Siebel Developer's Reference

Version 7.8, Rev. A

September 2005

Contents

1 What's New in This Release

What's New in Siebel Developer's Reference, Version 7.8, Rev A

Table 2 lists changes described in this version of the documentation to support Release 7.8 of the software.

Table 1. What's New in Siebel Developer's Reference, Version 7.8, Rev A

Topic	Description
Throughout	References to Siebel Wireless Messaging are deleted.
CSSBCFile Methods	Clarified documentation of the following methods: ■ CreateFile ■ DeleteFile ■ GetFile ■ PutFile ■ UpdateSrcFromLink
Chapter 4, "User Properties"	Added or clarified documentation of the following user properties: ■ Default Bookmark View ■ DefaultPrefix ■ Display Mask Char ■ Encryption User Properties (all) ■ NoDelete Field ■ On Field Update Set n ■ PostMainViewData ■ Use Literals For Like
Applet List (Base/EditList)	Clarified documentation of this applet Web template.
Precedence	Clarified documentation of order of precedence of operations.
Functions in Calculation Expressions	Expanded documentation of InvokeServiceMethod function.

What’s New in Siebel Developer’s Reference, Version 7.8

Table 2 lists changes described in this version of the documentation to support Release 7.8 of the software.

Table 2. New Features in Siebel Developer’s Reference, Version 7.8

Topic	Description
Chapter 2, “Business Component Classes”	Methods that are not supported for use by Siebel customers are deleted. Obsolete classes are deleted.
Using Methods in Business Component Classes	This added topic describes requirements for invoking methods in business component classes.
Chapter 4, “User Properties”	Documentation for several business component user properties is added. User properties that are not intended for Siebel customer use are deleted.
Setting Numbered Instances of a User Property	This added topic describes requirements for using multiple instances of a user property on a business component.

2 Business Component Classes

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
- CSSBCFINOppty Class
- CSSBCFINSActivity Class
- CSSBCForecast Class
- CSSBCForecastBase Class
- CSSBCForecastItem Class
- CSSBCForecastItemDetail Class
- CSSBCOppty Class
- CSSBCPharmaSpecializedAct Class
- CSSBCProposal Class
- CSSBCServiceRequest Class
- CSSBCUser Class

Using Methods in Business Component Classes

Only the methods documented in the *Siebel Object Interfaces Reference* are supported for use in scripting. The typical means for invoking any of the methods that are accessible in business component classes is by using the InvokeMethod method. Do not assume that any method that is accessible in business component classes can be invoked directly, that is, by its method name and arguments only, unless it is specifically stated that the method can be invoked directly.

The syntax of InvokeMethod varies depending on the scripting language that you use, so specific syntax is not provided in the descriptions of methods that are accessible in business component classes.

NOTE: Input arguments for methods that are accessible in business component classes are listed in the order in which they must be provided in the call, independent of the scripting language you use.

Some methods can be used to underlie custom buttons and commands. Whether a method underlying a button or command is invoked by custom script or by script provided in the preconfigured application, typically the method must be invoked by using InvokeMethod.

For information on InvokeMethod and its syntax, see *Siebel Object Interfaces Reference*.

For information about configuring buttons and commands, see *Configuring Siebel Business Applications*.

NOTE: Intercepting a method and augmenting its logic before or after it is invoked can cause unpredictable behavior in your Siebel application.

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 73.

- All Mode Sort
- DB2 Optimization Level

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 24.

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

Business Component 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 73.

- 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 73.

- 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 a conditional Siebel expression against the current row and returns Y if the expression is true, or N in the result parameter.

Argument	Description
expr_string	The conditional expression to be evaluated.

Origin Implemented in CSSBCBase.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	Browser Script	Custom Button	Command	External Interfaces	Comment
Yes	Yes	No	No	Yes	

EvalExpr

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

Origin Implemented in CSSBCBase.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	Browser Script	Custom Button	Command	External Interfaces	Comment
Yes	Yes	No	No	Yes	

IsActive

The IsActive method determines whether the row is active by reading the Active field value and returns Y or N.

