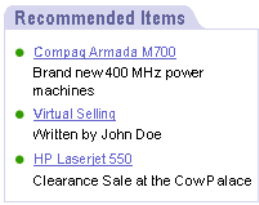


Figure 69. List Brief/Bullet Applet

List Brief/Bullet/Border

dCCAppletListBriefBulletBorder.swt

For an example of the List Brief/Bullet/Border applet, see Figure 70.

Figure 70. List Brief/Bullet/Border Applet

List Brief/Bullet/Shaded

dCCAppletListBriefBulletShade.swt

For an example of the List Brief/Bullet/Shaded applet, see Figure 71.

Figure 71. List Brief/Bullet/Shaded Applet

List Brief/Image Bullet

dCCAppletListBriefImgBullet.swt

For an example of the List Brief/Image Bullet applet, see Figure 72.

Figure 72. List Brief/Image Bullet Applet

List Brief/Image Bullet/Border

dCCAppletListBriefImgBulletBorder.swt

For an example of the List Brief/Image Bullet/Border applet, see Figure 73.

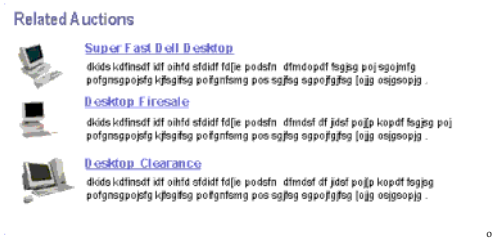


Figure 73. List Brief/Image Bullet/Border Applet

List Brief/Image Bullet/Shaded

dCCAppletListBriefImgBulletShade.swt

For an example of the List/Image Bullet applet, see Figure 74.

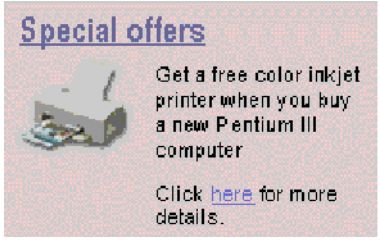


Figure 74. List Brief/Image Bullet/Shaded Applet

List Detailed/Image Bullet

dCCAppletListDetailedImgBullet.swt

For an example of the List Detailed/Image Bullet applet, see Figure 75.

Figure 75. List Detailed/Image Bullet Applet

List Detailed/Image Bullet/Record Navigation

dCCAppletListDetailedImgBulletRecNav.swt

For an example of the List Detailed/Image Bullet/Record Navigation applet, see Figure 76.

Figure 76. List Detailed/Image Bullet/Record Navigation Applet

List Detailed/Image Bullet/Record Navigation 2

dCCAppletListDetailedImgBulletRecNav2.swt

For an example of the List Detailed/Image Bullet/Record Navigation 2 applet, see Figure 77.

Figure 77. List Detailed/Image Bullet/Record Navigation 2 Applet

Form/Title Only

dCCAppletFormTitle.swt

For an example of the Form/Title Only applet, see Figure 78.

Figure 78. Form/Title Only Applet

List/Categorized/Bulleted/Tabbed

dCCAppletListCategorizedBulletTab.swt

For an example of the List/Categorized/Bulleted/Tabbed applet, see Figure 79.

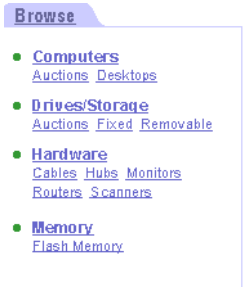


Figure 79. List/Categorized/Bulleted/Tabbed Applet

List/Categorized/Bulleted

dCCAppletListCategorizedBullet.swt

For an example of the List/Categorized/Bulleted applet, see Figure 80.

Figure 80. List/Categorized/Bulleted Applet

Form/Item Detail 1

dCCAppletFormItemDetail.swt

For an example of the Form/Item Detail 1 Applet template, see Figure 81.

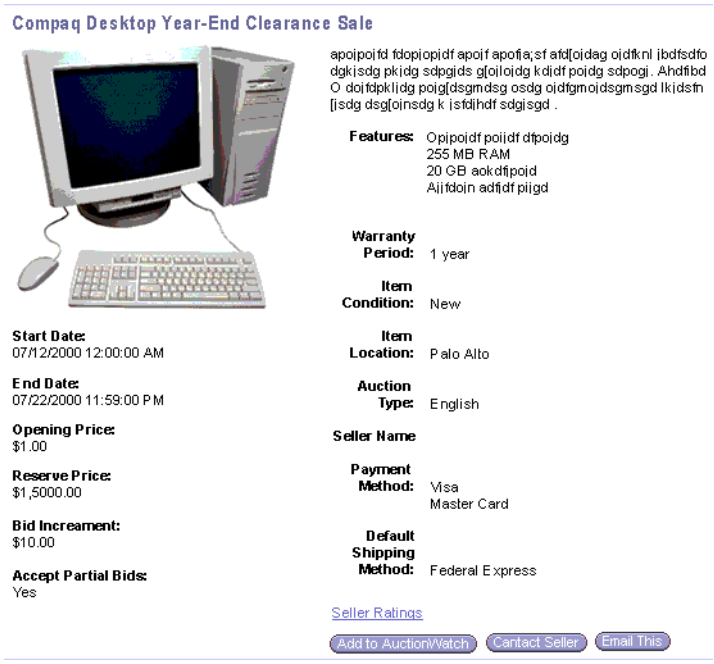


Figure 81. Form/Item Detail 1 Applet

Form/1-Column/Basic

dCCAppletFormBasic.swt

For an example of the Form/1-Column/Basic applet, see Figure 82.

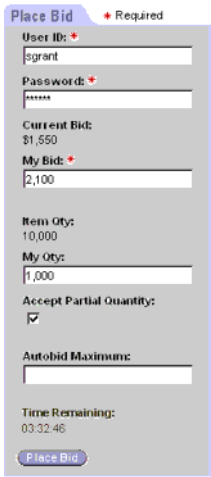


Figure 82. Form/1-Column/Basic Applet

List/Light

dCCAppletListLight.swt

For an example of the List/Light template, see Figure 83.

Figure 83. List/Light Applet

Form/Totals

dCCAppletFormTotals.swt

For an example of the Form/Totals template, see Figure 84.

Figure 84. Form/Totals Applet

List Tabbed

dCCAppletListTabbed.swt

For an example of the List Tabbed applet, see Figure 85.

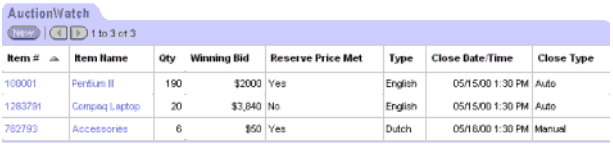


Figure 85. List Tabbed Applet

Form/4 Column

dCCAppletForm4Col.swt (formerly dCCAppletForm2Col.swt)

For an example of the Form/4 Column applet, see Figure 86.

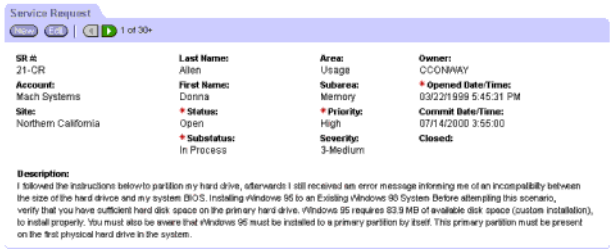


Figure 86. Form/4 Column Applet

Form/1-Column

dCCAppletForm1Col.swt

For an example of the Form/1-Column applet, see Figure 87.

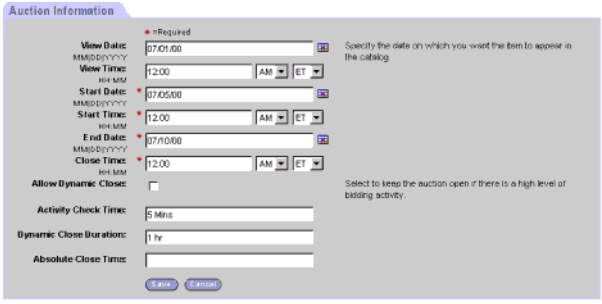


Figure 87. Form/1-Column Applet

List/Horizontal

dCCAppletListHorizontal.swt

For an example of the List/Horizontal applet, see Figure 88.

Figure 88. List/Horizontal Applet

Real-Time Shopping Cart

For an example of the Real-Time Shopping Cart applet, see Figure 89.

Figure 89. Real-Time Shopping Cart

Go To View List

For an example of the Go To View List applet, see Figure 90.

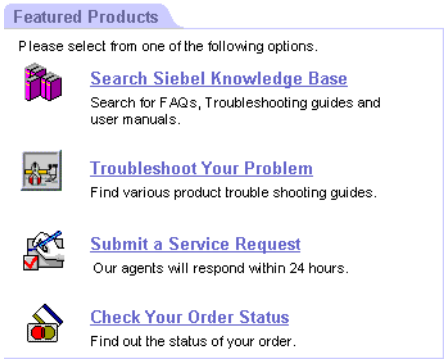


Figure 90. Go To View List

Applet Templates

DotCom Applet List Brief Bullet

SWT Filename: dCCAppletListBriefBullet.swt

This template creates a bulleted list of records with record title and a brief description. See Figure 91 for an example.

Figure 91. DotCom Applet List Brief Bullet

Includes Tree

```
dCCAppletListBriefBullet.swt
   CCApplet_NamedSpacer.swt
   dCCTitle_Portal.swt
      CCLayoutButtons.swt
   CCTogglebar_drop.swt
   dCCListTitleNoRule.swt
   dCCListBodyBullet.swt
```

Table 82 lists the mappable items for this template.

Table 82. Mappable Items

ID	Description
2	Title; DrillDown; New
90	Title
106	Find
107	Search
109-111	Control
131	New
134	Reset
135	Cancel
136	Save
157	Label
184	DrillDown Title
203	MinimizeApplet
204	MaximizeApplet
207	MoveAppletUp

Table 82. Mappable Items

ID	Description
208	MoveAppletDown
211	ShowApplet
212	HideApplet
501	Field
502-511	Field
555	Label
1100	Outside Applet Help Text

DotCom Applet List Brief Bullet/Border

SWT Filename: dCCAppletListBriefBulletBorder.swt

This template creates a bulleted list of records with record title and a brief description. See Figure 92 for an example.

Figure 92. DotCom Applet List Brief Bullet/Border

Includes Tree

```
dCCAppletListBriefBulletBorder.swt

   CCApplet_Spacer.swt
```

dCCListTitleNoRule.swt

dCCListBodyBullet.swt

Table 83 lists the mappable items for this template.

Table 83. Mappable Items

ID	Description
90	Title; DrillDown; New
106	Find
107	Search
109-111	Control
131	New
134	Reset
135	Cancel
136	Save
184	DrillDown Title
501	Field
502-511	Field
1100	Outside Applet Help Text

DotCom Applet List Brief Bullet / Shade

SWT Filename: dCCAppletListBriefBulletShaded.swt

This template creates a bulleted list of records with record title and a brief description. See Figure 93 for an example.

Figure 93. DotCom Applet List Brief Bullet/Shade

Includes Tree

```
dCCAppletListBriefBulletShaded.swt
   CCApplet_Spacer.swt
   CCTogglebar_drop.swt
   dCCListTitleNoRule.swt
   dCCListBodyBullet.swt
```

Table 84 lists the mappable items for this template.

Table 84. Mappable Items

ID	Description
2	Title; DrillDown; New
90	Title
106	Find
107	Search
109-111	Control
131	New

Table 84. Mappable Items

ID	Description
134	Reset
135	Cancel
136	Save
184	DrillDown Title
501	Field
502-511	Field
1100	Outside Applet Help Text

DotCom Applet List Brief ImgBullet

SWT Filename: dCCAppletListBriefImgBullet.swt

This template creates a bulleted list of records with record title, image, and a brief description. See Figure 94 for an example.

Figure 94. DotCom Applet List Brief ImgBullet

Includes Tree

```
dCCAppletListBriefImgBullet.swt

   CCApplet_NamedSpacer.swt

   dCCTitle_Portal.swt

      CCLayoutButtons.swt
```

```
CCTogglebar_drop.swt
dCCListTitleNoRule.swt
dCCListBodyImgBullet.swt
```

Table 85 lists the mappable items for this template.

Table 85. Mappable Items

ID	Description
1	Anchor; Title
2	Title; DrillDown; New
90	Title
106	Find
107	Search
109-111	Control
131	New
134	Reset
135	Cancel
136	Save
157	Label
184	DrillDown Title
203	MinimizeApplet
204	MaximizeApplet
207	MoveAppletUp
208	MoveAppletDown
211	ShowApplet
212	HideApplet
501	Small Image (30x30)

Table 85. Mappable Items

ID	Description
502	Title
503	Description
504-513	Other Text
520-529	Other
555	Label
1100	Outside Applet Help Text

DotCom Applet List Brief ImgBullet / Border

SWT Filename: dCCAppletListBriefImgBulletBorder.swt

This template creates a bulleted list of records with record title, image, and a brief description. See Figure 95 for an example.

Figure 95. DotCom Applet List Brief ImgBullet/Border

Includes Tree

```
dCCAppletListBriefImgBulletBorder.swt

   dCCListTitleNoRule.swt

   dCCListBodyImgBullet.swt
```

Table 86 lists the mappable items for this template.

Table 86. Mappable Items

ID	Description
1	Anchor; Title
90	Title; DrillDown; New
106	Find
107	Search
109-111	Control
131	New
134	Reset
135	Cancel
136	Save
184	DrillDown Title
501	Small Image (30x30)
502	Title
503	Description
504-513	Other Text
520-529	Other
1100	Outside Applet Help Text

DotCom Applet List Brief ImgBullet / Shade

SWT Filename: dCCAppletListBriefImgBulletShaded.swt

This template creates a bulleted list of records with record title, image, and a brief description. See Figure 96 for an example.

Figure 96. DotCom Applet List Brief ImgBullet/Shade

Includes Tree

```
dCCAppletListBriefImgBulletShaded.swt

   dCCListTitleNoRule.swt

   dCCListBodyImgBullet.swt
```

Table 87 lists the mappable items for this template.

Table 87. Mappable Items

ID	Description
1	Anchor; Title
90	Title; DrillDown; New
106	Find
107	Search
109-111	Control
131	New
134	Reset
135	Cancel
136	Save

Table 87. Mappable Items

ID	Description
184	DrillDown Title
501	Small Image (30x30)
502	Title
503	Description
504-513	Other Text
520-529	Other
1100	Outside Applet Help Text

DotCom Applet List Brief ImgBullet 2

SWT Filename: dCCAppletListBriefImgBullet2.swt

This template creates a bulleted list of records with record title, image, and a brief description. See Figure 97 for an example.

Figure 97. DotCom Applet List Brief ImgBullet 2

Includes Tree

```
dCCAppletListBriefImgBullet2.swt

   CCApplet_NamedSpacer.swt

   dCCTitle_Mapped.swt
```

```
   CCLayoutButtons.swt
CCTogglebar_drop.swt
dCCListBodyImgBullet2.swt
```

Table 88 lists the mappable items for this template.

Table 88. Mappable Items

ID	Description
1	Anchor; Title
2	Small Image (30x30)
90	Title; DrillDown Title; Label
157	Label
184	DrillDown Title
203	MinimizeApplet
204	MaximizeApplet
207	MoveAppletUp
208	MoveAppletDown
211	ShowApplet
212	HideApplet
501	Small Image (30x30)
502	Title
503	Description
504-513	Other Text
520-529	Other
555	Label
1100	Outside Applet Help Text

DotCom Applet List Categorized (No Tab)

SWT Filename: dCCAppletListCategorizedNoTab.swt

This template creates top-level items in an hierarchical list. See Figure 98 for an example.

Figure 98. DotCom Applet List Categorized (No Tab)

Includes Tree

`dCCAppletListCategorizedNoTab.swt`

Table 89 shows the mappable items for this template.

Table 89. Mappable Items

ID	Description
90	Title
501	Image
502	Field

DotCom Applet List Categorized Bullet

SWT Filename: dCCAppletListCategorizedBullet.swt

This template creates top-level bulleted items in an hierarchical list. See Figure 99 for an example.

Figure 99. DotCom Applet List Categorized Bullet

Includes Tree

```
dCCAppletListCategorizedBullet.swt

   dCCListCategorized.swt
```

Table 90 shows the mappable items for this template.

Table 90. Mappable Items

ID	Description
132	Field
502	Field

DotCom Applet List Categorized Bullet / Tabbed

SWT Filename: dCCAppletListCategorizedBulletTab.swt

This template creates top-level bulleted items in an hierarchical list. Items are surrounded by the standard applet treatment. See Figure 100 for an example.

Figure 100. DotCom Applet List Categorized Bullet/Tabbed

Includes Tree

```
dCCAppletListCategorizedBulletTab.swt

   CCApplet_Spacer.swt

   CCTitle.swt

   dCCButtons_List.swt

   dCCListCategorized.swt
```

Table 91 shows the mappable items for this template.

Table 91. Mappable Items

ID	Description
2	Back
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control

Table 91. Mappable Items

ID	Description
131	New
132	Field
134	Reset
135	Cancel
136	Save
139-141	Control
502	Field
1500	Required Legend

DotCom Applet List Categorized Tabbed

SWT Filename: dCCAppletListCategorizedTab.swt

This template creates top-level bulleted items in an hierarchal list. Items are surrounded by the standard applet treatment. See Figure 101 for an example.

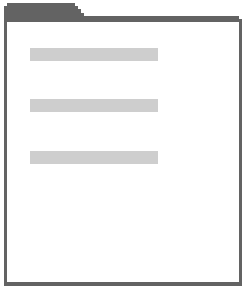


Figure 101. DotCom Applet List Categorized Tabbed

Includes Tree

```
dCCAppletListCategorizedTab.swt
```

```
CCApplet_Spacer.swt
CCTitle.swt
dCCButtons_List.swt
```

Table 92 lists the mappable items for this template.

Table 92. Mappable Items

ID	Description
2	Back
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
131	New
134	Reset
135	Cancel
136	Save
139-141	Control
501	Image
502	Field
1500	Required Legend

DotCom Applet List Categorized TOC

SWT Filename: dCCAppletListCategorizedTOC.swt

This template creates a table of contents from top-level categories. Map the TOC title to ID 90. Map an image or bullet to ID 501. Map the item name to ID 502. See Figure 102 for an example.

Figure 102. DotCom Applet List Categorized TOC

Includes Tree

```
dCCAppletListCategorizedTOC.swt

   CCApplet_Spacer.swt
```

Table 93 lists the mappable items for this template.

Table 93. Mappable Items

ID	Description
1	Anchor
90	TOC Title
501	Image
502	Field

DotCom Applet List Detailed ImgBullet

SWT Filename: dCCAppletListDetailedImgBullet.swt

This is a standard template for displaying detailed description of products in list format. It supports product image, title description, and buttons on a per record basis. See Figure 103 for an example.

Figure 103. DotCom Applet List Detailed ImgBullet

Includes Tree

```
dCCAppletListDetailedImgBullet.swt
   CCApplet_Spacer.swt
   dCCTitle_Mapped.swt
      CCLayoutButtons.swt
   CCTogglebar_drop.swt
   dCCListBodyImgBulletDetailed.swt
```

Table 94 lists the mappable items for this template.

Table 94. Mappable Items

Id	Description
2	Small Image (30x30)
90	Title: DrillDown title; Label
142-143	Control
145-146	Control
157	Control

Table 94. Mappable Items

Id	Description
158	Control
184	DrillDown Title
203	MinimizeApplet
204	MaximizeApplet
207	MoveAppletUp
208	MoveAppletDown
211	ShowApplet
212	HideApplet
501	Small Image (30x30)
502-503	Item Name
504-505	Label; Field
510-512	Label; Field
555	Label

DotCom Applet List Detailed ImgBullet RecNav

SWT Filename: dCCAppletListDetailedImgBulletRecNav.swt

This is the standard applet for displaying detailed descriptions of products in list format. It supports product image, title description, and buttons on a per record basis. See Figure 104 for an example.

Figure 104. DotCom Applet List Detailed ImgBullet RecNav

Includes Tree

```
dCCAppletListDetailedImgBulletRecNav.swt
   CCApplet_Spacer.swt
   dCCTitle_RecNav.swt
      CCRecordNav.swt
   CCToggebar_drop.swt
   dCCListBodyImgBulletDetailed.swt
   CCRecordNav.swt
```

Table 95 lists the mappable items for this template.

Table 95. Mappable Items

ID	Description
2	Small Image (30x30)
121	First
122	Previous
123	Next

Table 95. Mappable Items

ID	Description
124	Last
142-143	Control
145-146	Control
157-158	Control
501	Small Image (30x30)
502-503	Item Name
504-505	Label; Field
510-512	Label; Field
1100	Outside Applet Help Text

DotCom Applet List Detailed ImgBullet RecNav2

SWT Filename: dCCAppletListDetailedImgBulletRecNav2.swt

This is the standard applet for displaying detailed descriptions of products in list format. It supports product image, title description, and buttons on a per record basis. See Figure 105 for an example.

Figure 105. DotCom Applet List Detailed ImgBullet RecNav2

Includes Tree

`dCCAppletListDetailedImgBulletRecNav2.swt`

```
CCApplet_Spacer.swt
dCCTitle_RecNav.swt
   CCRecordNav.swt
CCToggebar_drop.swt
dCCListBodyImgBulletDetailed2.swt
CCRecordNav.swt
```

Table 96 lists the mappable items for this template.

Table 96. Mappable Items

ID	Description
2	Small Image (30x30)
121	First
122	Previous
123	Next
124	Last
142-143	Control
145	Control
501	Small Image (30x30)
502-503	Item Name
504-505	Label; Field
506-508	Control
510-511	Label; Field
1100	Outside Applet Help Text

DotCom Applet List Horizontal

SWT Filename: dCCAppletListHorizontal.swt

This template creates a list where records are shown across the screen rather than down. It is useful for creating comparison applets. Image, title, and a brief description are supported. Record navigation is supported. See Figure 106 for an example of this template.

Figure 106. DotCom Applet List Horizontal

Includes Tree

```
dCCAppletListHorizontal.swt
   CCApplet_NamedSpacer.swt
   CCTitle_Named.swt
      CCTitle.swt
   CCTogglebar_drop.swt
   dCCListButtonsTop.swt
      dCCButtons_List.swt
      CCRecordNav.swt
      CCTogglebar_drop.swt
      CCListButtonsTopRight.swt
   dCCListBodyHorizontal.swt
   CCBottomApplet_NamedList.swt
```

Table 97 lists the mappable items for this template.

Table 97. Mappable Items

ID	Description
2	Control
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
121	First
122	Previous
123	Next
124	Last
131	New
134	Reset
135	Cancel
136	Save
139-141	Control
142-143	Control
145	Control
150-151	Control
160-164	Control
161-164	Control
501	Small Image (30x30)
502	Item Name
503-505	Field

Table 97. Mappable Items

ID	Description
510-512	Label; Field
1100	Outside Applet Help Text
1500	Required Legend

DotCom Applet List Light

SWT Filename: dCCAppletListLight.swt

This specialized list template presents an additional totals row at the bottom of the list. The totals row is demarked by a double line above the row. Columns that produce totals must be marked as such in Tools. A totals row label can be added by mapping a label to ID 199. One label applies to all row totals, so the label used should be generic. See Figure 107 for an example of this template.

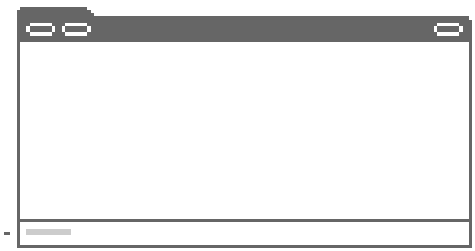


Figure 107. DotCom Applet List Light

Includes Tree

```
dCCAppletListLight.swt
   CCApplet_NamedSpacer.swt
   CCTitle_Named.swt
      CCTitle.swt
   CCTogglebar_drop.swt
```

```
dCCListButtonsTop.swt
   dCCButtons_List.swt
   CCRecordNav.swt
   CCTogglebar_drop.swt
   CCListButtonsTopRight.swt
dCCListHeaderTotals.swt
dCCListBodyTotalsNoRowHilite.swt
CCBottomApplet_NamedList.swt
```

Table 98 lists the mappable items for this template.

Table 98. Mappable Items

ID	Description
2	Control
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
121	First
122	Previous
123	Next
124	Last
131	New
132-133	Control
134	Reset
135	Cancel
136	Save

Table 98. Mappable Items

ID	Description
139-141	Control
142-143	Control
144	Selected Row
145	Control
146	Save
147	Pick Control
150-151	Control
160-164	Control
161-164	Control
199	Totals Label
501-520	...
1100	Outside Applet Help Text
1500	Required Legend

DotCom Applet List Search Results

SWT Filename: dCCAppletListSearchResults.swt

This applet defines the search results list. The list does not support record selection, so the selection highlight is eliminated. See Figure 108 for an example of this template.

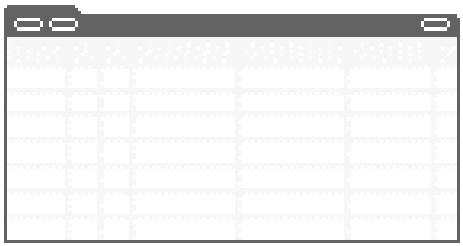


Figure 108. DotCom Applet List Search Results

Includes Tree

```
dCCAppletListSearchResults.swt
   CCApplet_NamedSpacer.swt
   CCTitle_Named.swt
      CCTitle.swt
   dCCListButtonsTop.swt
      dCCButtons_List.swt
      CCRecordNav.swt
      CCTogglebar_drop.swt
      CCListButtonsTopRight.swt
   dCCListHeader.swt
   dCCListBodySearchResults.swt
   CCBottomApplet_NamedList.swt
```

Table 99 lists the mappable items for this template.

Table 99. Mappable Items

ID	Description
2	Control
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
121	First
122	Previous
123	Next
124	Last
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-141	Control
142-143	Control
144	Select
145	Control
146	Control
147	...
150-151	Control

Table 99. Mappable Items

ID	Description
160-164	Control
161-164	Control
501-520	...
1100	Outside Applet Help Text
1500	Required Legend

DotCom Applet List Subcategory

SWT Filename: dCCAppletListSubCategory.swt

This template presents comma-delimited subcategory links that fill the space available to them. See Figure 109 for an example of this template.

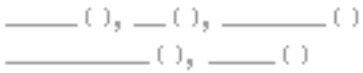


Figure 109. DotCom Applet List Subcategory

Includes Tree

`dCCAppletListSubCategory.swt`

Table 100 lists the mappable items for this template.

Table 100. Mappable Items

ID	Description
502	Field
503	Field

DotCom Applet List Subcategory 1 Per Row

SWT Filename: dCCAppletListSubCategory_1PerRow.swt

Figure 110 shows an example of this template.

Figure 110. DotCom Applet List Subcategory 1 Per Row

Includes Tree

`dCCAppletListSubCategory_1PerRow.swt`

Table 101 lists the mappable items for this template.

Table 101. Mappable Items

ID	Description
501	Image; Subcategory; Count
502	Subcategory
503	Count

DotCom Applet List Subcategory 4-Per-Column

SWT Filename: dCCAppletListSubCategory_4PerColumn.swt

This template presents subcategory links, shown as four links per column. See Figure 111 for an example.

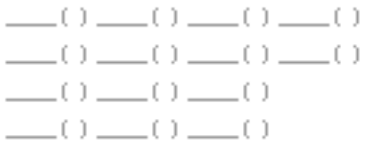


Figure 111. DotCom Applet List Subcategory 4-Per-Column

Includes Tree

`dCCAppletListSubCategory_4PerColumn.swt`

Table 102 lists the mappable items for this template.

Table 102. Mappable Items

ID	Description
501	Image: Subcategory; Count
502	Subcategory
503	Count

DotCom Applet List Subcategory 6-Per-Column

SWT Filename: dCCAppletListSubCategory_6PerColumn.swt

This template presents subcategory links, shown as six links per column. See Figure 112 for an example.

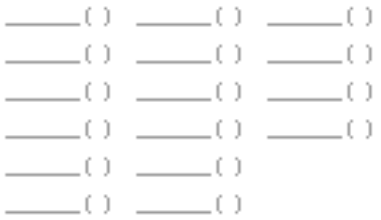


Figure 112. DotCom Applet List Subcategory 6-Per-Column

Includes Tree

`dCCAppletListSubCategory_6PerColumn.swt`

Table 103 lists the mappable items for this template.

Table 103. Mappable Items

ID	Description
501	Image: Subcategory; Count
502	Subcategory
503	Count

DotCom Applet List Subcategory Indented

SWT Filename: dCCAppletListSubCategoryIndented.swt

This template presents comma-delimited subcategory links that fill the space available to them. The links are indented to emphasize the hierarchical nature of the data. See Figure 113 for an example.

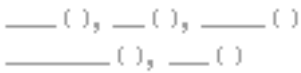


Figure 113. DotCom Applet List Subcategory Indented

Includes Tree

`dCCAppletListSubCategoryIndented.swt`

`CCApplet_NamedSpacer.swt`

`dCCListTitleNoRule.swt`

Table 104 lists the mappable items for this template.

Table 104. Mappable Items

ID	Description
90	Title;DrillDown Title; New
106	Find
107	Search
109-111	Control
131	New
134	Reset
135	Cancel
136	Save
184	DrillDown Title
502	Field

Table 104. Mappable Items

ID	Description
503	Field
1100	Outside Applet Help Text

DotCom Applet List Tabbed

SWT Filename: dCCAppletListTabbed.swt

This is the standard applet template for lists. A list can typically display between seven and ten visible columns. It is possible to map more visible columns, but this is not recommended since they may create undesirable text-wrapping or in extreme cases force horizontal scrolling. The template supports mapping up to twenty fields. This is done so that you may mark the majority of fields as available but hidden. Fields marked as such will not appear by default in the list, but will appear in the columns displayed dialog. See Figure 114 for an example of this template.

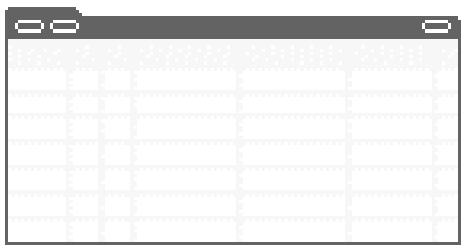


Figure 114. DotCom Applet List Tabbed

Includes Tree

```
dCCAppletListTabbed.swt

   CCApplet_NamedSpacer.swt

   CCTitle_Named.swt

      CCTitle.swt
```

```
dCCListButtonsTop.swt
   dCCButtons_List.swt
   CCRecordNav.swt
   CCTogglebar_drop.swt
   CCListButtonsTopRight.swt
dCCListHeader.swt
dCCListBodyNoRowHilite.swt
CCBottomApplet_NamedList.swt
```

Table 105 lists the mappable items for this template.

Table 105. Mappable Items

ID	Description
2	Control
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
121	First
122	Previous
123	Next
124	Last
131	New
132-133	Control
134	Reset
135	Cancel
136	Save

Table 105. Mappable Items

ID	Description
139-141	Control
142-143	Control
144	Selected Row
145	Control
146	Save
147	Pick Control
150-151	Control
160-164	Control
161-164	Control
501-520	Field
1100	Outside Applet Help Text
1500	Required Legend

DotCom List Merged (Base/EditList)

SWT Filename: dCCAppletListMerged_B_EL.swt

This is a specialized applet template. It can be used on report and summary type pages where it is desirable to stack applets one on top of the other without white space between them. See Figure 115 for an example of this template.

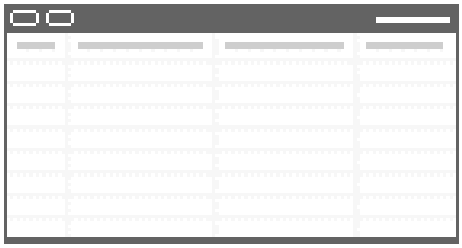


Figure 115. DotCom List Merged (Base/Edit)

NOTE: In this template the applet title appears in the upper right corner.

Includes Tree

```
dCCAppletListMerged_B_EL.swt

   dCCListButtonsTopWithTitle.swt

      dCCButtons_List.swt

      CCRecordNav.swt

      CCTogglebar_drop.swt

   dCCListHeader.swt

   dCCListBodyNoRowHilite.swt
```

Table 106 lists the mappable items for this template.

Table 106. Mappable Items

ID	Description
2	...
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
121	First
122	Previous
123	Next
124	Last
131	New
132-133	Control
134	Reset
135	Cancel
136	Save
139-141	Control
142-143	Control
144	Selected Row
145	Control
146	Save
147	Pick Control
501-520	Field

DotCom Applet Form 1-Column

SWT Filename: dCCAppletForm1Col.swt

This is a standard one-column template. Labels appear to the left of the fields. Buttons appear at the bottom of the form. See Figure 116 for an example of this template.

Figure 116. DotCom Applet Form 1-Column

Includes Tree

```
dCCAppletForm1Col.swt
   CCApplet_NamedSpacer.swt
   CCTitleNamed.swt
      CCTitle.swt
   CCTogglebar_drop.swt
   dCCForm1Col.swt
      dCCButttons_Form.swt
   CCBottomApplet_NamedForm.swt
```

Table 107 lists the mappable items for this template.

Table 107. Mappable Items

ID	Description
2	Back
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-110	Control
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
157-158	Control
1000	FormSection
1001	FormSection
1002	FormSection
1003	FormSection
1004	FormSection
1005	FormSection
1006	FormSection
1300-1305	Required; Label; Field
1306-1311	Required; Label; Field

Table 107. Mappable Items

ID	Description
1312-1317	Required; Label; Field
1318-1323	Required; Label; Field
1324-1329	Required; Label; Field
1330-1335	Required; Label; Field
1336-1341	Required; Label; Field
1500	Required Legend

DotCom Applet Form 2-Column

SWT Filename: dCCAppletForm2Col.swt

This is a standard two-column template. Labels appear to the left of the fields. There are two columns of label/field pairs followed by one wide column that spans both columns. See Figure 117 for an example.

Figure 117. DotCom Applet Form 2-Column

Includes Tree

```
dCCAppletForm2Col.swt
   CCApplet_NamedSpacer.swt
   CCTitleNamed.swt
      CCTitle.swt
```

```
CCTogglebar_drop.swt
dCCFormButtonsTop.swt
   dCCButttons_Form.swt
   CCRecordNav.swt
   CCTogglebar_drop.swt
   CCFormButttonsTopRight.swt
dCCForm2Col.swt
CCBottomApplet_NamedForm.swt
```

Table 108 lists the mappable items for this template.

Table 108. Mappable Items

ID	Description
2	Back
91-92	Inside Applet Help Text
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-110	Control
121	First
122	Previous
123	Next
124	Last
131	New
132	Edit
133	Delete
134	Reset

Table 108. Mappable Items

ID	Description
135	Cancel
136	Save
139-143	Control
150-152	Control
157-158	Control
160-164	Control
1001	FormSection
1002	FormSection
1003	FormSection
1004	FormSection
1030	FormSection
1035	FormSection
1100-1104	Required; Label; Field
1105-1109	Required; Label; Field
1110-1114	Required; Label; Field
1115-1119	Required; Label; Field
1130-1134	Label; Field
1135-1139	Label; Field
1500	Required Legend
2001	FormSection
2002	FormSection
2003	FormSection
2004	FormSection
2100-2104	Required; Label; Field

Table 108. Mappable Items

ID	Description
2105-2109	Required; Label; Field
2110-2114	Required; Label; Field
2115-2119	Required; Label; Field

DotCom Applet Form 4-Column

SWT Filename: dCCAppletForm4Col.swt

This is the standard four-column form template. Fields can be mapped for up to four columns. Labels appear above field values. Validation errors appear at the top of the form Instructional text can be added to ID 91; it spans all four columns. See Figure 118 for an example.

Figure 118. DotCom Applet Form 4-Column

The form defines a large number of control placeholders, some spanning one column, some spanning two columns, and some spanning all four columns.

Includes Tree

```
dCCAppletForm4Col.swt

   CCApplet_NamedSpacer.swt

   CCTitleNamed.swt
```

```
      CCTitle.swt
   CCTogglebar_drop.swt
   dCCFormButtonsTop.swt
      dCCButttons_Form.swt
      CCRecordNav.swt
      CCTogglebar_drop.swt
      CCFormButttonsTopRight.swt
   CCForm4Col.swt
   CCBottomApplet_NamedForm.swt
```

Table 109 lists the mappable items for this template.

Table 109. Mappable Items

ID	Description
2	Control
91	Inside Applet Help Text
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-110	Control
121	First
122	Previous
123	Next
124	Last
131	New
132	Edit
133	Delete

Table 109. Mappable Items

ID	Description
134	Reset
135	Cancel
136	Save
139-143	Control
150-152	Control
157-158	Control
160-164	Control
1001-1009	FormSection
1020	FormSection
1296-1300	Required; Label; 2-Column Wide Field
1301-1310	Required; Label; Field
1311-1315	Required; Label; 2-Column Wide Field
1316-1330	Required; Label; Field
1331-1335	Required; Label; 2-Column Wide Field
1336-1340	Required; 4-Column Wide Label; 4-Column Wide Field
1360-1374	Required; Label; Field
1500	Required Legend
1801-1810	Required; Label; Field
1816-1830	Required; Label; Field
1860-1874	Required; Label; Field
2001-2009	FormSection
2296-2300	Required; Label; 2-Column Wide Field
2301-2310	Required; Label; Field

Table 109. Mappable Items

ID	Description
2316-2330	Required; Label; Field
2331-2335	Required; Label; 2-Column Wide Field
2360-2374	Required; Label; Field
2801-2810	Required; Label; Field
2811-2815	Required; Label; 2-Column Wide Field
2816-2830	Required; Label; Field
2860-2874	Required; Label; Field

DotCom Applet Form Item Detail

SWT Filename: dCCAppletFormItemDetail.swt

This is a standard product detail form. It supports product title mapped to ID 1301 and product image mapped to ID 1300. See Figure 119 for an example of this template.

Figure 119. DotCom Applet Form Item Detail

Includes Tree

```
dCCAppletFormItemDetail.swt

   CCApplet_Spacer.swt

      dCCFormItemDetail.swt
```

Table 110 lists the mappable items for this template.

Table 110. Mappable Items

ID	Description
132-133	Control
141	Control
142	Control
157-158	Control
159	Control
160	Control
1102-1107	Label; Field
1112-1133	Label; Field
1300	Large Image (120X120)
1301	Item Name

DotCom Applet Form Search Top

SWT Filename: dCCAppletSearchTop.swt

This creates the scoping drop-down list for any dot-com search.

Includes Tree

```
dCCAppletSearchTop.swt

   CCTitle.swt

   dCCFormSearch.swt
```

Table 111 lists the mappable items for this template.