Origin Implemented in CSSBCBase.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	Browser Script	Custom Button	Command	External Interfaces	Comment
Yes	Yes	Yes	No	Yes	

Revise

The Revise method creates a new revision of the current record. It is commonly used to create revisions of quote, order, and agreement records.

This method is similar to BusComp_CopyRecord, except:

- Revise marks the current record as inactive if the Active Field user property is defined.
- Revise locks the current record if the Locked Field and Locked By Field user properties are defined.
- Revise increments the revision number of the new record. The field for the revision number is specified by the Revision Field user property.
- Revise copies values in particular fields of the existing record to the new record, as specified by the Revision Copy Field user property.

For information about the BusComp_CopyRecord event, see *Siebel Object Interfaces Reference*.

See also

"Active Field" on page 74

"Revision Field" on page 159

"Revision Copy Field n" on page 159

Origin Implemented in CSSBCBase.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	**Browser Script**	**Custom Button**	**Command**	**External Interfaces**	**Comment**
Yes	Yes	Yes	No	Yes	

Example: Command From the Siebel Call Center Site Map, choose Agreements > List. In the Agreement List Applet No Parent in the resulting My Agreement List View, the Revise method underlies the Revise command in the main menu. By choosing Revise, the new record's Revision field is incremented.

Sequence

The Sequence method regenerates the sequence numbers for all the records in the current business component for which sequencing has been set. The starting value of the sequence numbers is determined in one of the following ways:

- For a business component that has a sequenced field, there is a corresponding sequence business component that has a Sequence field. The Predefault Value property of that Sequence field is the first option for defining the starting value for the sequence.

  For example, the FS Agreement Item business component has the sequenced field Line Number, as specified by the Sequence Field user property. In the corresponding FS Agreement Item.Line Number (Sequence) business component, the Sequence field's Predefault Value property can define the starting sequence value, usually other than 1.

- If a predefault value is not set for the sequence, as described above, the sequence defaults to a starting value of 1.

NOTE: Configuring a sequence field on a business component requires several tasks to be completed. For detailed information on creating sequence fields, see *Configuring Siebel Business Applications*.

See also

"Sequence Field" on page 160

"Sequence Use Max" on page 161

Origin Implemented in CSSBCBase

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	**Browser Script**	**Custom Button**	**Command**	**External Interfaces**	**Comment**
Yes	Yes	Yes	Yes	Yes	

SetAspect

The SetAspect method sets and overrides the default aspect of the current business component.

This method is typically called by applet code or script to override the aspect of the business component with the applet's aspect.

See also

"Aspect User Properties" on page 80

Argument	Type	Description
aspect	string	The name of the aspect to set as current.
reset_bool	string	Optional. The value of the parameter is Y or N. If the value is Y, the aspect of the business component is reset to its default aspect, if it has one. If the value is N or the parameter is not included, the aspect specified in the first parameter is used.

Origin Implemented in CSSBCBase

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	Browser Script	Custom Button	Command	External Interfaces	Comment
Yes	Yes	No	No	No	

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.

Business Component 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 73.

- Maintain Master Account (required)
- Master Account Field (required)
- Parent Account Field (required)
- 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 back-end 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 purposes. 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 32.

- ClearGridBeginEndDate
- CompleteActivity

- IsPrimaryInMVG
- SetEmployeeId
- SetGridBeginEndDate

Business Component 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 73.

- 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 this record's type of Activity.

The fields listed in Table 3 are required for CSSBCActivity.

Table 3. Fields Required by CSSBCActivity

Field Name	Description
Primary Owned By Primary Owner Id	Key fields to visibility control.
Orig Appt Id	Original Appointment Id.

Table 3. Fields Required by CSSBCActivity

Field Name	Description
Done Due Due Date Exchange Date No Sooner Than Date Planned Planned Completion Repeating Expires Started	Key date field on which both Activity and Calendar are dependent.
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.
Percent Complete	
Personal Postal Code Service Region	
Previous Activity Id	

The CSSBCActivity class is required when the 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.