Table 111. Mappable Items

ID	Description
91	Inside Applet Help Text
1101-1130	Label; Field
1500	Required Legend

DotCom Applet Form Title

SWT Filename: dCCAppletFormTitle.swt

This is a simple applet used to present a free-form title. Title is derived from the applet's title property. See Figure 120 for an example.

Figure 120. DotCom Applet Form Title

Includes Tree

`dCCAppletFormTitle.swt`

DotCom Applet Links

SWT Filename: dCCAppletLinks.swt

This template creates a list of links with image and description. It is useful for creating small table of contents-type applets. See Figure 121 for an example.

Figure 121. DotCom Applet Links

Includes Tree

```
dCCAppletLinks.swt

   CCApplet_Spacer.swt
```

Table 112 lists the mappable items for this template.

Table 112. Mappable Items

ID	Description
1097	Title
1098	Intro Text
1099	Thematic Image
1100	Text
1101-1130	Image; Link (Required); Description; Text

Dotcom Form 4-Col Merged (Base/Edit/New)

SWT Filename: dCCAppletForm4ColMerged_B_E_N.swt

This is a specialized four-column form template. Fields can be mapped for up to four columns. Labels appear above field values. Validation errors appear at the top of the form. Instructional text can be added to ID 91; it spans all four columns. See Figure 122 for an example.

Figure 122. DotCom Form 4-Col Merged (Base/Edit/New)

The form defines a large number of control placeholders, some spanning one column, some spanning two columns, and some spanning all four columns.

The standard applet styles are supported.

The applet is specialized in that it does not support an applet tab. The applet title is shown in the button bar. This allows applets of this kind to be stacked together without white space between them. Useful for creating summary views.

Includes Tree

```
dCCAppletForm4ColMerged_B_E_N.swt

   dCCFormButtonsTopWithTitle.swt

      dCCButtons_Form.swt

      CCRecordNav.swt

      CCTogglebar_drop.swt

   dCCForm4ColBody.swt
```

Table 113 lists the mappable items for this template.

Table 113. Mappable Items

ID	Description
2	Inside Applet Help Text
91	Inside Applet Help Text
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-110	Control
121	First
122	Previous
123	Next
124	Last
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
157-158	Control
1001-1009	FormSection
1020	FormSection
1296-1300	Required; Label; 2-Column Wide Field
1301-1310	Required; Label; Field

Table 113. Mappable Items

ID	Description
1311-1315	Required; Label; 2-Column Wide Field
1316-1330	Required; Label; Field
1331-1335	Required; Label; 2-Column Wide Field
1336-1340	Required; 4-Column Wide Label; 4-Column Wide Field
1360-1374	Required; Label; Field
1801-1810	Required; Label; Field
1816-1830	Required; Label; Field
1860-1874	Required; Label; Field
2001-2009	FormSection
2296-2300	Required; Label; 2-Column Wide Field
2301-2310	Required; Label; Field
2316-2330	Required; Label; Field
2331-2335	Required; Label; 2-Column Wide Field
2360-2374	Required; Label; Field
2801-2810	Required; Label; Field
2811-2815	Required; Label; 2-Column Wide Field
2816-2830	Required; Label; Field
2860-2874	Required; Label; Field

View Templates

In the view diagrams below, the gray areas represent applet regions where one or more applets can be placed. Applets rendered on the Web expand horizontally to fit the column to which they have been assigned. The amount of displayed data determines how much vertical space an applet consumes.

DotCom View 100 66 33 100

SWT Filename: dCCView_100_66_33_100.swt

This is a standard view template. Applets in the first column consume 66 percent of the window width. Applets in the second column consume 33 percent of the window width. Applets placed in top or bottom regions consume the full window width. See Figure 123 for an example.

Figure 123. DotCom View 100 66 33 100

Includes Tree

```
dCCView_100_66_33_100.swt
   dCCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   dCCHTMLFooter.swt
```

Table 114 lists the mappable items for this template.

Table 114. Mappable Items

ID	Description
101	Salutation Applet
102-106	Applet
201	Mini-Applet
202-206	Applet
302-306	Applet
402-406	Applet
502-506	Applet
602-606	Applet

DotCom View 25 50 25

SWT Filename: dCCView_25_50_25.swt

This is the standard view template. Applets in the first column consume 25 percent of the window width. Applets in the second column consume 50 percent of the window width. Applets placed in the third column consume 25 percent of the window width. Applets placed in the top region consume the full window width. See Figure 124 for an example.

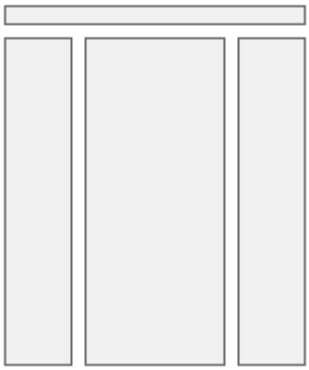


Figure 124. DotCom View 25 50 25

Includes Tree

```
dCCView_25_50_25.swt
   dCCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   dCCHTMLFooter
```

Table 115 lists the mappable items for this template.

Table 115. Mappable Items

ID	Description
101	Salutation Applet
102-111	Applet
202-211	Applet
302-311	Applet

DotCom View 25 50 25 Home

SWT Filename: dCCView_25_50_25_home.swt

This is a standard view template. Applets in the first column consume 25 percent of the window width. Applets in the second column consume 50 percent of the window width. Applets in the third column consume 25 percent of the window width. Applets placed in the top region consume the full window width. See Figure 125 for an example.

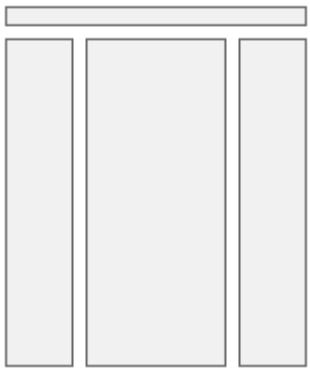


Figure 125. DotCom View 25 50 25 Home

Includes Tree

```
dCCView_25_50_25home.swt
   dCCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   dCCHTMLFooter.swt
```

Table 116 lists the mappable items for this template.

Table 116. Mappable Items

ID	Description
102-111	Applet
202-211	Applet
302-311	Applet

DotCom View 50 50

SWT Filename: dCCView_50_50.swt

This is a standard view template. Applets in the first column consume 50 percent of the window width. Applets in the second column consume 50 percent of the window width. See Figure 126 for an example.

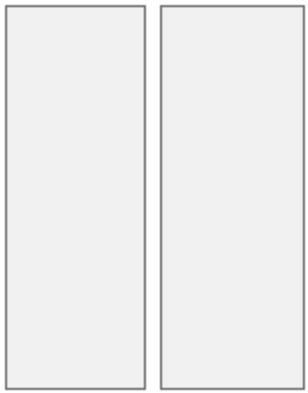


Figure 126. DotCom View 50 50

Includes Tree

```
dCCView_50_50.swt
   dCCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   dCCHTMLFooter
```

Table 117 lists the mappable items for this template.

Table 117. Mappable Items

ID	Description
101	Salutation Applet
102-111	Applet

Table 117. Mappable Items

ID	Description
202-211	Applet
302-311	Applet

DotCom View 66 33

SWT Filename: dCCView_66_33.swt

This is a standard view template. Applets in the first column consume 66 percent of the horizontal window width. Applets in the second column consume 33 percent of the window width. See Figure 127 for an example.

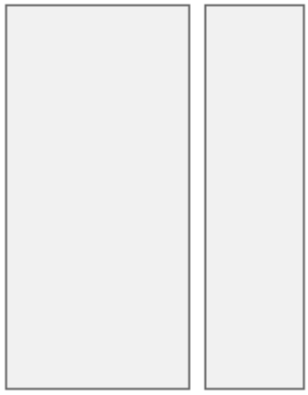


Figure 127. DotCom View 66 33

Includes Tree

```
dCCView_66_33.swt

   dCCHTMLHeader.swt

      CCStylesChoice.swt

   CCThreadbar.swt
```

```
dCCHTMLFooter
```

Table 118 lists the mappable items for this template.

Table 118. Mappable Items

ID	Description
101	Salutation Applet; Layout Controls
102-111	Applet
202-211	Applet
302	Applet

DotCom View Admin

SWT Filename: dCCViewAdmin1.swt

This template displays subviews as tabs across the top of the view. See Figure 128 for an example. It is useful for admin views that need to display nonrelated views that are not easily categorized.

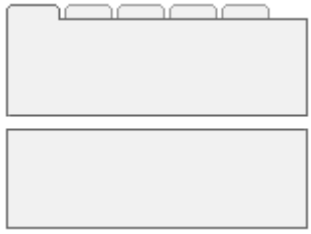


Figure 128. DotCom View Admin

Includes Tree

```
dCCViewAdmin1.swt
   dCCHTMLHeader.swt
      CCStylesChoice.swt
```

```
CCThreadbar.swt
dCCSubViewbar_Tabs.swt
   CCApplet_Spacer.swt
dCCHTMLFooter
```

Table 119 lists the mappable items for this template.

Table 119. Mappable Items

ID	Description
5	Child Applet with Pointer
6	Child Applet
7-9	Grandchild Applet
10-12	Grandchild Applet
13-15	Grandchild Applet
201	Mini-Applet

DotCom View Basic

SWT Filename: dCCView_Basic.swt

This is a standard view template. All applets consume the full window width and appear stacked on top of each other. See Figure 129 for an example.

Figure 129. DotCom View Basic

Includes Tree

```
dCCView_Basic.swt
   dCCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   dCCHTMLFooter
```

Table 120 lists the mappable items for this template.

Table 120. Mappable Items

ID	Description
101	Salutation Applet
102-111	Applet

DotCom View Detail

SWT Filename: dCCViewDetail.swt

This is a standard view template. It shows a parent applet, noncontext views as tabs, categorized subviews in a drop-down list, a child applet, and multiple grandchild applets. See Figure 130 for an example.

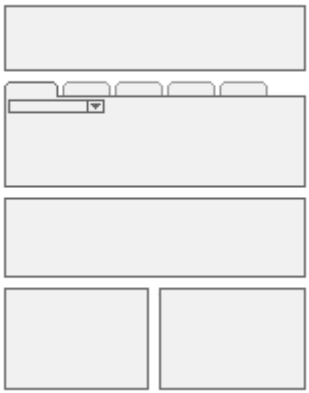


Figure 130. DotCom View Detail

Includes Tree

```
dCCViewDetail.swt
   dCCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   dCCViewbar_Tabs.swt
      CCApplet_Spacer.swt
   dCCSubViewbar_Drop.swt
   dCCHTMLFooter
```

Table 121 lists the mappable items for this template.

Table 121. Mappable Items

ID	Description
1	Parent Applet
2	Child Applet
3-5	Grandchild Applet
6-7	Grandchild Applet
8-9	Grandchild Applet
201	Mini-Applet

DotCom View Detail MultiChild

SWT Filename: dCCViewDetailMultiChild.swt

This is a standard view template. It shows a parent applet, noncontext views as tabs, child applet, categorized subviews in a drop-down list, and multiple grandchild applets. See Figure 131 for an example.

Figure 131. DotCom View Detail MultiChild

Includes Tree

`dCCViewDetailMultiChild.swt`

```
dCCHTMLHeader.swt
   CCStylesChoice.swt
CCThreadbar.swt
dCCViewbar_Tabs.swt
   CCApplet_Spacer.swt
dCCSubViewbar_Drop.swt
dCCHTMLFooter
```

Table 122 lists the mappable items for this template.

Table 122. Mappable Items

ID	Description
1	Parent Applet
2	...
3-5	Child or Grandchild Applet
6-7	Grandchild Applet
8-9	Grandchild Applet
201	Mini-Applet

DotCom View Detail2

SWT Filename: dCCViewDetail2.swt

This is a standard view template. It shows a parent applet, noncontext views as tabs, a child applet, categorized subviews as tabs, and multiple grandchild applets. See Figure 132 for an example.

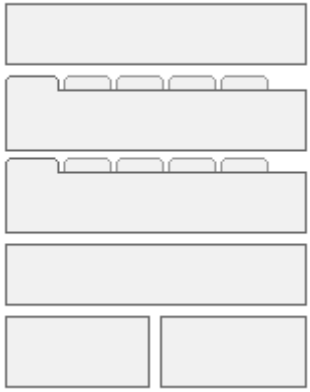


Figure 132. DotCom View Detail 2

Includes Tree

```
dCCViewDetail2.swt
   dCCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   dCCViewbar_Tabs.swt
      CCApplet_Spacer.swt
   dCCSubViewbar_Tabs.swt
      CCApplet_Spacer.sw
   dCCHTMLFooter.swt
```

Table 123 lists the mappable items for this template.

Table 123. Mappable Items

ID	Description
1	Parent Applet
2	Child Applet
3	Child Applet
4	Grandchild Applet
6-7	Grandchild Applet
8-9	Grandchild Applet
201	Mini-Applet

Page Containers

Framed Versus Unframed

All applications ship with HTML frames enabled and with a set of nonframed templates that can be applied to the customer applications.

NOTE: Siebel employee applications require frames. The removal of frames can only be done in the customer applications, and the page container is where you would do it.

Unframing Customer applications is discussed in *Siebel Interactive Designer Administration Guide*.

DotCom Page Container (Framed)

SWT Filename: dCCPageContainer_Frames.swt

This is the framed Dotcom application container page. It contains definitions for the banner, screenbar, viewbar, and content frames.

Includes Tree

```
dCCPageContainer_Frames.swt
   CCStylesChoice.swt
   dCCFrameBanner.swt
      CCStylesChoice.swt
   dCCFrameScreenbar.swt
      CCStylesChoice.swt
```

```
      dCCScreenbar_Tabs.swt
   dCCFrameViewbar.swt
   CCStylesChoice.swt
```

Table 124 lists the mappable items for this template.

Table 124. Mappable Items

ID	Description
11-18	...
19	...

DotCom Page Container (Hybrid)

SWT Filename: dCCPageContainer_Hybrid.swt

This is the Hybrid frame container. Hybrid signifies an application that takes on aspects of the Customer and Employee applications. In this case, the banner treatment is the DotCom style (no application menus). The viewbar is the Employee style (supports Search Center and History bar).

Includes Tree

```
dCCPageContainer_Hybrid.swt
   CCStylesChoice.swt
   dCCFrameBanner.swt
      CCStylesChoice.swt
   dCCFrameScreenbar.swt
      CCStylesChoice.swt
      dCCScreenbar_Tabs.swt
   dCCFrameViewbar_Hybrid.swt
      CCStylesChoice.swt
   CCFrameContent_Logic.swt
```

```
CCFrameContent_VSD.swt
CCFrameContent_VS.swt
CCFramdContent_VD.swt
CCFrameContent_V.swt
```

Table 125 lists the mappable items for this template.

Table 125. Mappable Items

ID	Description
11-19	...
21-22	...
33-34	Control
35	...
36-37	Control
38	Search Center

DotCom Page Container No Frames

SWT Filename: ddCCPageContainer_NoFrame.swt

This is the nonframed page container.

Includes Tree

```
ddCCPageContainer_NoFrame.swt
   CCStylesChoice.swt
   CCThreadbar.swt
```

Table 126 lists the mappable items for this template.

Table 126. Mappable Items

ID	Description
11-18	...
19	...

Specialized Applets

DotCom Applet Find

SWT Filename: dCCAppletSearchFind.swt

This template displays the fields for a query-based search.

Includes Tree

```
dCCAppletSearchFind.swt

   dCCFormSearch.swt
```

Table 127 lists the mappable items for this template.

Table 127. Mappable Items

ID	Description
91	Inside Applet Help Text
132	Menu
133	New

Table 127. Mappable Items

ID	Description
141-142	Control
143	Control
1101-1130	Label; Field

DotCom Applet License Base 1 Column

SWT Filename: dCCAppletLicenseBase1Col.swt

Includes Tree

```
dCCAppletLicenseBase1Col.swt
   CCApplet_NamedSpacer.swt
   CCTitle_Named.swt
      CCTitle.swt
   CCTogglebar_drop.swt
   CCBottomApplet_NamedList.swt
```

Table 128 lists the mappable items for this template.

Table 128. Mappable Items

ID	Description
2	...
91	Inside Applet Help Text
132-133	Control
141-142	Control
157	Control
158	Control
1100-1101	Label

Table 128. Mappable Items

ID	Description
1300-1301	Label
1500	Required Legend

DotCom Applet Parametric Search Head

SWT Filename: dCCAppletPSearchHead.swt

This template creates the scoping fields for a parametric search.

Includes Tree

```
dCCAppletPSearchHead.swt
   CCApplet_Spacer.swt
   CCTitle.swt
   dCCListButtonsTop.swt
      dCCButtons_List.swt
      CCRecordNav.swt
      CCTogglebar_drop.swt
      CCListButtonsTopRight.swt
   CCListBodyInverted.swt
```

Table 129 lists the mappable items for this template.

Table 129. Mappable Items

ID	Description
2	Control
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)

Table 129. Mappable Items

ID	Description
109-111	Control
121	First
122	Previous
123	Next
124	Last
131	New
134	Reset
135	Cancel
136	Save
139-141	Control
142-143	Control
144	Selected Row
145	Control
146	Control
147	Pick Control
150-151	Control
160-164	Control
161-164	Control
499	Record Title Row
501-520	Control
1100	Outside Applet Help Text
1500	Required Legend

DotCom Applet Parametric Search Tail

SWT Filename: dCCAppletPSearchTail.swt

This template creates the result list for a parametric search.

Includes Tree

```
dCCAppletPSearchTail.swt
   CCListBodyNoRowHilite.swt
   CCBottomApplet.swt
```

Table 130 lists the mappable items for this template.

Table 130. Mappable Items

ID	Description
142-143	Control
144	Selected Row
145	Control
146	Save
147	Pick Control
501-540	Field

DotCom Applet Realtime Cart

SWT Filename: dCCAppletRealtimeCart.swt

This template creates the real-time shopping cart applet.

Includes Tree

```
dCCAppletRealtimeCart.swt
```

Table 131 lists the mappable items for this template.

Table 131. Mappable Items

ID	Description
132-133	Control
133	Control
140	Icon; Mapped Title; Item Name; Quantity; Line Items Label; Line Items Field: Total Price Label; Total Price Field
141	Control
500	Mapped Title
501	Item Name
502	Quantity
1222	Line Items Label
1223	Total Price Label
1322	Line Items Field
1323	Total Price Field
2222	Label
2223	Label
2322	Field
2323	Field

DotCom Applet Search Advanced

SWT Filename: dCCAppletSearchAdvanced.swt

This template displays the fields for an advanced search.

Includes Tree:
`dCCAppletSearchAdvanced.swt`

dCCFormSearch.swt

Table 132. Mappable Items

ID	Description
91	Inside Applet Help Text
132	Menu
133	New
141-142	Control
143	Control
1101-1130	Label; Field
1199	Label

DotCom Applet Search Advanced Tabbed

SWT Filename: dCCAppletSearchAdvancedTabbed.swt

This template displays the fields for an advanced search and includes the standard applet tab treatment.

Includes Tree:

dCCAppletSearchAdvancedTabbed.swt

CCTitle.swt

dCCFormSearch.swt

Table 133. Mappable Items

ID	Description
91	Inside Applet Help Text
132	Menu
133	New
141-142	Control

Table 133. Mappable Items

ID	Description
143	Control
1101-1130	Label; Field
1199	Label
1500	Required Legend

DotCom Applet Search Basic

SWT Filename: dCCAppletSearchBasic.swt

This template displays the fields for a basic search.

Includes Tree

```
dCCAppletSearchBasic.swt

   dCCFormSearch.swt
```

Table 134 lists the mappable items for this template.

Table 134. Mappable Items

ID	Description
91	Inside Applet Help Text
132	Menu
133	New
141-142	Control
143	Control
1101-1130	Label; Field

DotCom Applet Totals

SWT Filename: dCCAppletFormTotals.swt

This is a specialized form template. It is used to create multiline form totals information that are found beneath a quote or order.

Includes Tree

```
dCCAppletFormTotals.swt

   dCCFormTotals.swt

   dCCButtons_Form.swt
```

Table 135 lists the mappable items for this template.

Table 135. Mappable Items

ID	Description
2	Back
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-110	Control
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
157-158	Control
1112-1117	Field; Label

DotComp Applet List Form

SWT Filename: dCCAppletListForm.swt

Includes Tree

dCCAppletListForm.swt

Table 136 lists the mappable items for this template.

Table 136. Mappable Items

ID	Description
90	Title
106	Find
107	Search
109-111	Control
131	New
132-133	Control
134	Reset
135	Cancel
136	Save
142	Control
144	Selected Row
146	Save
147	Pick Control
180	Div
184	DrillDown Title
501	Field
502-521	Field
1100	Outside Applet Help Text

Cascading Style Sheets 8

The topics in this section describe in general terms the CSS class names in use in Siebel Systems Web applications and the interface elements they control. A working knowledge of CSS and HTML is assumed.

The CSS settings discussed in this documentation can be found in the files called main.css and dCCmain.css or the Employee and Customer Style Sheets, respectively. In general the class names declared in these style sheets use CSS Level 1 attributes. The attributes most often manipulated are font characteristics, link characteristics, and background colors.

Expand the Cascading Style Sheets list in the Contents tab to see a list of links to descriptions of general declarations.

Body, Td, Input, Select, Textarea, A

Controls font-family and size as well as default link color.

MVG Format Definitions

.mvgBorder, .mvgBack

Controls the background color and border of forms and lists appearing in pop-up windows.

Global Menu Definitions

.globalMenu

Controls the appearance of page item links that are attached to the Page Container and typically shown in the banner areas within DotCom applications.

Navigation Tabs

tier1Back, .tier1Rule, .tier1On, .tier1Off

Controls the background and link colors of screen-level tabs.

tier2Back, .tier2Rule, .tier2On, .tier2Off

Controls the background and link colors of elements appearing in the viewbar frame including second-level show drop-down, History drop-down, Dashboard Icon, Favorites drop-down, and Search Center Icon.

tier3Back, .tier3Rule, .tier3On, .tier3Off

Controls the background and link colors of detail tabs, also known as noncontext views.

tier4Back , .tier4Rule, .tier4On, .tier4Off

Controls the background and link colors of subview tabs, also known as grouped views.

Threadbar

.threadbar, .threadbarDiv

Controls the link characteristics of the threadbar that appears at the top of many views. ThreadbarDiv controls characteristics of the separators between thread links.

List Definitions

.Row, .RowCenter, .RowRight

Controls the background color, text color, and alignment characteristics of field values as shown within list rows.

.listRowOff

Controls the background color of a deselected row.

.listRowOn

Controls the background color of a selected row in Standard Interactivity mode.

.listRowError

Controls the background color of a row which produced an error during submission. Not used in the release templates.

.listRowEven, .listRowOdd

Controls the background color of even or odd rows respectively. Not used in the release templates.

.ListBorder

Establishes a border around all lists. In HI mode this border is used to display a highlight when the applet has been selected.

.Header

Controls the font and alignment characteristics of field column headers.

Login Page Definitions

The following settings control the look-and-feel of the HTML-based login page that is viewed when logging into the Siebel Web server from a supported Web browser. They do not affect the login page as seen in other contexts (for example, when starting up the mobile client).

.loginTop, .loginMid, .loginBtm

Controls the background colors of the three regions found in the HTML-based login page.

.loginAppTitle

Controls the font and color characteristics of the application title.

.loginError

Controls the color of the error message should a login attempt produce an error.

.loginCopy

Controls the color characteristics of the copyright text.

.loginForm, .loginBody, .loginText

Controls the general text characteristics of the login page.

.loginLabel

Controls the specific characteristics of the labels associated with login fields.

.loginField

Controls the characteristics of login fields.

Banner Definitions

The banner is the topmost element in the visible UI. It contains global navigation elements and the Powered by Siebel logo.

.banner

Controls the text, line, and background characteristics of the banner area.

.bannerDiv, .bannerDiv2, .bannerDiv3

Controls the color-characteristics of bevel that appears above and below the banner elements.

Message Layer

Used by the Communications toolbar to display lengthy status messages.

#MsgLayer

Defines a DOM-accessible region that the toolbar can access in order to insert text messages.

.Message

Defines the text and background characteristics of the message displayed to users.

Mini-Button Definitions

.minibutton, .minibuttonOn, .minibuttonOff

Defines the text, link and background characteristics for both On and Off states. See the SWF file for more information about how these states are declared.

SmartScript Definitions

.smartDialog

Controls the font and background characteristics of SmartScript forms.

Search Center Definitions

.SrchCntrTitle

Controls the font characteristics of the Search Center applet title, which is slightly larger than normal applet titles.

Single-Column (sc) Form Mode

The single-column form-elements are used primarily in DotCom applications. It is characteristic for labels to appear beside fields in these applications.

.scLabel

Controls the font characteristics of the label in a label/field pair.

.scLabelRight

Controls the font characteristics of the label in a label/field pair. The class forces the label to appear right-aligned. When running in right-to-left languages this class name should be changed to align left.

.scField

Controls the font characteristics of fields.

Multi-Column Editable (mce) Form Mode

The multi-column form element is used primarily in Employee applications. To conserve space labels typically appear above fields. In addition, field width is set so that fields display with a uniform width. The mce class names affect only the display of fields running in Standard Interactivity mode. If the application is run in High Interactivity mode, the generated controls determine field size based upon internal dimensioning algorithms.

.mceLabel

Controls the font characteristics of the label in a label/field pair.

.mceField

Controls the font characteristics of fields. For Employee applications fields declared within this class names scope will display 190 pixels wide. For DotCom applications the field width is declared as 120 pixels.

.mceReadOnly

Controls the characteristics of disabled text. Not used in this release.

.mceWideFields

Controls the font characteristics of wide fields, that is, fields that are mapped in placeholders that span two or four columns. The field will size to fit the width allotted to it. Again, this is only true if the application is running in Standard Interactivity mode. For Employee applications running in High Interactivity mode, the generated controls determine field width.

Rich Text Component Classes

.rtcEmbedded, .rtcPopup, .rtcReadOnly, .rtcTextarea

Controls the dimensions and font characteristics of containers that hold the Rich Text Component.

Layout Styles

.LayoutButtonOn, .LayoutButtonOff

Controls the background color of layout buttons. Can be used with transparent GIFs to create two button states. Not used in this release.

.LayoutView, .LayoutStyleHide, .LayoutStyleMax,.LayoutStyleMin

Used to define the color and font characteristics of applets when shown in layout mode.

Applet Select

.Selected TD.AppletHIFormBorder

Controls the border-color of applets that have been selected. When an applet is encased with a tag that implements the .Selected class, the applet border highlight is turned on.

.Selected TD.AppletHIListBorder

Controls the border-color of applets that have been selected. When an applet is encased with a tag that implements the .Selected class, the applet border highlight is turned on.

Applet Style

The style sheet declares eight applet styles. Each style declares substyles; through inheritance, it is possible to switch the root applet style and receive all the other substyle differences automatically. A description of the substyles follows.

.AppletStyle#

Controls number one through eight. The .AppletStyle# is the root class or logical CSS container for an applet.

.AppletButtons

Controls the font and color characteristics of elements that appear on the button bar.

.AppletBorder

Controls the border characteristics of the applet. The border is defined as the rectangular region around the form or list, not including tabs or button bars.

.AppletHIFormBorder, .AppletHIListBorder

Controls the highlight color of the applet when it is selected. See the section Applet Select.

.AppletBlank

Controls the background color of empty space in the applet. Used to make sure that the area to the right of the applet tab appears white (not the background color of the form or list).

.AppletBack

Controls the background color of the form or list.

AppletTitle

Controls the background color, text, and link characteristics of applet titles. These titles usually appear within tabs at the top of the applet.

Calendar Definitions

.calendarBorder

Controls the external border for the Calendar applet.

.calendarActivityBack

Defines the characteristics for a given slot in the Daily or the Weekly Calendars (that surrounds a range of activities inside).

.calendarActivity

Defines the characteristics for a *single day* activity being displayed in the Calendar (Monthly, Weekly or Daily).

.calendarMultiDayActivity

Defines the characteristics for a *multiday* activity being displayed in the Calendar (Monthly, Weekly or Daily).

.calendarInterval

Defines the characteristics for an "interval" in any of the calendars, which is an hour for the Daily, a day for the Weekly, and a week in case of the Monthly.

.calendarDayBar

Defines the characteristics for the header portion of a given day in the Monthly. This is used when the day is not the current "view day" for the Monthly.

.calendarDay

Defines the external characteristics for the header portion of a day in the Monthly calendar. This is one level above (in scope) the CalendarDayBar and CalendarDayBarDark.

.calendarBorder2

Defines the border for the Home Page Calendar applet (which has a different look and feel).

.calendarDayBarDark

Defines the characteristics for the header portion of a given day in the Monthly. This is activated when the day is the current "view day" for the Monthly in order to highlight it.

Service Calendar Definitions

.ServiceCalRow
Defines the external border for the Service Calendar applet.

.calendarServiceBorder
Defines a new row in the Service Calendar.

.calendarServiceActivityOn
Defines an interval with the activity in it.

.calendarServiceActivityOff
Defines an interval without activity. It is a blank cell.

Tree Style

.treeBack
Controls the background and border characteristics of a tree applet.

.treeInactive
Controls the link characteristics of a nonselected node.

.treeActive
Controls the link characteristics of the selected node.

DotCom (Customer Applications) Definitions

DotCom (Customer Applications) definitions are included in both main.css and dCCmain.css so that they apply cases in which a DotCom view is shared with an employee application.

.dCCItemTitle

Controls the text characteristics of product titles.

.dCCItemLabel, .dCCItemLabelLeft

Controls the text characteristics of labels in product forms and lists. For RTL languages the alignments found in these classes should be reversed.

.dCCItemValue

Controls the text characteristics of product field values.

.dCCItemValue150

Controls the text characteristics of product field values. Forces the field to be a width of 150 pixels.

.dCCAppletRule1

Controls the size and color characteristics of rules in DotCom applets.

.dCCAppletShade1

Controls the background characteristics of DotCom applets that declare shaded backgrounds.

.dCCAppletBorder1

Controls the border characteristics of rules in DotCom applets.

.dCCAppletTitle

Controls the text, line, and background characteristics of applet titles in DotCom applets.

Dashboard Definitions

.dashbrdBorder

Controls the border-color and size of the dashboard. Currently not in use.

.dashbrdBack

Controls the background color of the dashboard.

Site Map Definitions

.screenName1
Controls the link characteristics of the screen link as seen on the top of the site map page.

.screenName2
Controls the link characteristics of the screen link as seen lower in the site map page.

.viewName
Controls the link characteristics of the view link.

.fader
Controls the background characteristics of the rule that divides sections within the site map.

Table of Contents Definitions

.TOCRule

Controls the size and color of the horizontal rule that appears beside the title in the table of contents categorized applet.

.TOCTitle

Controls the title text that appears on pages that use the table of contents categorized applet.

Page Header Definitions

.PageHeader

Controls the page title text that appears on some DotCom home pages.

.PageRule

Controls the size of the horizontal rule that appears beside the page header text.

Miscellaneous Definitions

.Required
Controls the color of required text. In this release, required is not text but a symbol, so this class is not used. However, it is reserved for future use.

.Welcome
Controls the text characteristics of the greeting text that appears within salutation applets.

.CmdTxt , .CmdTxtNormal
Controls text characteristics of the outside applet text.

.error
Controls text characteristics of the error text as seen in error messages at the top of forms and lists.

.divider
Controls the border definition for the divider that appears between elements on the viewbar frame.

External Content (EC)

EC:News

.NewsTable
Controls the table definition in the Headlines section.

.NewsCompanyName A
Controls the Company Name in the Headlines section.

.NewsTimeStamp
Controls the Time Stamp in the Headlines section.

.NewsBullet
Defines the style for the appearance of the Bullets.

.NewsHeadline
Defines the style definition for each headline.

.NewsSource
Defines the style for the news source description string.

.NewsSummary
Defines the style definition for each Summary headline.

.NewsTotalStories
Defines the style definition for Story Count.

.NewsPageControl
Defines the style definition for Navigation.

.NewsColumnLogo
Defines the style definition for provider LOGO.

ePortal Definitions

Search Pages

.ExSearchTable

Defines the style for table definition in Websearch, Mapsearch and Yellowpages.

.ExSearchLabel

Defines the style for each row in Websearch, Mapsearch and Yellowpages.

.ExSearchField

Controls the font and width characteristics of fields that appear in search applets such as Websearch, Mapsearch, and Yellowpages.

Company Header

.CompanyName

Defines the style for Company Name in Company Briefing.

.CompanyLink

Defines the style for Company Homepage-URL in Company Briefing.

Profile Data Definitions

.ProfCopyright

Defines the style for CopyRight text in DNB Profile in Account/Company Briefing.

.ProfLabel

Defines the style for each Label element in DNB profile.

.ProfData

Defines the style for each Data element in DNB profile.

.ProfSection

Defines the style for each Section Header in DNB profile.

Subsidiary

.SubsidCompanyName

Defines the style for Parent Company in Corporate-Relations in Account/Company Briefing.

.SubsidSubsidName

Defines the style for Child Company in Corporate-Relations in Account/Company Briefing.

.SubsidCopyright

Defines the copyright text in Corporate-Relations in Account/Company Briefing.

.SubsidDisclaimer

Defines the disclaimer text in Corporate-Relations in Account/Company Briefing.

.SubsidCompleteList

Defines the style for Link which gives the complete Corporate-Relations in Account/Company Briefing.

Add Info

.AddInfoBullet

Defines the style for Bullet in Additional info section in Account/Company Briefing.

.AddInfoHeader

Defines the style for each Section Heading in Additional info section in Account/Company Briefing.

.AddInfoWebSite

Defines the style for each link in Additional info section in Account/Company Briefing.

.AddInfoDescr

Defines the style of the Description text for each link in Additional info section in Account/Company Briefing.

Search

.CnsHead

Defines the style for the Header in CompanySearch results screen.

.CnsRow
Defines the style for each result line in CompanySearch results screen.

.CnsLink
Defines the style for the Acct Topic Management LINK in DUNS assignment (ATM) screen.

PMD Reports

.reportTxtLarge
Controls the font characteristics of report titles within PMD reports.

.reportTxtMedium
Controls the font characteristics of section titles within PMD reports.

.reportBox
Controls the decorative borders shown around some sections within PMD reports.

.reportLbl
Controls the font characteristics of field labels within PMD reports.

.reportLblBtmBorder
Controls the decorative borders shown around some list rows within PMD reports.

.reportLblTopBorder
Controls the decorative borders shown around some list rows within PMD reports.

.reportLblBtmBorder
Controls the decorative borders shown around some list rows within PMD reports.

.reportLblTopBorder
Controls the decorative borders shown around some list rows within PMD reports.

.reportFld
Controls the font characteristics of field values within PMD reports.

.reportObjNum
Controls the font, border, and background characteristics of objectives section numbers as they appear in the All Objectives report.

.reportObjTitle

Controls the font, border, and background characteristics of objectives section titles as they appear in the All Objectives report.

.reportAllObjTitle

Controls the font characteristics of the report title as it appears in the All Objectives report.

.reportAllObjSubtitle

Controls the font characteristics of the report subtitle as it appears in the All Objectives report.

Operators, Expressions, and Conditions 9

This links below provide access to topics that describe the supported syntax elements for queries and for sort and search specifications.

Precedence

Precedence is the order in which Siebel applications evaluate the various operators within a single expression. The system will evaluate operators with higher precedence before operators with lower precedence. In addition, operators with equal precedence will be evaluated left to right.

The levels of precedence for the various Siebel application operators are listed below.

Level	Operator
1	()
2	*
3	+, -, NOT logical operator
4	AND logical operator
5	OR logical operator

The order of precedence within an expression can be altered by using parentheses. Siebel applications evaluate the expression within the parentheses first, before evaluating the expression outside.

Comparison Operators

The table below describes the purpose of each comparison operator and gives an example of how it is used.

Operator	Purpose	Example
=	Equality test	[Last Name] = “Smith”
< >	Inequality test	[Role] < > “End-User”
>	Greater than	[Revenue] > 5000
<	Less than	[Probability] < .7
> =	Greater than or equal to	[Revenue] > = 5000
< =	Less than or equal to	[Probability] < = .7

Logical Operators

The table below explains what a value of TRUE or FALSE means for each logical operator.

Operator	Returns TRUE	Returns FALSE
NOT	If the condition evaluates to FALSE	If the condition evaluates to TRUE
AND	If all component conditions evaluate to TRUE	If any component condition evaluates to FALSE
OR	If any component condition evaluates to TRUE	If all component conditions evaluate to FALSE

Arithmetic Operators

The table below describes the purpose of each arithmetic operator and gives an example of how it is used.

Operator	Purpose	Example
+	Add	[Record Number] + 1
-	Subtract	[Record Number] - 1
*	Multiply	[Subtotal] * 0.0625
/	Divide	[Total Items] / [Total Orders]
^	Exponent	[Grid Height] ^ 2

Pattern Matching with LIKE and NOT LIKE

NOTE: The Search Engine Table property for View and Applet must be based on the same table as the index. For example, if the search index is based on S_EVT_ACT, then the view should be based on action and the applet should be based on action.

The LIKE operator is used in character string comparisons with pattern matching. The syntax is as follows:

```
char1 LIKE char2
```

where *char1* is the value to be compared with the pattern and *char2* is the pattern to which *char1* is compared. The NOT logical operator can be used in conjunction with LIKE to exclude patterns. The syntax including the NOT logical operator is:

```
char1 NOT LIKE char2
```

or

```
NOT (char1 LIKE char2)
```

While the equal (=) operator does exact matching, the LIKE operator matches a portion of one character value to another. Patterns can use special characters to denote different characters. These characters are given in the following table.

Character	Purpose	Example
*	Zero or more characters	[Last Name] LIKE “Sm*” would return all records whose [Last Name] value starts with the characters “Sm”, as in “Smith”, “Smythe”, “Smart”, and so on. [Last Name] LIKE “*om*” would return all records whose [Last Name] field contains the characters “om”, as in “Thomas”, “Thompson”, “Tomlin”, and so on.
?	One character	[First Name] NOT LIKE “Da?” would return all records whose [First Name] value was three characters long and did not start with the letters “Da”. Records with “Ted”, “Tom”, and “Sam” would be returned, but “Dax” and “Dan” would not. NOT ([First Name] LIKE “?o?”) would return all records whose [First Name] value was three characters long and did not have as its middle character “o”. Records with “Ted” and “Sam” would be returned, but “Tom” and “Bob” would not.

Character is case-sensitive when executing pattern matching. In addition, the parentheses are required when including the NOT logical operator (the second variation of the NOT syntax).

NOTE: On occasion, using the wildcard * to find all entries in a field can cause a performance problem. If it does, use IS NOT NULL instead. For more information, see “NULL” on page 656.

NULL

NULL in SQL represents a value that is not known or is not applicable. Expression evaluation with NULL is somewhat different than with other values. Since NULL is not a value, comparison functions do not operate normally when one or both of the operands are NULL. For instance, NULL = NULL is not TRUE.