ClearGridBeginEndDate

The ClearGridBeginEndDate method returns the business component to regular mode after it has been set to calendar mode with the SetGridBeginEndDate method.

See also

"SetGridBeginEndDate" on page 35

Origin Implemented in CSSBCActivity.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	**Browser Script**	**Custom Button**	**Command**	**External Interfaces**	**Comment**
Yes	Yes	No	No	No	You can invoke this method from applets, business components, or business services.

CompleteActivity

The CompleteActivity method commits the activity record and calculates the costs associated with an activity. It then updates the parts, time, and expense records of the activity with the costs of each item based on the price list, rate list, and cost list.

In the preconfigured application, the CompleteActivity method is invoked by the Complete Activity business service. The Complete Activity business service is called when an activity record is saved.

This method is typically used with the Action business component, but may be used with other business components in the CSSBCActivity class.

This method cannot be called when the applet is in query mode.

Origin Implemented in CSSBCActivity.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	**Browser Script**	**Custom Button**	**Command**	**External Interfaces**	**Comment**
Yes	Yes	Yes	Yes	No	You can invoke this method from applets, business components, and business services.

IsPrimaryInMVG

The IsPrimaryInMVG method returns Y or N to indicate whether the logged-in user is the primary in the multi-value group of a specified field.

This method can be used to determine whether the logged-in user is allowed to perform operations that are limited to the primary in the multi-value group. For example, only the primary may be allowed to change the primary.

Argument	Type	Description
fieldname	string	The name of the field to check.

Origin Implemented in CSSBCActivity.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	Browser Script	Custom Button	Command	External Interfaces	Comment
Yes	Yes	No	No	No	You can invoke this method from applets, business components, or business services.

SetEmployeeId

The SetEmployeeId method sets the criteria of the next SQL query on the current activity business component to the row Id and login provided as input arguments, to enable visibility of that employee's calendar records.

Argument	Type	Description
emp_login_id	string	The row Id of the employee whose calendar records are returned.
emp_login_name	string	The Login of the employee whose calendar records are returned.

Origin Implemented in CSSBCActivity.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	Browser Script	Custom Button	Command	External Interfaces	Comment
Yes	Yes	No	No	No	You can invoke this method from applets, business components, and business services.

SetGridBeginEndDate

The SetGridBeginEndDate method sets the business component to Calendar mode and sets the beginning and ending dates for the grid.

This method is required in script that manipulates the business component in Calendar mode, such as manipulating the instances of a recurring activity.

CAUTION: To avoid performance impact, set the date interval so that a large number of records, for example more than 1000, are not returned. Typically, the interval should be set as a month or a week. In the preconfigured application, this method is typically invoked with intervals of a calendar month or a calendar week.

The business component is returned from Calendar mode to its regular mode by the ClearGridBeginEndDate method.

Argument	Type	Description
beginDate	string	The beginning date for the grid in the object manager's time zone in the form *mm/dd/yyyy*. The beginning time for this date is interpreted as midnight at the beginning of the day.
endDate	string	The ending date for the grid, also in *mm/dd/yyyy* form. The ending time for this date is interpreted as midnight at the end of the day.

Origin Implemented in CSSBCActivity.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	Browser Script	Custom Button	Command	External Interfaces	Comment
Yes	Yes	No	No	No	You can invoke this method from applets, business components, or business services.

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 37.

- AffiliatedContacts
- AllContacts

Business Component 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 73.

- AutoAssignSearch
- BC Opportunity
- BC Position
- Contact-Activity BC Name
- Contact-Opportunity BC Name
- eAuto Status Field Name
- eAuto Status Field Value
- Enable Dispatch Board
- 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 73.

- 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.

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 39.

- CreateFile
- GetFile
- GetFile
- PutFile
- UpdateSrcFromLink
- Inherited CSSBCBase Methods

Business Component User Properties

Not applicable

Field User Properties

Not applicable

Dependencies and Limitations

CSSBCFile requires the fields listed in Table 4. 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 business component has fields named AccntDockStatus, AccntFileDate, AccntFileName, and so on.