SQL and Siebel applications provide special functions and grammar to support NULL, including the IS NULL unary operator and IfNull function. Comparisons, string concatenations, and Boolean operations have special behavior to handle NULL.

NULL is typed like a value. An operand or result can be NULL string, NULL number, NULL Boolean, and so on.

IS NULL Unary Operator

The = operator is not useful in determining whether a value is NULL because the value of a NULL operand is unknown. Siebel applications provides the IS NULL operator, which evaluates to TRUE if its operand is NULL and to FALSE if its operand is not NULL.

IfNull Function

The IfNull function has two arguments and returns the value of either the first or second argument depending on whether the first argument is NULL. IfNull *(a,b)* returns *a* if *a* is not NULL or returns *b* if *a* is NULL.

The return type of IfNull is the type of its first argument, even if the first argument is NULL. The second argument is converted to the type of the first argument before its value is returned.

Comparisons with NULL

When either side of a comparison is NULL, the comparison returns NULL of type Boolean. Otherwise, the comparison returns TRUE or FALSE. For example, 1 > 2 is FALSE, and 1 < NULL is NULL.

Flag Fields and NULL

Use caution when querying flag fields. The comparison operators < > and NOT IN do not allow the evaluation of fields that are null. Since flag fields are defaulted to null, a workflow condition of < > '*Y*' will not work. There are three ways to work around this problem:

- Use IS NOT NULL as comparison operator.
- Use IN ('N',NULL).
- Predefault the business component field to 'N'.

Arithmetic Operations with NULL

When either side of an arithmetic operation is NULL, the operation returns NULL of the appropriate type, except for string concatenation. In a string concatenation operation, NULL simply adds no characters. For example, 1 + 2 is 3, 1 + NULL is NULL (of type Integer), “Fred” + “, Smith” is “Fred, Smith”, but “Fred” + NULL is “Fred.”

Functions in Calculation Expressions

Calculation expressions are calculated field and validation expressions.

- "Using Calculated Fields with Chart Coordinates" on page 665
- "Using Julian Functions" on page 666
- "Calculated Field Rules" on page 667
- "Example of String Concatenation and the IIf Function" on page 667
- "Syntax for Predefault and Postdefault Fields" on page 668

NOTE: AccountId (), ContactLogin(), JobTitle(), and OrganizationId() are meant to be used as predefault value and postdefault value in business components. They are not supported in Siebel VB or through COM objects.

You cannot use custom functions in calculated expressions.

Use only numbers between -2147483647 and 2147483648 in field validation expressions.

Table 137 describes the functions you can use in these expressions.

Table 137. Functions Used in Calculated Expressions

Function	Result Type	Query	Description
AccountId()	String	Yes	Returns the current user's Account ID (OU_ID).
ContactLoginId()	String	Yes	Returns the contact ID of the currently logged in user. If you do not use the contact login method for a Web-based application, the function cannot retrieve any value and returns an empty string. It is recommended that you use the contact login method and an external security authentication service (for example, LDAP).

Table 137. Functions Used in Calculated Expressions

Function	Result Type	Query	Description
Count ("*mvlink*")	Integer	No	Returns the number of rows in the multi-value group defined by the MVL *mvlink*.
Currency ()	String	Yes	Returns the currency code for the current position (for example, USD).
DivisionId ()	Integer	Yes	Returns the current user's Division ID (BU_ID). To limit visibility to employees from the same division as the person logged in, add the following to the search specification property of the Applet: [Division Id] = DivisionId()

Table 137. Functions Used in Calculated Expressions

Function	Result Type	Query	Description
DivisionName ()	String	Yes	Returns the division name of a user who is an employee. Use to limit visibility to employees from the same division as the person logged in. Also use to display the division name of the user logging the service request. Create a new calculated field so that, when the service request is created, the calculated field displays the division name of the current logged user that is creating the service request. Using the following configuration, the new joined field Reported By Division will be predefaulted to this value, and will never receive another value after this service request creation event. To create a calculated field that displays the division name of the current logged user creating a service request: **1** In the Service Request business component, create a new calculated field: Calculated: TRUE Calculated Value: DivisionName() Name: Division (Calc) Parent Name: Service Request Type: DTYPE_TEXT **2** In the Service Request Business Component, also create a new join to S_SRV_REQ_X table: Column: ATTRIB_03 Join: S_SRV_REQ_X Name: Reported By Division

Table 137. Functions Used in Calculated Expressions

Function	Result Type	Query	Description
DivisionName () (*continued*)			Pre Default Value: Field: 'Division Name' Read Only: TRUE Expose the joined field Reported By Division in the relevant applets. You may also want to expose the calculated field Division (Calc), just to check the logic and set Visible = False later for the control or list column exposed.
EXISTS	String	Yes	For example: IIf(EXISTS([Participant-Employee Login] = LoginName()), "Y", "N").
GetProfileAttr ("*Attribute*")	String	Yes	Returns the value stored in the profile attribute if that attribute has been defined. Used in personalization to retrieve values of attributes in a user profile and to pass information from a script to the UI. Set a session-specific personalization attribute equal to the value of the shared global and reference the personalization attribute in a calculated field. **NOTE:** For an undefined attribute or for an attribute that has not been set up, GetProfileAttr returns NULL. This is important when you are using comparison operators. For example: ■ GetProfileAttr("Attribute") = "" always returns FALSE either if the Attribute does not exist or exists and the value is different than "". ■ GetProfileAttr("Attribute") IS NULL returns TRUE if the Attribute does not exist and FALSE otherwise.
IfNull (*expr1*, *expr2*)	Type of *expr1*	Yes	Returns the value of *expr1* unless *expr1* is NULL, in which case the value of *expr2* is returned.

Table 137. Functions Used in Calculated Expressions

Function	Result Type	Query	Description
IIf (*testExpr*, *expr1*, *expr2*)	Type of *expr1*	No	If *testExpr* is TRUE, returns the value of *expr1*; otherwise returns the value of *expr2*. **NOTE:** If working with DTYPE_NUMBER fields, the Data Type of *expr1* determines the Data Type of the resulting value.
InvokeServiceMethod ("[ServiceName]", "[MethodName]", "[InputProp1 = val1, InputProp2 = val2]", "[OutputProp")	String	No	Invokes a business service from a calculated field and returns [Output Prop]. **NOTE:** Do not expose a calculated expression that invokes a business service in a list applet. Doing so may result in poor performance because the business service will be repeatedly instantiated each time the field appears in the list.
JobTitle ()	Integer	Yes	Returns the Job Title of the currently logged-in employee. Similar to PositionId () and DivisionId ().
JulianDay ()	Date	Yes	Equal to the Oracle (and Sagent) Julian Day, for all dates in the 20th and 21st centuries.
JulianMonth ()	Date	Yes	Equal to the JulianYear() * 12 + currentMonth, where January = 1.
JulianQtr ()	Date	Yes	Equal to the JulianYear() * 4 + currentQuarter, where currentQuarter = (currentMonth - 1) / 3 + 1 rounded down to the next integer.
JulianWeek ()	Date	Yes	JulianDay() / 7, rounded down to the next integer.
JulianYear ()	Date	Yes	Equal to the current year + 4713.
Language ()	String	Yes	Returns the language code (for example, ENU) which is the active client language setting, set by the Language parameter in the CFG file, or by the /L parameter when starting a Siebel application. **NOTE:** This is not the Resource Language parameter.
Left (*text*, *integer*)	String	Yes	Returns the leftmost *n* characters in the text string or field. For example, Left (“Adams”, 2) returns “Ad.”

Table 137. Functions Used in Calculated Expressions

Function	Result Type	Query	Description
Len()	String	Yes	Returns the length of a string or string variable. The number of characters is specified between parentheses.
LocalCurrency ()	String	Yes	Returns the currency code for this machine (for example, JPY).
LoginId ()	String	Yes	Login ID (for example, 0-3241).
LoginName ()	String	Yes	Login name (for example, BSTEVENS).
Lookup (*type*, *value*)	String	No	Finds a row in S_LST_OF_VAL where the TYPE column matches the *type* argument and the VALUE column matches the *value* argument. The function returns the value of the ORDER_BY column for that row. The primary purpose of the Lookup function is to avoid additional joins in a business component.
LookupExpr (*type*, *value_expr*)	String	No	Searches the rows in S_LST_OF_VAL where the TYPE column matches the *type* argument. Evaluates the contents of the VALUE column treated as an expression. Returns the value of the ORDER_BY column for the first row for which the expression evaluates to TRUE. The LookupExpr function essentially performs an in-memory linear parse evaluate search, so you should make sure that there are fewer than 30 rows in the LOV type.
LookupName (*type*, *lang_ind_code*)	String	Yes	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. Returns the language-independent code (the CODE column) for the row. This function is used to obtain the *untranslated* value in the specified LOV.

Table 137. Functions Used in Calculated Expressions

Function	Result Type	Query	Description
LookupValue (*type*, *lang_ind_code*)	String	No	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. Returns the display value (the VAL column) for the row. LookupValue tries to find the display value for the specified *lang_ind_code*. If not found, LookupValue just returns the *lang_ind_code* itself as the value. This function is used to obtain the translation of the specified untranslated value in the specified LOV into the currently active language. **NOTE:** The LookupValue() function cannot be used directly in the Pre Default Value property of a field. Instead, use a separate calculated field for the lookup, and reference the calculated field in the Pre Default Value property of the original field.
OrganizationId ()	Integer	Yes	Returns the organization ID of the currently logged in user.
OrganizationName ()	String	Yes	Returns the organization name of a user who is an employee.
ParentBCName ()	String	Yes	Parent (master) business component name for active link (for example, Opportunity).
ParentFieldValue (*field_name*)	String	Yes	Returns the value of the *field_name* field in the parent business component. The result is not typed correctly but is always of type String. Also, the result does not change if the parent row is updated. The parent business component field must be exported by using Link Specification = TRUE.
PositionId ()	String	Yes	Position ID of currently logged-in employee (for example, 0-4432).
PositionName ()	String	Yes	Position name of currently logged-in employee.
Right (*text*, *integer*)	String	Yes	Returns the right-most *n* characters in the text string or field. For example, Right (“Adams”, 2) returns “ms”

Table 137. Functions Used in Calculated Expressions

Function	Result Type	Query	Description
RowIdToRowIdNum ([*Id*])	String	Yes	Converts an alphanumeric row ID to a unique, pure numeric row ID in the Service Request business component. In Siebel eBusiness 6.*x*, this expression has changed functionality; it is used for the predefault value of the "SR Number" field.
Sum ([*mvfield*])	Integer	No	Sums the values from a field in child records into a field in a parent record. The child record being summed from must be defined as a multivalue field that is part of a multi-value group that is associated with the business component of the field being summed to.
ToChar (*num_expr*, *format*)	String	No	Returns a string that represents a number or date in a format specified by the optional format parameter. (For example, ToChar (10, "##.##") returns "10.00").
Timestamp ()	Date Time	Yes	Today's date and time (for example, 01/02/96 11:15:22). The Timestamp function can also be used in queries. For example: `Created >= Timestamp() - 0.1` against an MS SQL Server database would return those records created within the last one-tenth of a day.
Today ()	Date	Yes	Today's date (for example, 1/26/96).

Using Calculated Fields with Chart Coordinates

Suppose you want to set the following coordinates:

0-200

200-999

1000-4999

5000-24999

25000 +

To set the coordinates

1. Create a calculated field that has the one relevant value of the five coordinate values (for example, if the record has a value of 500, the calculated field's value will be “200-999”).
2. To see the coordinates in the chart in the order you want, create a list of values that has the five values listed above.

Using Datetime Fields in Calculations

It is possible to perform calculations with datetime fields in calculated fields. When a number is entered in a datetime field, days are represented by integers and hours, and minutes and seconds are represented by fractions.

For example, to add one minute to the current date and time, use the following expression, which is derived from the fact that one day has 1440 minutes:

```
Timestamp() + 1/1440
```

In this example the product delivery interval, measured in seconds, is added to the current date and time:

```
Timestamp() + [Product Delivery Interval]/86400
```

NOTE: The Type property of the calculated field must be of type DTYPE_DATETIME.

Using Julian Functions

The Julian functions must include Today() or a field name as a parameter.

For example, you need to use either JulianMonth([Created]) (of a field) or JulianMonth(Today()) (of the current date).

Calculated Field Rules

Calculated fields have the following rules and restrictions:

- Calculated fields do not support updates (even simple expressions like [Field]), unless specialized business components override SqlSetFieldValue.
- Calculated fields cannot be stored in columns.
- Validation criteria on calculated fields are ignored.
- Queries on calculated fields are *always* supported.
- Sorting on calculated fields is *never* supported.
- When a query is performed on a calculated field, the action taken by your Siebel application (and thus the resulting performance) depends on which functions are used within the calculation. Functions that can be incorporated directly into the WHERE clause in the SQL statement are incorporated. Functions that cannot be directly incorporated—such as IIf() and Lookup()—result in testing each record in the business component to determine which records to display to the user, at a considerable performance cost.
- A calculated field may be based on the results of another calculated field in the same business component. There is no limitation on the number of levels of calculated fields based on other calculated fields.

Example of String Concatenation and the IIf Function

In the following example of assigning expressions to the Calculated Value property, the string constant "," must be enclosed in two double quotation marks because the entire value is quoted with double quotation marks. If the [Last Name] field is NULL and [First Name] is "Bob", the [Full Name] field contains ", Bob."

```
[Field]
Name = "Full Name"
TextLen = 102       // Last Name + First Name + 2
Calculated = "TRUE"
CalculatedValue = "[Last Name] + "," + [First Name]"
```

Alternatively, the next expression contains just "Bob" if the [Last Name] field is NULL and the [First Name] field is "Bob". (The CalculatedValue expression must be all on one line.)

```
CalculatedValue = "[Last Name] + IIf ([Last Name] IS NULL,
"", ",") + [First Name]"
```

Syntax for Predefault and Postdefault Fields

The Pre Default Value property of a field (Predefault Value in the Object List Editor) automatically assigns a value to that field for a new record. The user can modify the field if it is displayed and not set to Read Only. For example, Currency Code has a predefault value of System: Currency. The currency code for a new contact is automatically set to the default system currency.

The Post Default Value property of a field assigns a value to a field before the record is written to the database, if one has not been entered by the user. For example, Personal Contact has a postdefault value of N. If the user does not designate a new contact as personal, the system assumes that it is not.

Table 138 provides the syntax to be used in predefault fields.

Table 138. Predefault Values and Functions

Function	Result Type	Description
System: Creator	String	Login name (for example, BSTEVENS).
System: CreatorId	String	Login Id (for example, 0-3241).
System: OrganizationId	String	Organization ID (for example, 1-24E1).
System: OrganizationName	String	Organization name (for example, Siebel Service).
System: Position	String	Position name (for example, VP of Sales).
System: PositionId	String	Position Id (for example, 0-4432).
System: Today	Date	Today's date (for example, 1/26/96). When using System: Today as the predefault value for a field, the data type for that field must be DTYPE_DATE.
System: Timestamp	DateTime	Today's date and time (for example, 01/02/96 11:15:22).

Table 138. Predefault Values and Functions

Function	Result Type	Description
System: Currency	String	Currency for this position (for example, USD). Determined by the setting for the Currency field in the Divisions or Organizations view under the Group Administration screen. If the division has a different Currency setting from the organization, the division Currency setting will be used.
System: LocalCurrency	String	Currency for this machine (for example, JPY).

Parent: 'BusComp.Field', 'BusComp.Field'	String	Value in parent business component field.
		The field in the parent business component must have Link Specification set to TRUE for values to be defaulted.
		You can have multiple 'BusComp.Field' constructs separated by commas; the list is checked from first to last until a value is found, for example:
		`Parent: 'ServiceRequest.Account', 'Account.Name'`
		NOTE: A space is required after every comma that separates the fields for this function to work correctly. If the business component has an apostrophe in its name, you must enclose the name in double quotes, for example:
		`Parent: "FINS AG Agent's Contracts.Status Of Contract"`
		You can also terminate a chain of Parent calls with a System call, for example:
		`Parent: 'Opportunity.Currency Code', 'Account.Currency Code', System: Currency`

Table 138. Predefault Values and Functions

Function	Result Type	Description
Field: ‘FieldName’	String	Value in field in current business component field “FieldName.” Field: 'FieldName' does not work in the Predefault Value property if FieldName is a join field.
Expr: ‘Today() - 1’	String	Value of expression. Example: Today() - 1.

Calculated Field Values and Field Validation

The Calculated Value property specifies an expression for calculating the value of a field. A field's Validation property allows you to restrict the values for a single value field (validation is not supported for MVFs). The validation property is evaluated when the field is accessed and modified in the GUI only. The validation properties from the current applet's business component fields are evaluated. If these fields are part of other business components as well, those validation properties are not evaluated.

NOTE: The Calculated Value and field Validation properties are limited to 255 characters.

The syntax for the Calculated Value or Validation property is the same as the QBE syntax, with different but overlapping functions. The comparison, logical AND, and logical OR operators are valid for these properties. The syntax follows.

The Calculated Value or Validation expression must be all on one line.

Calculated Value or Validation Statement

```
:   condition
:   expression
```

Condition

```
   :   comparison
   :   condition [AND | OR] condition
```

Comparison

```
   :   [~] [= | < | > | <= | >= | [NOT] [~] LIKE] expression
```

Expression

```
     :   constant
     :   identifier
     :   function
```

Constant

```
        :   number
        :   string (double quoted)
        :   integer
        :   currency
        :   date (double quoted)   “MM/DD/YY”
```

```
          (separator must be "/")
:   time (double quoted)   "HH:MM:SS"
          (separator must be ":")
:   date and time (double quoted)
    "MM/DD/YY HH:MM:SS"   (space required)
:   Boolean
:   phone number (double quoted)
```

identifier

```
:   [field name]
```

For date and time formats in controls or list columns, use the format specified in the Control Panel. In Search Specification or predefined query form, use the business component format.

To reference a field value, you must use [Field Name]. Also, string constants must be enclosed in double quotation marks ("string").

You can use the tilde (~) modifier to make case-insensitive string comparisons in Calculated Value expressions.

NOTE: A field's Validation property cannot be used to make the field required based on another field value.

When a Calculated Value statement references more than one field value, and the fields have different data types, the order of the data types can have an effect on the calculation.

For example, the Quote Item business component has the calculated field Line Total, whose calculated value is `[Item Price] * [Quantity]`. The data type of Item Price is DTYPE_INTEGER; the data types of both Quantity and Line Total is DTYPE_CURRENCY.

If Item Price is `2.25` and Quantity is `5`, Line Total is calculated to be `11.25`. However, if the calculated value of Line Total is changed to `[Quantity] * [Item Price]`, and you use the same values, Line Total is calculated to be `11.00`.

Field Object Data Types

All Field objects have a data type. Single-value fields have a data type value. Multivalue fields inherit the data type from the source field. Many types can convert to other types during calculations. Many operations produce different results with different types. For example, “10” + “10” produces “1010”, while 10 + 10 produces 20.

Siebel applications calculations are “left-centric”; for example, “10” + 10 produces “1010”, while 10 + “10” produces 20. In the first example, the right argument 10 converts itself to a string, but in the second example, the right argument “10” converts itself to a number.

Field objects can have any of the following Siebel data types:

- DTYPE_BOOL
- DTYPE_CURRENCY
- DTYPE_DATE
- DTYPE_DATETIME
- DTYPE_ID
- DTYPE_INTEGER
- DTYPE_NOTE
- DTYPE_NUMBER
- DTYPE_PHONE
- DTYPE_TEXT
- DTYPE_TIME

NOTE: Users cannot query on fields of type DTYPE_NOTE.

Search Syntax

Query By Example

Searching or query by example (QBE) can be performed through the user interface list columns or controls as predefined queries, or specified in the Search Specification property. The syntax is slightly different when done through the user interface but, in all cases, the syntax is simple BNF (Backus-Naur Format).

QBE Statement

```
:   condition
:   expression
```

Condition

```
   :   comparison
   :   NOT condition
   :   condition [AND | OR] condition
```

Comparison

```
   :   expression [~] [= | < | > | <= | >= | [NOT] [~] LIKE]
  expression
```

Expression

```
     :   constant
     :   identifier
     :   function
```

Constant

```
        :   number
        :   string (double quoted)
        :   date (double quoted)   "MM/DD/YY"
                 (separator must be "/")
        :   time (double quoted)   "HH:MM:SS"
                 (separator must be ":")
        :   date and time (double quoted)
           "MM/DD/YY HH:MM:SS"   (space required)
```

Identifier

```
:   [field name]
```

NOTE: For date and time formats in controls and list columns, use the format specified in the Control Panel. In Search Specification or predefined query form, use the business component format.

Search Specification

Assigning a search expression to an object definition's Search Specification property is similar to the predefined query's expression; however, identifying the business component and specifying the reserved word "Search" is not required.

NOTE: The Search Specification expression must be all on one line. If more than one line is used, an "Invalid search specification..." error message appears when you access the involved view.

The syntax would be assigned to the Search Specification property as:

1. "[Close Date] > ""04/15/95"""
2. "[Opportunity] LIKE ""C*"""
3. "[Revenue] > 500000 AND [State] = ""CA"""
4. "[Revenue] > 500000 OR [Revenue] < 10000"
5. "([Revenue] > 500000 AND [State] = ""CA"") OR ([Revenue] > 200000 AND [State] = ""FL"")"
6. "NOT ([State] = ""CA"")"

NOTE: Add the predefined variable DivisionId() to the search specification property of an applet in order to limit the visibility of the applet to employees from the same division as the person logged in:
"DivisionId() = [Division Id]".

In the preceding examples, the fields declared must exist within the designated object definition (like business component or Report) and must adhere to the object type's declaration standards.

When drilling down on a record, if the search specification of the target applet is different from the originating applet, the first record of the destination view will be displayed rather than the drilled-down record.

NOTE: A search done through a Search Specification property is always case-sensitive. You can use the ~ modifier, however, to make the search case-insensitive. For example, you might use `[Last Name] ~LIKE 'g*'` or `[Last Name] ~= 'GRANER'`

Searching and Sorting on Division ID and Division Name

The two functions DivisionId() and DivisionName() are available for search and sort specifications and calculated values but are not available for the scripting languages.

To return the Division Id in eScript in the standard application

- Use the following code:

```
var oEmpl = TheApplication().GetBusObject("Employee");

var bcEmp = oEmpl.GetBusComp("Employee");

bcEmp.ActivateField("Division Id");

bcEmp.ActivateField("Login Id");

bcEmp.SetSearchSpec "Login Id", TheApplication().LoginId();

bcEmp.ExecuteQuery(ForwardOnly);

bcEmp.FirstRecord;

var divId = bcEmp.GetFieldValue("Division Id");
```

Searching Multi-Value Groups with [NOT] EXISTS

You can specify the [NOT] EXISTS operator in a QBE or Search Specification referring to a multi-value group field. A multi-value group field is the user interface mechanism for displaying the child records of a parent record within the parent record's applet. For example, assume the following:

- Opportunities are a separate entity and business component.
- Contacts are a separate entity and business component.
- Both the Opportunity and Contact business components are included in an Opportunity business object.
- There is a many-to-many relationship between opportunities and contacts (that is, opportunities can be worked by one or more contacts, but a contact can work one and only one opportunity).
- A form applet views the Opportunity business component with the following fields: Opportunity Name, Contact First Name, and Contact Last Name.
- The form applet is opportunity-focused. That is, the purpose of the form applet is to display and manage opportunity information (any contact information displayed is specific to the opportunity).

NOTE: When using QBE on multivalue fields (MVF), include only those MVF that are exposed in the originating business component.

Because the form applet is opportunity-focused, the opportunity name is a standard text box control, whereas the contact's first and last names are defined as multi-value group fields. Contact's first and last names are defined using multi-value group fields instead of standard edit controls, because the only way to display multiple contacts for an opportunity in an opportunity-focused applet is through a multi-value group field.

When you enter “Wine Festival” as a search specification in the opportunity name, you are asking the Opportunity business component to return all opportunities that have a name of “Wine Festival.” When you enter “Smith” as a search specification in the contact last name, however, you are asking the Opportunity business component (not the Contact business component) to return all opportunities that have contacts with a last name of “Smith.”

This multivalue query transcends business components and, therefore, requires the [NOT] EXISTS keyword, as shown in the following syntax examples:

Syntax for QBE (placed directly in the last name field in the user interface):

```
EXISTS(Smith)
```

Syntax for a predefined query (Opportunity is the business component):

```
Opportunity.Search = “EXISTS ([Last Name] = ““Smith””)”
```

Syntax for a search specification (placed directly in the Search Specification property in the business component or applet):

```
EXISTS ([Last Name] = 'Smith')
```

Select records based on multiple child and grandchild criteria:

```
EXISTS ([ChildField1] = 'X' AND [ChildField2] = 'Y')
```

```
EXISTS ([GrandchildField1] = 'A' AND [GrandchildField2] = 'B')
```

Searching on Primary Fields

If you have an MVF with a primary ID field specified and the Use Primary Join attribute checked, then querying without EXISTS finds all the records where the primary record in the MVG matches that particular search spec.

If you specify EXISTS, then it will find every record where any of the records in the MVG match the search spec. If you do not specify a primary ID field for the MVG or set the Use Primary Join attribute to unchecked, then the only available query is one that uses EXISTS. In this case, if you specify a query that does not use EXISTS, it will automatically be assumed and inserted as part of the search spec.

The default behavior for querying on multi-value groups is that specifying a value for a MVG or MVF queries for the primary value.

For example, if you query on the Account Team with the value VSILVER (and the MVG has been configured to support a primary), it will return the records where VSLIVER is the primary position on the team.

NOTE: In a view with sales team visibility, do not attempt to constrain the account team by using query by example. Use a view with All visibility. For example, you are logged in as SADMIN and you are on My Accounts view. When you now create a new query where a login name is entered for Account Team (for example, VSILVER) you cannot expect to receive all accounts where SADMIN is on the team and VSILVER is the primary.

Sort Syntax

Sorting can be accomplished by either clicking the Sort Ascending/Descending buttons, modifying the predefined query expression, or assigning a sort expression to an object type that supports a sort expression.

- "Sorting Through Predefined Queries" on page 680
- "Sorting Through the Object Property Sort Specification" on page 680
- "Sorting Through the User Interface" on page 681

Sorting Through Predefined Queries

If you have saved a query, you can modify the expression through the PreDefined Query view and add a sort expression. You can specify one or more fields, with each field further refined as either ascending or descending. The syntax for the predefined query is as follows:

```
'Business Component'.Sort = "[Field] [[(DESC[ENDING])], [Field]
[(DESC[ENDING]],...]"
```

- Business Component is the name of the business component to sort on, if contained in the overall business object.
- Sort is a reserved word that indicates a sort expression follows (as opposed to "Search").
- Field is the name of the field to sort on.

Sorting Through the Object Property Sort Specification

Assigning a sort expression to an object definition's Sort Specification property is similar to the predefined query's expression; however, identifying the business component and specifying the reserved word Sort is not required. The preceding syntax would be assigned to an object definition's Sort Specification property as follows:

1. "[Close Date]"
2. "[Opportunity] (DESCENDING)"

3 "[Revenue]"

4 "[Revenue] (DESCENDING)"

5 "[Revenue] (DESC), [State]"

Sorting Through the User Interface

You can contrast sorting through the user interface with sorting through a predefined query or through the Search Specification property. Sorting through the user interface is available by using the sort buttons, to list applets only, not to form applets.

To specify sort ascending or descending, after retrieving data, the end user selects a list column to sort on by clicking on the list column header and clicking one of the sort buttons.

For details on sorting using the user interface, see *Fundamentals*.

Sorting Versus Searching

ORDER BY in the SQL is generated from the sort specification.

WHERE in the SQL is generated by the search specification.

NOTE: GROUP BY is not supported for business components.

Online Help Development 10

This section provides information about the help implementation in Siebel eBusiness Applications. It also contains information about customizing and migrating online help.

The topics in this section assume that you are familiar with Siebel Tools, help development, HTML authoring, the use of cascading style sheets, and JavaScript.

- "Help Implementation Overview" contains an overview of the implementation of online help in Siebel eBusiness Applications to provide you with the information you need to customize Siebel online help.
- If your Siebel eBusiness Application is an employee application, "Employee Applications" provides all the information you need to implement customized help for your application.
- If your Siebel eBusiness Application is a customer application, "Customer Applications" provides all the information you need to implement customized help for your application.
- If you need to deploy customized online help globally, see "Global Deployment."
- "Help Source Files" contains a detailed list of all the files used in the help systems a description of the cascading style sheet and JavaScript file used in the help systems.

Help Implementation Overview

The topics in this section provide a high-level overview of the implementation of online help in Siebel 7 employee, partner, and customer applications. It also provides some tips about editing the HTML files that make up the help system.

The following topics are included:

Employee Applications

Employee applications are generally used by internal employees of an enterprise. Siebel Call Center and Siebel Sales are examples of employee applications. In these applications, the help system is delivered in HTML format, and the help is context-sensitive at the screen level. When a user accesses help, the application calls the Siebel Web Engine (SWE) GotoPage method, which uses SWE code to display the correct help topic in a separate browser window.

The start page of the employee applications help system is siebstarthelp.htm, shown in Figure 133. For information about using the help system, see "Using the Employee Applications Online Help" on page 685.

"Help Source Files" contains information about the files that make up the employee applications help system.

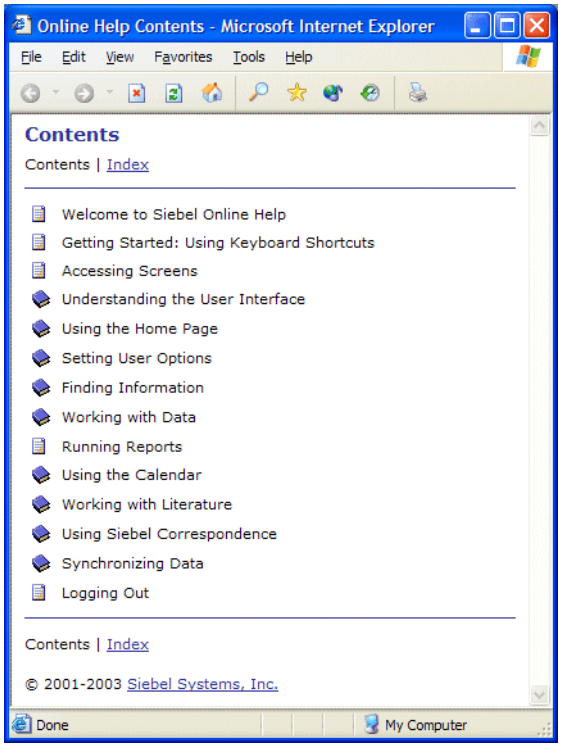


Figure 133. Start Page Employee Applications Help

Using the Employee Applications Online Help

After you have accessed the online help, you can use the Web browser's functionality to navigate in the help system and to print topics. In addition, the online help has an index that lets you find topic by keyword.

To link to help topics you recently visited

- To return to the last topic you viewed, click the Web browser's Back button.
- To view a topic you viewed before clicking the Web browser's Back button, click the Web browser's Forward button.

To print a help topic

- Click the Web browser's Print button.

  The current HTML page is printed to your default printer. The HTML page may consist of more than one topic.

To return to the main contents page

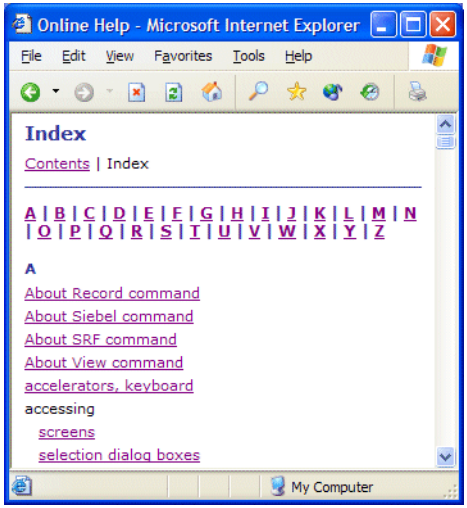
- In the help window, click the Contents hyperlink that appears at the start and end of each HTML page that makes up the help system.

  The start page of the employee applications help system appears.

To search the help by keyword

1. In the help window, click the Index hyperlink.

   The Index topic, shown below, appears.



2. Click a letter hyperlink at the start of the page to go to entries for that letter.
3. Click the hyperlink for the keyword.

   The help topic appears.

Customer Applications

Customer applications are generally used by external partners, customers, and prospects of an enterprise. Siebel eSales and Siebel eService are examples of customer applications. In these applications, the help system is delivered in HTML format, and the applications are configured to show the start page of the help system in a separate browser window whenever a user accesses help. This is done by using the SWE GotoURL method.

The start page of the customer applications help system is siebcomgeneric.htm, shown in Figure 134. For information about using the help system, see "Using the Customer Applications Online Help" on page 688.

"Help Source Files" contains information about the files that make up the customer applications help system.

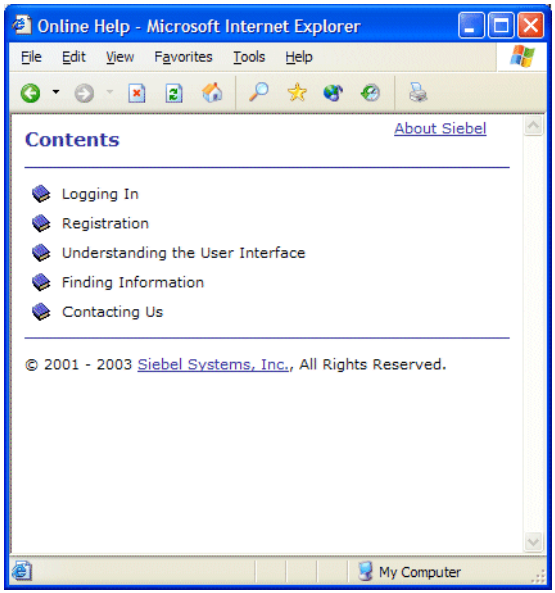


Figure 134. Start Page Customer Applications Help

Using the Customer Applications Online Help

Once you have accessed the online help, you can use the Web browser's functionality to navigate in the help system and to print topics.

To link to help topics you recently visited

- To return to the last topic you viewed, click the Web browser's Back button.
- To view a topic you viewed before clicking the Web browser's Back button, click the Web browser's Forward button.

To print a help topic

- Click the Web browser's Print button.

  The current HTML page is printed to your default printer. The HTML page may consist of more than one topic.

To return to the main contents page

- In the help window, click the Contents hyperlink that appears at the start and end of each HTML page that makes up the help system.

  The start page of the customer applications help system appears.

About Editing HTML Files

Siebel applications help files reside in a folder called "help." When you have determined which HTML files you need to change, it is recommended that you copy those files to your local machine to make changes.

Testing and Distributing Changes

After you make your changes, you should verify your changes before distribution.

To test your changes

1. Use a standard HTML authoring tool to verify links.

   NOTE: You should verify the links in the changed HTML file, and also make sure no links were broken in the existing HTML files.

2. In a Web browser, open the HTML file you changed and review the content.

When you have completed testing, you must distribute the updated files to the appropriate Siebel Servers, Siebel Mobile Web Clients, or Siebel Dedicated Web Clients.

Employee Applications

The topics in this section describe the details of the help implementation in Siebel 7 employee applications. It explains the location of the help files and provides information about using Siebel Tools to change help calls. In addition, different options for customizing the online help to suit your requirements are discussed. If you customized the online help in earlier versions of your Siebel application, you can also find instructions for migrating your customized online help.

The following topics are included:

NOTE: The help implementation is identical for employee and partner applications. Therefore, the information contained in the topics above applies to both employee and partner applications.

Location of Employee Application Help Files

The location of the help files is determined by the location where you installed your Siebel applications and by the type of Siebel client:

Siebel Web Client

Siebel Web Client runs in a standard browser from the client personal computer and does not require any additional persistent software installed on the client. The browser connects through a Web server to the Siebel eBusiness Application server, which executes business logic and accesses data from the Siebel Database.

In this implementation, help files are installed in the following location on the server:

< *install dir* > \public\ < *install language* > \help, where

- *install dir* is the directory where you installed the Siebel Web Server Extensions
- *install language* is the language you selected during installation

During the installation process, your Web server is configured so that
< *install dir* > \public\ < *install language* > \
becomes the root directory for the URL
http:// < *hostname* > / < *Siebel application name* >

When a Help Identifier property in a Screen object maps to a URL in the Help Id object, that URL is relative to http:// < *hostname* > / < *Siebel application name* > .

Example

If you are running Siebel Call Center on the server siebsrvr, when you request the help for the Accounts screen, the page that will appear is http://siebsrvr/callcenter/help/siebaccounts.htm. This means that you can put the help files in any directory that can be referenced from http:// < *hostname* > / < *Siebel_application_name* > .

Assume that in < *install dir* > \public\ < *install language* > \, you create a directory called customizedhelp and you create a new help topic file for the Accounts screen, called accountshelp.htm. After that, you configure your Web server so that this directory is exposed to a Siebel application user. In this case, you would create Help Id objects that point to the URL customizedhelp; the Help Id object properties would be:

Property	Value
Name	ID_SCREEN_ACCOUNTS
Project	Repository Help Id
Type	Screen
HTML Help URL	customizedhelp/accountshelp.htm

Siebel Dedicated Web Client and Siebel Mobile Web Client

Siebel Dedicated Web Client is a Microsoft Windows client delivered through a Web browser that provides direct connectivity to a database server. It requires software to be installed on the client machine, but does not require a local database, Web server, or Siebel eBusiness Application server for serving up interactive user sessions. Siebel Server is still required for functionality like Assignment Manager and Replication.

Siebel Mobile Web Client is a portable Microsoft Windows client delivered through a Web browser that is designed for local data access and does not need to be connected to a server. Siebel Mobile Web Client meets the needs of field professionals who do not have continuous access to a network. Siebel Mobile Web Client uses a local database on each mobile machine. Periodically, the client must access the Siebel Remote Server through a dial-up, WAN, or LAN connection to synchronize data changes with the Siebel Database on the database server and Siebel File System. This client requires installation of Siebel software on the user's personal computer.

The software installed on the user's machine for the Siebel Dedicated Web Client and Siebel Mobile Web Client is identical—the only difference is the type of connectivity provided.

In these implementations, the location of the help files is determined by the installation directory on the client:

< *install path* > \public\ < *install language* > \help, where

- *install path* is the complete path to the location where you installed the Siebel application
- *install language* is the language you selected during installation

For example, D:\sea\webclient\public\enu\help.

During the installation process, your local Web server is configured so that
< *install path* > \public\ < *install language* > \
becomes the root directory for the URL http://localhost/.

When a Help Identifier property in a Screen object maps to a URL in the Help Id object, that URL is relative to http://localhost.