Table 4. 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

Table 4. Fields Required by CSSBCFile

Field Name	Value Type
<*Prefix*>FileSrcType	TEXT
<*Prefix*>FileDockStatFlg	BOOL

CSSBCFile Methods

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

CreateFile

The CreateFile method places a copy of an external file into the Siebel File System and attaches it to the current record by updating the relevant fields in the business component. If the value of *keepLink* is Y, then the link to the external file is stored.

Argument	Type	Description
srcFilePath	string	The path to the source file.
keyFieldName	string	The name of the <*Prefix*>FileName field, which stores the name of the file in the Siebel File System. For example, for the Account Attachment business component, this field is AcctsFileName.
keepLink	string	Indicator of whether to keep a link to the external file. Allowed values are Y and N.
altSrcFileName	string	Optional. An alternative filename for the file that is created in the Siebel File System, if it is different from the name of the source file from which it is copied.

Returns The string “Success” is returned if the operation succeeded, else “Error” is returned.

Origin Implemented in CSSBCFile.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	Browser Script	Custom Button	Command	External Interfaces	Comment
Yes	No	Yes	Yes	Yes	

DeleteFile

The DeleteFile method deletes a file in the Siebel File System or an external file.

Argument	Type	Description
fileName	string	The name of the file to be deleted, including its path.
internal	string	Optional. This argument indicates whether the file is an internal file, that is, a file in the Siebel File System. Allowed values are True if the file is an internal file or False if the file is not an internal file. A value of False is assumed if this argument is not provided.

Returns The string “Success” is returned if the operation succeeded, else “Error” is returned.

Origin Implemented in CSSBCFile.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	**Browser Script**	**Custom Button**	**Command**	**External Interfaces**	**Comment**
Yes	No	Yes	Yes	Yes	

GetFile

The GetFile method copies a file in the Siebel File System that is attached to the current record into a temporary directory. The method returns the path to the file in the temporary directory.

The temporary directory is defined in the cfg file for the Siebel application as the value of the `TmpDir` parameter in the `[Siebel]` section.

This method allows a user to view or edit a file attachment.

See also

"PutFile" on page 41

Argument	Type	Description
keyFieldName	string	The name of the <*Prefix*>FileName field, which stores the name of the file in the Siebel File System. For example, for the Account Attachment business component, this field is AcctsFileName.

Returns The return value is one of the following:

- The string "Success, *outFilePath*" if the operation succeeded. *OutFilePath* is the path to the file that is copied into the temporary directory.
- The string "Error" if the file is not copied successfully to the temporary directory.
- The string "OutOfDate" if the file in the File System is copied to the temporary directory, but that file is not the most recent version of the file. The most recent version of the file was not available in the File System to be copied.

Origin Implemented in CSSBCFile.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	Browser Script	Custom Button	Command	External Interfaces	Comment
Yes	No	Yes	Yes	Yes	

PutFile

The PutFile method replaces a file in the Siebel File System that is attached to the current record with a copy of a file in a specific directory. The method updates relevant business component fields.

This method is used to update a file attachment.

See also

"GetFile" on page 40

Argument	Type	Description
fileName	string	The name of a file, with its full path, from which the attached file is updated.
keyFieldName	string	The name of the <*Prefix*>FileName field, which stores the name of the file attachment in the Siebel File System that is to be updated. For example, for the Account Attachment business component, this field is AcctsFileName.

Returns The string "Success" is returned if the operation succeeded, else "Error" is returned.

Origin Implemented in CSSBCFile.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	Browser Script	Custom Button	Command	External Interfaces	Comment
Yes	No	Yes	Yes	Yes	

UpdateSrcFromLink

The UpdateSrcFromLink method replaces a file in the Siebel File System that is attached to the current record with an external file to which the replaced file is linked. The method updates relevant business component fields.

This method is used to update a file attachment when the external file to which it is linked is modified.