Example

If you are running Siebel Call Center on your local machine, when you request the help for the Accounts screen, the page that will appear is http://localhost/help/siebaccounts.htm. This means that you can put the help files in any directory that can be referenced from http://localhost/.

Assume that in <*install path*>\public\<*install language*>\, you created a directory called customizedhelp and you created a new help topic file for the Accounts screen, called accountshelp.htm.

In this case, you would create Help Id objects that point to the URL customizedhelp; the Help Id object properties would be:

Property	Value
Name	ID_SCREEN_ACCOUNTS
Project	Repository Help Id
Type	Screen
HTML Help URL	customizedhelp/accountshelp.htm

Online Help and Siebel Tools

In Siebel Tools, Screen, View, and Help Id objects are used to establish the link between a screen or a view and a help topic file.

NOTE: Standard Siebel eBusiness Applications do not contain view-level help topic references; they are all at the screen level. See "Implementing Help for a View" on page 699 for instructions about adding view-level help references.

The following topics are included:

Screen and View Objects

Each Screen and View object has a Help Identifier property that is used to establish the link with the Help Id object.

The format of the Help Identifier property is ID_*type_objdefname*, where:

- *type* is SCREEN or VIEW
- *objdefname* identifies the screen or view

Help Id Objects

Each Help Id object has a HTML Help URL property that is used to identify the HTML file that contains the help topics for the screen or the view by mapping the Help Identifier to a specific URL. You can use the same value for the HTML Help URL property for different Help Id objects.

The format of the HTML Help URL property is *help*/*helptopics*.htm, where:

- *help* is the server folder where the HTML topic files reside
- *helptopics*.htm is the HTML file that appears when a user invokes help

See "Location of Employee Application Help Files" on page 690 for additional information about where the help files are installed.

Help Properties of Screens and Views

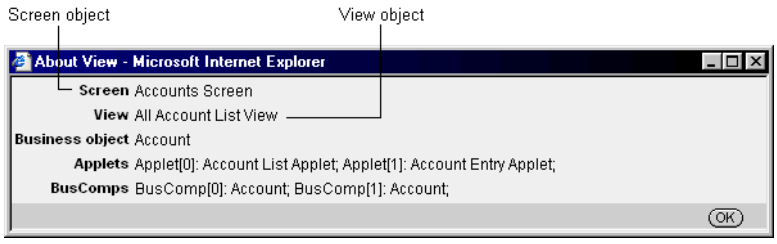
To determine which HTML file contains the help topics for a screen or a view you must: use your Siebel application to determine the Screen or View object used in the screen or view, then use Siebel Tools to find the Help Identifier property of the Screen or View object and then find the corresponding HTML Help URL property in the Help Id object.

NOTE: Some screens are associated with generic help topics. In this case, you must use a different filename if you want to customize the help. See "Customizing Help for Screens with Generic Help Topic Files" on page 702 for instructions.

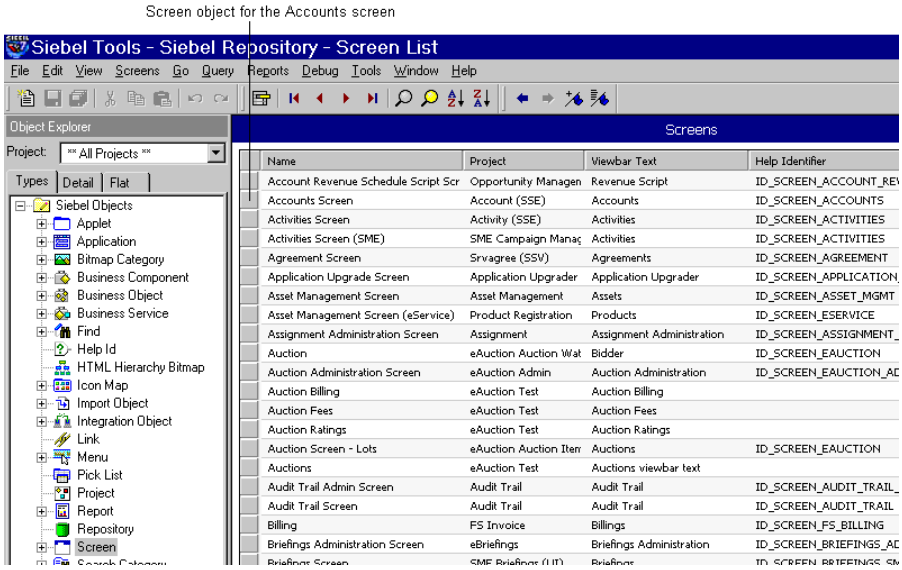
Example

To determine which HTML file contains the help topics for the Accounts screen, you must find which Screen object is used in the Accounts screen.

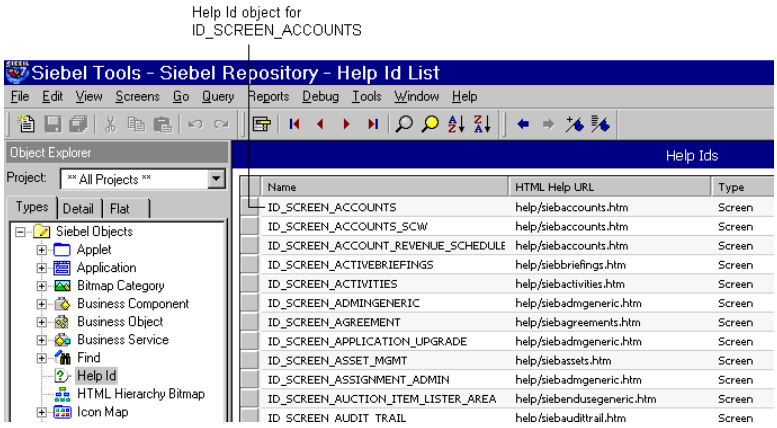
To do this, in your Siebel eBusiness Application, navigate to the Accounts screen, then choose Help > About View to access the About View dialog box, shown below.



In Siebel Tools, find the Accounts screen object and find the value of its Help Identifier property (ID_SCREEN_ACCOUNTS) in the Object List Editor window, shown below.



When you know the value of the Help Identifier property of the Screen object (ID_SCREEN_ACCOUNTS), you can find the value of the Help Id object's HTML Help URL property (help/siebaccounts.htm) in the Object List Editor window, as shown below.



Default Help Topic

The start page of the employee applications help system (siebstarthelp.htm) is defined as the default help topic for employee applications. When a user accesses help from a screen or view for which the Help Identifier property is not defined, siebstarthelp.htm appears in a separate browser window.

To change the default help topic

1. In Siebel Tools, in the Object Explorer (Types tab) find and select the Web Page object type.
2. In the Object List Editor, find and select the CC Help Page object definition.
3. In the Object Explorer (Types tab), expand the Web Page object type and select the Web Page Item object type.
4. In the Object List Editor, modify the Caption property of the Online Help object definition to point to your default help topic.

   For example, change the Caption property from `help/siebstarthelp.htm` to `help/index.htm`.

5 Recompile the repository file.

6 Test your changes and distribute the repository file and your new default topic file to the appropriate Siebel Servers, Siebel Mobile Web Clients, and Siebel Dedicated Web Clients.

For example, for Siebel Web Client implementations, copy index.htm to the correct Siebel Server location (<*install dir*>\public\<*install language*>\help).

For more information, see "Testing and Distributing Changes" on page 688.

Implementing Help for a Screen

If you add a custom screen in your implementation of Siebel eBusiness Applications, you can create help for that screen. To do this, you must use Siebel Tools to define the Screen object with a Help Identifier property, and then add a Help Id object with a HTML Help URL property for the screen. The documentation team can develop the content using the name of the file that will be distributed to the appropriate Siebel Servers, Siebel Mobile Web Clients, and Siebel Dedicated Web Clients.

To add help for a new screen

1 In Siebel Tools, if the Screen object does not exist, create it.

For example, create the My New Screen Screen object.

See *Siebel Tools Reference* for instructions.

2 In the Screen object, define the Help Identifier property.

For example, ID_SCREEN_MYNEWSCREEN.

3 Create a new Help Id object with the following properties:

Property	Value	Example
Name	Value of the Help Identifier property of the screen	ID_SCREEN_MYNEWSCREEN
Project	Repository Help Id	Repository Help Id

Property	Value	Example
Type	Screen	Screen
HTML Help URL	Name of the file that will contain the help content	help/mynewscreen.htm

4 Recompile the repository file.

NOTE: You must compile the Repository Help Id project *and* the Screen object. In our example, you must compile the My New Screen Screen object and the Repository Help Id project.

5 Create a new HTML file with help content for the screen.

6 Save the HTML file using the name defined in the HTML HELP URL property of the Help Id object associated with the screen.

For example, mynewscreen.htm.

7 Test your changes and distribute the repository file and the new HTML file to the appropriate Siebel Servers, Siebel Mobile Web Clients, and Siebel Dedicated Web Clients.

For example, for Siebel Web Client implementations, copy mynewscreen.htm to the correct Siebel Server location (<*install dir*>\public\<*install language*>\help).

For more information, see "Testing and Distributing Changes" on page 688.

Implementing Help for a View

To add help for a view, you must use Siebel Tools to define the View object with a Help Identifier property, and then add a Help Id object with a HTML Help URL property for the view. At the same time, the documentation team can develop the content, and they can use the name of the file that will be distributed to the appropriate Siebel Servers, Siebel Mobile Web Clients, and Siebel Dedicated Web Clients.

To add help for a view

1. In Siebel Tools, if the View object does not exist, create it.

   See *Siebel Tools Reference* for instructions.

2. In the View object, define the Help Identifier property.

   For example, for the Opportunity Detail - Contacts View, use ID_VIEW_OPPORTUNITY_DETAIL_CONTACTS as the Help Identifier.

   NOTE: If you leave the Help Identifier property blank for the View object, the Screen-specific help will be used as the default help topic for the view.

3. Create a new Help Id object with the following properties:

Property	Value	Example
Name	Value of the Help Identifier property of the view	ID_VIEW_OPPORTUNITY_DETAIL_CONTACTS
Project	Repository Help Id	Repository Help Id
Type	View	View
HTML Help URL	Name of the file that contains the help topics	help/siebopportunities_detailcontacts.htm

4. Recompile the repository file.

   NOTE: You must compile the Repository Help Id project *and* the View object. In our example, you must compile the Opportunity Detail - Contacts View object and the Repository Help Id project.

5. Create a new HTML file with the help content for the view.

6. Save the HTML file using the name defined in the HTML HELP URL property of the Help Id object associated with the view.

   For example, siebopportunities_detailcontacts.htm.

7 Test your changes and distribute the repository file and the new HTML file to the appropriate Siebel Servers, Siebel Mobile Web Clients, and Siebel Dedicated Web Clients.

For example, for Siebel Web Client implementations, copy siebopportunities_detailcontacts.htm to the correct Siebel Server location (<*install dir*>\public\<*install language*>\help).

For more information, see "Testing and Distributing Changes" on page 688.

Changing the Keyboard Shortcut for Accessing Help

Since most users are accustomed to accessing help by pressing the F1 key, you may want to map F1 to display online help in your Siebel application. To do this, you must add an accelerator in Siebel Tools.

To change the keyboard shortcut for accessing help

1 In Siebel Tools, select the Command object type in the Object Explorer.

The Commands list appears in the Object List Editor.

2 In the Commands list in the Object List Editor, find the Contents Help (SWE) command object and select the row.

3 In the Object Explorer, click Accelerator.

The Accelerators list appears below the Commands list in the Object List Editor.

4 Change the two Accelerator objects as shown below:

Name	Display Name	Key Sequence
1	F1	F1
2	F1	F1

5 Recompile the repository file.

6 Distribute the repository file to the appropriate Siebel Servers, Siebel Mobile Web Clients, and Siebel Dedicated Web Clients.

Help Menu Items

If you create a new application using Siebel Tools, you can add options to the Help menu in your application. For example, you may want to add an option that allows the user to access the *Bookshelf* from within the application.

The Help application menu option is configured using the Command and Menu objects.

The Command object created for online help is Contents Help (SWE).

See *Siebel Tools Reference* for instructions about using the Command and Menu objects.

Customizing and Adding Help

You can customize help content included for a screen to suit your Siebel implementation and requirements. Standard Siebel eBusiness Applications do not include help files for views. If you want to add help at the view level, you must implement help for the view and then create a new HTML file with the help content for the view. See "Implementing Help for a View" on page 699 for instructions.

The following topics are included:

Customizing Help for Screens with Generic Help Topic Files

Some screens are associated with generic help topics, which means that the HTML Help URL property of the Help Id object for a screen is one of the following:

- help/siebadmgeneric.htm for administrative screens
- help/siebendusegeneric.htm for end user screens

See "Help Properties of Screens and Views" on page 695 for information about finding the HTML Help URL property.

To customize help for screens with generic help content

1. Open the HTML file referenced in the HTML Help URL property (siebendusegeneric.htm or siebadmgeneric.htm).
2. If the following < META > tag appears at the top of the file, remove it.

   ```
   <META HTTP-EQUIV="REFRESH" CONTENT="1; URL=siebwelcome.htm">
   ```

3. Save the file with a different name.
4. In Siebel Tools, update the HTML Help URL property of the Help Id object to reflect the new filename.
5. Follow the instructions in "Customizing Help Content" on page 703.

Customizing Help Content

Since the help consists of HTML pages, you can use any HTML editor to change the content of a help topic.

To customize help content for a screen

1. Find the HTML Help URL property associated with the screen.

   See "Help Properties of Screens and Views" on page 695 for instructions.

   For example, the HTML Help URL property for the Opportunities screen is help/siebopportunities.htm.

   NOTE: If the HTML Help URL property for the screen is siebendusegeneric.htm, siebadmgeneric.htm, or siebstarthelp.htm, use the instructions in "Customizing Help for Screens with Generic Help Topic Files" on page 702 instead.

2 Open the HTML file, make your changes, and save the file.

For example, open siebopportunities.htm, make your changes, and save the file.

NOTE: If a redirect tag exists in the file, be sure to remove the tag. The following is an example of a redirect tag:
`<META HTTP-EQUIV="REFRESH" CONTENT="1; URL=siebwelcome.htm">`

3 Test your changes and distribute the updated HTML file to the appropriate Siebel Servers, Siebel Mobile Web Clients, and Siebel Dedicated Web Clients.

For example, for Siebel Web Client implementations, copy siebopportunities.htm to the correct Siebel Server location (<*install dir*>\public\<*install language*>\help).

For more information, see "Testing and Distributing Changes" on page 688.

Adding Help for a Screen

If you added a custom screen to your implementation of Siebel eBusiness Applications, you can add help for that screen. See "Implementing Help for a Screen" on page 698 for instructions.

Adding Help for a View

To add help for a view, you must name the HTML topic file using the HTML Help URL property of the view. See "Implementing Help for a View" on page 699 for instructions.

Customizing the Help Index

The index topic in the help system allows the user to find topics by keyword (see "To search the help by keyword" on page 686). The index topic is a set of links to topics based on keywords. When you customize the content of the help system, you may want to update the entries in the index to reflect the new content. Another option is to remove access to the index topic entirely.

To customize the help index

1 Open siebindex.htm, make your changes, and save the file.

2 Test your changes and distribute the updated siebindex.htm to the appropriate Siebel Servers, Siebel Mobile Web Clients, and Siebel Dedicated Web Clients.

For example, for Siebel Web Client implementations, copy the updated siebindex.htm to the correct Siebel Server location (<*install dir*>\public\<*install language*>\help).

For more information, see "Testing and Distributing Changes" on page 688.

To remove access to the index topic

1 Open one of the HTML files, except siebindex.htm, that is part of the help system.

2 Delete the statements that reference siebindex.htm and save your changes.

3 Repeat Step 1 and Step 2 for each HTML file in the help system, except siebindex.htm.

4 Test your changes and distribute all the updated HTML files to the appropriate Siebel Servers, Siebel Mobile Web Clients, and Siebel Dedicated Web Clients.

For example, for Siebel Web Client implementations, copy all the updated HTML files to the correct Siebel Server location (<*install dir*>\public\<*install language*>\help).

For more information, see "Testing and Distributing Changes" on page 688.

Migrating Help

If you customized the online help in Siebel 98 (Version 4), Siebel 99 (Version 5), or Siebel 2000 (Version 6) and the customization was mostly related to task topics, it may be more effective to rewrite the content because the navigation in Siebel eBusiness Applications has changed considerably.

The following topics are included:

- "Converting Content to HTML Format Using Siebel File Names" on page 708
- "Converting Content to HTML Format Using Custom File Names" on page 709

Help Migration Options

There are several ways to migrate customized help content to HTML format:

- Update the Siebel HTML topic file with information from your customized topic file from a previous release.

  See "Updating Siebel Topic Files with Custom Content" on page 707.

- Convert existing content to HTML using the filename found in Siebel Tools.

  See "Converting Content to HTML Format Using Siebel File Names" on page 708.

- Convert existing content to HTML using a filename of your choice, and update the HTML Help URL property.

  See "Converting Content to HTML Format Using Custom File Names" on page 709.

If all users will be running on a MS Windows platform, you can use your current compiled Microsoft Windows help system as the help system for your Siebel application. See "Using WinHelp" on page 711 for more information.

NOTE: In this section, *rich text editor* refers to a text editor that supports hidden text and footnotes, such as Microsoft Word.

Sample Scenario

For clarity purposes, the procedures in this section are illustrated by an example that assumes that you customized the help content for the Accounts screen to suit your implementation. In earlier versions of Siebel applications, you would have made these changes in the sa_acct.rtf source topic file.

Updating Siebel Topic Files with Custom Content

You can update the online help with customized information contained in your help system from a previous Siebel release using the Siebel topic files included with your Siebel eBusiness Applications.

Advantages

- Maintains formatting, layout, and navigation elements of the original, Siebel-delivered online help.
- Links to the cascading style sheet remain in place.

Disadvantages

- None

To update the Siebel HTML file with information from a customized .rtf file

1. Find the help properties of the screen.

   See "Help Properties of Screens and Views" on page 695 for instructions.

   In the sample scenario, the HTML Help URL property associated with the Accounts screen is help/siebaccounts.htm.

2. Open the HTML file referenced in the HTML Help URL property.

   In the sample scenario, open siebaccounts.htm.

   NOTE: If a redirect tag exists in the file, be sure to remove the tag. The following is an example of a redirect tag:

   ```
   <META HTTP-EQUIV="REFRESH" CONTENT="1; URL=siebwelcome.htm">
   ```

3. Open the source .rtf file (used to create the help content in the previous version) in a rich text editor.

   In the sample scenario, open sa_acct.rtf.

4. Use copy and paste functionality to copy content from the .rtf file to the HTML file.

   In the sample scenario, copy content from sa_acct.rtf to siebaccounts.htm.

5 Apply the appropriate HTML tags to format content and save the HTML file.

See "Help Source Files" for a description of the cascading style sheet used in the help.

In the sample scenario, save siebaccounts.htm.

6 Test your changes and distribute the updated HTML file to the appropriate Siebel Servers, Siebel Mobile Web Clients, and Siebel Dedicated Web Clients.

In the sample scenario, for Siebel Web Client implementations, replace <*install dir*> \public\ <*install language*> \help\siebaccounts.htm with your version of siebaccounts.htm created from the .rtf file.

For more information, see "Testing and Distributing Changes" on page 688.

Converting Content to HTML Format Using Siebel File Names

If you customized the help content in previous versions of your Siebel application, you can use the method described in this section to convert the customized help content to HTML format. Use your source topic file (rich text format, .rtf) as a starting point and follow the instructions.

Advantages

- The new help content exactly matches the customized content of the previous release.

Disadvantages

- Creates a risk of unexpected formatting results if the person customizing the help is not familiar with HTML.
- Creates risk of losing navigation elements in the help system (to table of contents and index topics) if the correct HTML code is not inserted in the new HTML file.

To convert a customized .rtf file to HTML using the Siebel filename

1 Find the help properties of the screen.

See "Help Properties of Screens and Views" on page 695 for instructions.

In the sample scenario, the HTML Help URL property associated with the Accounts screen is help/siebaccounts.htm. The Siebel filename for the topic file is "siebaccounts.htm."

2 Open the source .rtf file (used to create the help content in the previous version) in a rich text editor.

   In the sample scenario, open sa_acct.rtf.

3 Save the file in HTML format, using the appropriate Siebel filename.

   In the sample scenario, save sa_acct.rtf as siebaccounts.htm.

4 Open the newly created HTML file.

5 In the HTML file, add a reference to the Siebel help cascading style sheet (siebhelp.css) and add the necessary blocks of code to implement the navigation hyperlinks.

   NOTE: You can copy this information from one of the Siebel-delivered HTML files.

6 Clean up the HTML code to use styles defined in the style sheet and save the HTML file.

7 Test your changes and distribute the updated HTML file to the appropriate Siebel Servers, Siebel Mobile Web Clients, and Siebel Dedicated Web Clients.

   In the sample scenario, for Siebel Web Client implementations, replace <*install dir*> \public\ <*install language*> \help\siebaccounts.htm with your version of siebaccounts.htm.

   For more information, see "Testing and Distributing Changes" on page 688.

Converting Content to HTML Format Using Custom File Names

You can use existing content in .rtf format and convert it to HTML using your own file naming convention.

Advantages

- Requires very little work from your help developer or technical writer.

Disadvantages

- Siebel Tools developers have to update the help properties for each topic file that does not use a Siebel name.

- Increases risk of errors because the names of files may be entered incorrectly in Siebel Tools.

To convert existing content to HTML using a filename of your choice

1. Open the source .rtf file (used to create the help content in the previous version) in a rich text editor.

   In the sample scenario, open sa_acct.rtf.

2. Save the .rtf file in HTML format, with a name you choose.

   In the sample scenario, save sa_acct.rtf as sa_acct.htm.

3. Test your changes by opening the HTML file in a Web browser.

4. In Siebel Tools, find the help properties of the screen.

   See "Help Properties of Screens and Views" on page 695 for instructions.

5. Update the HTML Help URL property of the Help Id object to reflect the correct filename.

   In the sample scenario, update the HTML Help URL property for the ID_SCREEN_ACCOUNTS Help ID object to be help/sa_acct.htm.

6. Do one of the following:

   - If you only changed HTML Help URL properties, recompile the Repository Help Id project to create an SRF and implement the changes.
   - If you made other changes, recompile all affected projects to create an SRF and implement the changes.

7. Test your changes and distribute the repository file and the new HTML file to the appropriate Siebel Servers, Siebel Mobile Web Clients, and Siebel Dedicated Web Clients.

   In the sample scenario, for Siebel Web Client implementations, copy sa_acct.htm to the correct Siebel Server location (<*install dir*>\public\<*install language*>\help).

   For more information, see "Testing and Distributing Changes" on page 688.

Using WinHelp

If all users will be running on a Microsoft Windows platform, you can use your current compiled Microsoft Windows help system (WinHelp) as the help system for your Siebel eBusiness Applications.

It is important to note that this solution is *not recommended* and that there are several drawbacks to this implementation:

- Each time a user invokes help, the Web browser's File Download dialog box will appear, and the user must respond to access the help. The only way to avoid this is for your administrator to change security settings.
- When a user invokes help at a screen, the default topic in the WinHelp file appears in the help window, not the context-sensitive topic associated with the screen.

  In WinHelp, you specify the default topic for the help file in the [OPTIONS] section of the help project file (.hpj). If you do not specify a default topic, WinHelp uses the first topic of the first file listed in the help project (.hpj) file.
- From the help window, the user can access the Index, but the table of contents (usually available through the Contents tab of the Help Topics window) is not available. Microsoft is aware of this problem, but since WinHelp is no longer the Microsoft method of choice for help delivery, Microsoft will not fix this defect.

NOTE: The following procedure assumes that you want to use siebhelp.hlp (a WinHelp file) as the help system for your Siebel eBusiness Applications.

To use a compiled WinHelp file

1. In Siebel Tools, update the HTML Help URL property for all Help Id objects to reflect the correct filename.

   In this example, update the HTML Help URL property for all Help Id objects to be help/siebhelp.hlp.

2. Do one of the following:

   - If you only changed HTML Help URL properties, recompile the Repository Help Id project to create an SRF and implement the changes.

- If you made other changes, recompile all affected projects to create an SRF and implement the changes.

3. Distribute the repository file and the help file (Windows compiled help file) to the appropriate Siebel Servers, Siebel Mobile Web Clients, and Siebel Dedicated Web Clients.

   For example, for Siebel Web Client implementations, distribute the repository file and copy siebhelp.hlp to the correct Siebel Server location (<*install dir*>\public\<*install language*>\help).

Customer Applications

The topics in this section describe the details of the help implementation in Siebel 7 customer applications. It explains the location of the help files and provides information about using Siebel Tools to verify the calls to the online help. In addition, instructions are included for customizing the help content and migrating your existing customized online help from earlier versions.

The following topics are included:

Location of Customer Application Help Files

The location of the help files is determined by the location where you installed your Siebel applications.

Help files are installed in the following location on the server:

<*install dir*>\public\<*install language*>\help, where

- *install dir* is the directory where you installed the Siebel Web Server Extensions
- *install language* is the language you selected during installation

During the installation process, your Web server is configured so that
<*install dir*>\public\<*install language*>\
becomes the root directory for the URL
http://<*hostname*>/<*Siebel application name*>

When the Value property of the Url Web Page Item Parameter maps to a URL, that URL is relative to http:// < *hostname* > / < *Siebel application name* > .

For example, if you are running Siebel eSales on the server siebsrvr, when you request help, the page that will appear is http://siebsrvr/esales/help/comgeneric.htm. This means that you can put the help files in any directory that can be referenced from http:// < *hostname* > / < *Siebel_application_name* > .

For example, assume that in < *install dir* > \public\ < *install language* > \, you create a directory called customizedhelp and you create a start page for your help system, called myonlinehelp.htm. After that, you configure your Web server so that this directory is exposed to your application users. In this case, you would create a Web Page Item Parameter object definition that points to the URL customizedhelp; the Web Page Item Parameter object properties would be:

Property	Value
Name	Url
Value	customizedhelp/myonlinehelp.htm

Online Help and Siebel Tools

In Siebel Tools, a container Web Page object is used to establish the link between an application and its help start page, as shown in Figure 135.

Each Application object has a Container Web Page property which represents a Web Page object. Each Web Page object contains Web Page Item objects. The HelpButton Web Page Item has one parameter that identifies the start page of the help system for the application.

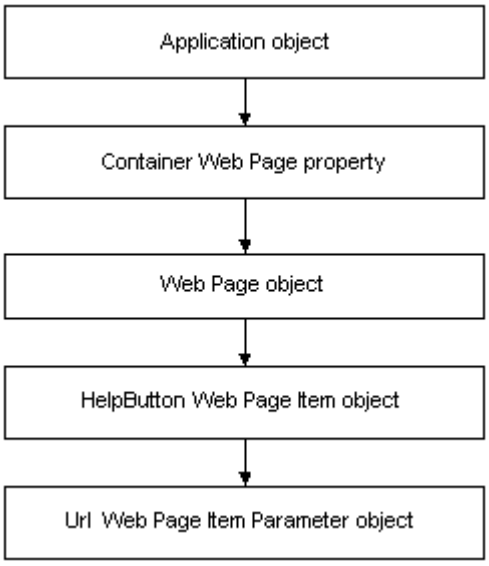


Figure 135. Establishing the Link Between an Application and Help

Changing Help Links

If you want to use a different start page for the help, you can change the help link for the application.

To change a help link

1. In Siebel Tools, in the Object Explorer (Types tab) select the Application object type.
2. In the Object List Editor, find the Container Web Page that the application uses.
3. In the Object Explorer (Types tab), select the Web Page object type.

4 In the Object List Editor, find and select the Web Page object definition that uses the Container Web Page.

5 In the Object Explorer (Types tab), expand the Web Page object type and select the Web Page Item object type.

6 In the Object List Editor, select the HelpButton Web Page Item object definition, and in the Object Explorer (Types tab), expand the Web Page Item object to expose and select the Web Page Item Parameter object type.

7 In the Url Web Page Item Parameter object definition, change the Value property to reflect the name of the HTML file you want to use as a start page.

   For example, help/index.htm.

8 Recompile to create an updated SRF.

9 Optionally, in the HTML file you want to use as a start page, add a reference to the Siebel help cascading style sheet (siebhelp.css) and add the necessary code to implement the navigation buttons.

   NOTE: You can use help/siebcomgeneric.htm as an example.

10 Distribute the repository file and the new HTML file to the appropriate Siebel Server.

Adding Help Links for New Applications

If you create a new application using Siebel Tools, you can add help links to your application.

To add a help link

1 In Siebel Tools, in the Object Explorer (Types tab) select the Application object type.

2 In the Object List Editor, find the value of the Container Web Page property for the Application object.

   For example, the Container Web Page property of the Siebel eSales Application object is CC Container Page (eSales). Container Web pages map to the Web Page object in Siebel Tools.

3 In the Object Explorer (Types tab), select the Web Page object type.

4 In the Object List Editor, find and select the Web Page object definition that uses the Container Web Page.

5 In the Object Explorer (Types tab), expand the Web Page object type to expose the Web Page Item object type.

6 In the Object List Editor, create a new Web Page Item object definition with the following properties:

Property	Value
Name	HelpButton
Type	Link
Caption	Help
Method Invoked	GotoURL
Item Identifier	A number between 11 and 19 that is not used by another Web Page Item object definition.
HTML Attribute	target = "_blank"

7 In the Object List Editor, select the Web Page Item object definition you created, and in the Object Explorer (Types tab), expand the Web Page Item object to expose the Web Page Item Parameter object type.

8 Create a Web Page Item Parameter object definition with the following properties:

Property	Value
Name	Url
Value	help/*siebcomgeneric*.htm siebcomgeneric.htm is the default start page for the help system. You can change this parameter to point to a different file, for example, index.htm.

9 Recompile to create an updated SRF.

10 Distribute the repository file to the appropriate Siebel Server.

11 If you are using a different start page, test your changes and upload the new HTML file to the Siebel Server.

For more information, see "Testing and Distributing Changes" on page 688.

Customizing Help Content

Since the help consists of HTML pages, you can use any HTML editor to change the content of a help topic.

To customize help content

1 Find the HTML file you want to change.

For a list of source files, see "Help Source Files."

2 Open the HTML file, make your changes, and save the file.

NOTE: If a redirect tag exists in the file, be sure to remove the tag. The following is an example of a redirect tag:

```
<META HTTP-EQUIV="REFRESH" CONTENT="1; URL=siebcomgeneric.htm">
```

3 Test your changes and update the HTML file on the Siebel Server.

For more information, see "Testing and Distributing Changes" on page 688.

Adding Help Content

You can add help content in the existing HTML files, as explained in "Customizing Help Content" on page 718, or you can add new HTML files to complement the existing content. If you add new HTML files, you must add links to these new files from the existing files to ensure that your users will be able to access the content.

To add help content by adding HTML files

1. Create the new HTML file with your content.

   TIP: Start with one of the files included in the Siebel help system to avoid losing style specifications.

2. Test your changes.

   For more information, see "Testing and Distributing Changes" on page 688.

3. Upload the HTML file to the Siebel Server.

   For the exact location of the help files, see "Location of Customer Application Help Files" on page 713.

Migrating Help

You can update the online help with customized information contained in your help system from a previous Siebel release by using the Siebel topic files included with your Siebel eBusiness Applications.

To update the Siebel HTML file with information from a customized HTML file

1. Find the name of the HTML file to update and open the file.

   See "Customer Applications Files" on page 726 for a complete list of files.

2. Open the HTML file used to create help content in the previous version.

   If you used the Siebel filename in earlier releases, the filename would be Siebel_eBusiness_Help.htm.

3. Use copy and paste functionality to update the Siebel HTML file.

4 Apply appropriate HTML tags to format content and save the HTML file.

NOTE: If a redirect tag exists in the file, be sure to remove the tag. The following is an example of a redirect tag:

```
<META HTTP-EQUIV="REFRESH" CONTENT="1; URL=siebwelcome.htm">
```

5 Test your changes and update the HTML file on the Siebel Server.

For more information, see "Testing and Distributing Changes" on page 688.

Global Deployment

The topics in this section provide high-level information about deploying online help in different languages.

The following topics are included:

■ "Language Folders" on page 721

■ "Localizing Online Help" on page 722

For additional information about global deployment, see *Global Deployment Guide*.

Language Folders

Your Siebel application comes with localized online help. Localized online help files are located in the language-specific folders on either the server or the Mobile or Dedicated Web client.

Help files are installed in the following location on the server:

<*install dir*>\public\<*install language*>\help, where

- *install dir* is the directory where you installed the Siebel Web Server Extensions
- *install language* is the language you selected during installation

For details about the location of the help files for different Siebel applications, see "Location of Employee Application Help Files" on page 690 and "Location of Customer Application Help Files" on page 713.

Localizing Online Help

If you are deploying your Siebel application in a language not available from Siebel Systems, Inc., and you want to deploy online help in that language, you must localize the online help. Figure 136 shows the typical steps involved in help localization.

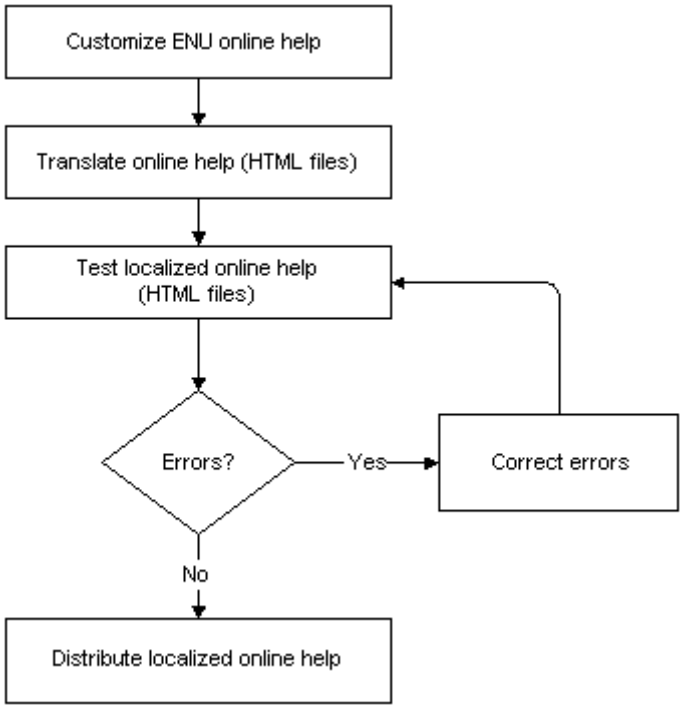


Figure 136. Help Localization Steps

To localize online help

1. (Optional) Customize the ENU (American English) help to suit your implementation.
2. Translate the HTML source files (that make up the help system) by modifying the flat files directly.

3 Test and distribute the localized online help to the appropriate Siebel Servers, Siebel Mobile Web Clients, and Siebel Dedicated Web Clients.

Help Source Files

The following topics list the source files that make up the help system for the employee, partner, and customer applications.

Siebel Systems, Inc. provides the text of the application online help in a format that you can edit to reflect your unique implementation. You can use the content described in this document as the foundation for the help system you distribute to users.

Note that information about your Siebel application (including the online help) is covered by your confidentiality agreement with Siebel Systems and thus cannot be shared with individuals who do not already have access to your Siebel implementation.

Employee Applications Files

The help system for employee applications consists of HTML files (listed under HTML Files), image files, and a cascading style sheet (siebhelp.css, described in "Cascading Style Sheet" on page 727).

HTML Files

In the HTML files that make up the help system, there are many files that have generic content (based on siebendusegeneric.htm). With the files already in place, you can simply use the existing file to customize the content to suit your implementation. The following table lists the files that do not have generic content.

Filename	Description
siebbasicscontrols.htm	Using Field Controls
siebbasicscontrolscal.htm	Selecting Date and Time Information
siebbasicscontrolscurr.htm	Using the Currency Calculator

Filename	Description
siebbasicsdata.htm	Working with Data
siebbasicsdatatoc.htm	Table of contents: Working with Data
siebbasicskbshortcuts.htm	Using Keyboard Shortcuts
siebbasicsnav.htm	Navigating the Application
siebbasicsselectiondbs.htm	Using Selection Dialog Boxes
siebcalendar.htm	Using the Calendar
siebcharts.htm	Using Charts
siebcorrespondence.htm	Using Siebel Correspondence
siebcorrespondencetoc.htm	Table of contents: Using Siebel Correspondence
siebewireless.htm	Mobile help topics
siebfindinginfotoc.htm	Table of contents: Finding Information
siebhomepage.htm	Using the Home Page
siebhomepagetoc.htm	Table of contents: Using the Home Page
siebindex.htm	Help Index
siebliterature.htm	Working with Literature
sieblitera turetoc.htm	Table of contents: Working with Literature
sieboptions.htm	Setting User Options
sieboptionstoc.htm	Table of contents: Setting User Options
siebpaging.htm	Siebel Paging help topics
siebquery.htm	Using Queries
siebquerytoc.htm	Table of contents: Using Queries
siebreports.htm	Working with Reports
siebsearch.htm	Using the Siebel Search Center
siebsearchtoc.htm	Table of contents: Using the Siebel Search Center
siebstarthelp.htm	Start page (main table of contents)

Filename	Description
siebsynch.htm	Synchronizing Data
siebsynchtoc.htm	Table of contents: Synchronizing Data
siebuserinterfacetoc.htm	Table of contents: Understanding the User Interface
siebusingcalendartoc.htm	Table of contents: Using the Calendar
siebwelcome.htm	Welcome to Siebel Online Help

Customer Applications Files

The help system for customer applications consists of HTML files (listed in HTML Files), image files, and a cascading style sheet (siebhelp.css, described in "Cascading Style Sheet" on page 727).

HTML Files

The following table lists the HTML files that make up the customer application help system.

Filename	Description
siebcomcontactus.htm	Contacting Us
siebcomfindinginfo.htm	Finding Information
siebcomgeneric.htm	Start page (main table of contents)
siebcomgenericcontactustoc.htm	Table of contents: Contacting Us
siebcomgenericfindinginfotoc.htm	Table of contents: Finding Information
siebcomgenericlogintoc.htm	Table of contents: Logging In
siebcomgenericregistrationtoc.htm	Table of contents: Registration
siebcomgenericuserinterfacetoc.htm	Table of contents: Understanding the User Interface
siebcomloggingin.htm	Logging In
siebcomregistration.htm	Registration

Filename	Description
siebcomsearch.htm	Using Search
siebcomupdatinguserprofile.htm	Updating Your User Profile
siebcomuserinterface.htm	Understanding the User Interface
siebcomwwattachments.htm	Viewing Attachments

Cascading Style Sheet

Siebel online help uses a cascading style sheet (siebhelp.css) to control the appearance of the help pages. A cascading style sheet includes typographical and formatting information on how the Web page should appear, such as the text font. A cascading style sheet gives the author control over the appearance of the page.

Index

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

W