See also

“CreateFile” on page 39

Argument	Type	Description
keyFieldName	string	The name of the <*Prefix*>FileName field, which stores the name of the file attachment in the Siebel File System that is to be updated. For example, for the Account Attachment business component, this field is AcctsFileName.

Returns The string “Success” is returned if the operation succeeded, else “Error” is returned.

Origin Implemented in CSSBCFile.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	Browser Script	Custom Button	Command	External Interfaces	Comment
Yes	No	Yes	Yes	Yes	

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 method is accessible from CSSBCFINOppty. For more information on this method, see “CSSBCFINOppty Methods” on page 44.

- SetSecureAdminView

Business Component 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 73.

- 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.

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.

Business Component 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 73.

- 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 47.

- ForecastGenerate
- RollupForecast

Business Component 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 73.

- 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 whose names begin with "Forecast 2000" are heavily dependent on each other. Forecast business components are also dependent on revenue business components. When you make changes to one of these business components, you should also make similar changes to its dependent business components. For example:

- If you make changes to Forecast 2000 -- Forecast Item Detail, you should also make similar changes to Forecast 2000 -- Forecast Item Detail Flat.
- If you add fields to revenue business components, and you want those changes included in forecasts, then you must add corresponding fields to the applicable forecast business component and columns to the tables underlying the forecast 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.

The ForecastGenerate method fills out the creation of a forecast after the forecast header (top-level) record is created. ForecastGenerate queries for the correct Revenue records, and then creates the detail records and summary records for the forecast.

ForecastGenerate can be used synchronously or asynchronously. In the preconfigured application, this method is invoked when a new forecast is saved.

- **Create forecast synchronously.** When a new forecast record is initially saved, ForecastGenerate is invoked to create detail and summary records immediately. This behavior is the default behavior in the preconfigured application. To use ForecastGenerate synchronously, you must have the following setting:
  - The Forecast: Use Server Task system preference must be set to FALSE, so that the application does not check the Forecast Service Manager server component.
- **Create forecast asynchronously.** When a new forecast record is initially saved, ForecastGenerate is invoked to create detail and summary records in the background, thereby allowing the user to use other parts of the application concurrently.

  To use ForecastGenerate asynchronously, you must have the following settings:
  - The Forecast: Use Server Task system preference must be set to TRUE, so that the application looks at the Forecast Service Manager server component.
  - The Forecast Service Manager server component (in the Forecast Service Management component group) must be enabled. If it is disabled, no detail or summary records are created; only the top-level forecast record is created.

NOTE: When the WriteRecord method saves a new forecast header record, the default behavior of the preconfigured application is to call ForecastGenerate to create the detail and summary records. You can opt instead to create header records only, with no detail or summary records. To do so, you must provide script outside of WriteRecord that uses InvokeMethod to call the SetWriteRecordsWithGenerate method with an input parameter of N. For subsequent new forecasts, ForecastGenerate is not invoked when new records are saved. No detail or summary records are created unless ForecastGenerate is called elsewhere in script. By calling SetWriteRecordsWithGenerate with an input parameter of Y, you can reset the application to call ForecastGenerate automatically when subsequent new forecast records are saved.

To not invoke ForecastGenerate during the save record process is different from asynchronously creating a forecast, in which ForecastGenerate is invoked automatically, but runs in the background to allow the user to do other tasks in the user interface.

Unless you plan to selectively turn ForecastGenerate on and off, it is likely that you do not need to invoke SetWriteRecordsWithGenerate in script. Typically, just set the Forecast: Use Server Task system preference and Forecast Service Manager server component to generate detail and summary records either synchronously or asynchronously whenever a new forecast record is created.

There may be a case for which you do not want ForecastGenerate to be triggered by WriteRecord. However, because typical use of forecasts is to include detail and summary records, it is unlikely that ForecastGenerate is not invoked at all. Thus, you will likely want to toggle the server component to invoke ForecastGenerate synchronously or asynchronously during WriteRecord, instead not invoking ForecastGenerate at all during WriteRecord.

Argument	Type	Description
copyFcstId	string	Forecast Row Id of the previous forecast in the same series. If copy forecast is used to create one forecast from the previous forecast, then changes made to the detail records of the copied forecast are included in the new forecast. If this method underlies the New button, then this field should be NULL.
bForce	string	Y or N. This parameter indicates whether to force regeneration if this forecast already exists.
bAutoForecast	string	Y or N. This parameter indicates whether to automatically associate subordinates' forecasts, if they exist. This parameter is typically set to Y.
bNeedRollup	string	Y or N. This parameter indicates whether to roll up the generated forecast automatically. If set to N, no summary records are generated for this forecast until the user manually selects Rollup from the user interface. This parameter is typically set to Y.
bImplicitCreateFcst	string	Y or N. This parameter indicates whether to create a missing subordinate's forecast. If set to Y, missing subordinates' forecasts are automatically generated. If set to N, missing subordinates' forecasts are disregarded. In the preconfigured application, a dialog box pops up when the new forecast record is being saved. The pop-up dialog box asks the manager whether to automatically create subordinates' forecasts, if they do not exist.

Origin Implemented in CSSBCForecast.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	Browser Script	Custom Button	Command	External Interfaces	Comment
Yes	Yes	Yes	Yes	Yes	You can also invoke this method from a business service.

RollupForecast

The RollupForecast method calculates the summary records for a forecast based on the detail records. Summary records are not updated dynamically; the RollupForecast method must be invoked, either by a control or programmatically, after detail records are modified. This method must have a current active row on the top-level forecast business component.

The RollupForecast method is typically used when a forecast is created to generate the initial summary records. This method underlies the Rollup button or menu choice on Forecast applets.

See also

"RollupParentForecast" on page 51

Origin Implemented in CSSBCForecast.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	**Browser Script**	**Custom Button**	**Command**	**External Interfaces**	**Comment**
Yes	Yes	Yes	Yes	Yes	You can also invoke this method from a business service.

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 51.

- RollupParentForecast

Business Component 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 recalculates the summary records for a forecast. Summary records are not automatically recalculated when line item detail records are changed.

The RollupParentForecast method underlies the Rollup button or menu choice on Forecast detail and summary applets. RollupParentForecast performs the same function as RollupForecast on CSSBCForecast. RollupParentForecast is called from child business components of CSSBCForecast, whereas RollupForecast is called directly from the CSSBCForecast business component.

Origin	Implemented in CSSBCForecastBase.
Invocable	This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	**Browser Script**	**Custom Button**	**Command**	**External Interfaces**	**Comment**
Yes	Yes	Yes	Yes	Yes	You can also invoke this method from a business service.

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

Not applicable

Business Component 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 73.

- 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*.

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 53.

- AutoAdjust
- UpdateSelectionFromRevn
- UpdateSelectionToRevn

Business Component 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 73.

- 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 is called on individual forecast detail records. It retrieves adjustments made to the previous forecast in the series and makes the same adjustment to the current selection of records.

In the preconfigured application, this method is used in script that underlies buttons and menu choices in forecast detail views.

Origin Implemented in CSSBCForecastItemDetail.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	**Browser Script**	**Custom Button**	**Command**	**External Interfaces**	**Comment**
Yes	Yes	Yes	Yes	Yes	You can also invoke this method from a business service.

UpdateSelectionFromRevn

The UpdateSelectionFromRevn method is called on individual Forecast detail records. It updates the Forecast detail record with data from the Revenue record with which the Forecast detail record is associated.

This method is typically used in script that underlies user interface controls.

Origin Implemented in CSSBCForecastItemDetail.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	**Browser Script**	**Custom Button**	**Command**	**External Interfaces**	**Comment**
Yes	Yes	Yes	Yes	Yes	You can also invoke this method from a business service.

UpdateSelectionToRevn

The UpdateSelectionToRevn method is called on individual Forecast detail records. It updates the Revenue record with which the Forecast detail record is associated with data from the Forecast detail record.

This method is typically used in script that underlies user interface controls.

Origin Implemented in CSSBCForecastItemDetail.

Invocable This method can be invoked through InvokeMethod only.

Can be invoked by:					
Server Script	**Browser Script**	**Custom Button**	**Command**	**External Interfaces**	**Comment**
Yes	Yes	Yes	Yes	Yes	You can also invoke this method from a business service.

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.

- Inherited CSSBCBase Methods

Business Component 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 73.

- Primary Position Modification

Field User Properties

Not applicable

Dependencies and Limitations

None

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

Business Component 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 73.

- 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 57.

- RequestAllFiles

Business Component 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.

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.

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.

- Inherited CSSBCBase Methods

Business Component 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 73.

- 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.

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.

- Inherited CSSBCBase Methods

Business Component 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 73.

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

Field User Properties

Not applicable

Dependencies and Limitations

None

3 Applet Classes

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:

NOTE: Only the methods documented in the Object Interfaces Reference are supported for use in scripting. Intercepting a method and augmenting its logic before or after it is invoked can cause unpredictable behavior in your Siebel application.

Relationship Between SWE and Non-SWE Classes

In Release 7.x and later, the Siebel Web Engine does a runtime conversion of all classes prefixed with "CSSFrame," and calls to their methods, to their counterparts that are prefixed with "CSSSWEFrame." Thus, the descriptions in this document apply to both classes in each of these pairs of classes. For example, calls to methods in the CSSFrameBase class are converted to calls to their corresponding methods in the CSSSWEFrameBase class. The description of functionality in "CSSFrameBase and CSSSWEFrameBase Classes" on page 63 applies to both classes.

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 61.

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 62.

- ExecuteQuery
- NewQuery
- Inherited CSSSWEBase Methods

Applet User Properties

The following applet user property is available for use in CSSSWEFrame. For more information on this and other user properties, see "User Properties" on page 73.

- Default Applet Method

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.

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.

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.

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 61.

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 64.

- GotoView

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 73.

- 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.

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 61.

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 63.

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 61.

Usage Guidelines

You can use this class to invoke file attachment methods.

Parent

CSSSWEFrameList

Accessible Methods

Not applicable

Applet User Properties

Not applicable

Control User Properties

Not applicable

Dependencies and Limitations

CSSSWEFrameListFile requires the underlying business component to use CSSBCFile.

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 61.

Usage Guidelines

You can use this class only in ERM-related applications. If you want to use a modified version of 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 73.

- 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 61.

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

Not applicable

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 73.

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

Control User Properties

Not applicable

Dependencies and Limitations

None

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 73.

- 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

Not applicable

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 73.

- ApplicationContextType
- DataSourceBuscompName
- DefaultAppletFocus
- DocumentContextType

Control User Properties

Not applicable

Dependencies and Limitations

None

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

Not applicable

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 73.

- 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.

4 User Properties

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
- Application
- Assignment
- Business Component
- Business Service
- Business Service Method Arg
- Control
- Field
- Integration Component
- Integration Component Field
- Integration Object
- List Column
- View
- Virtual Business Component

Only supported user properties are documented.

Setting Numbered Instances of a User Property

Several user properties can have multiple instances on a single business component. These user properties are represented as *User Property Name n*, such as On Field Update Invoke 1, On Field Update Invoke 2, and so on. In most cases, the number is not required if only one instance of the user property is set on the business component.

For the active instances of such numbered user properties, do not have gaps between numbers of more than 9 and do not append numbers of more than two digits. For example, the following scenarios would produce unwanted results:

- On Field Update Invoke 10, with no instances with numbers less than 10
- Deep Copy 9 and Deep Copy 19, with no instances numbered between 9 and 19

- Named Method 100

Active Field

The Active Field and the Active Value user properties together determine whether a record is active. An active record can be updated. A record that is not active cannot be updated.

For a given business component, the Active Field user property specifies the field on the business component that is the active flag. The Active Value user property specifies how to interpret the value of the flag.

See also "Active Value" on page 74.

Value	The value of the Active Field user property must be the name of a field on the current business component, not enclosed in quotes. The named field must be the active flag for the business component. If the business component has the Active Value user property and its value is Y, or if the Active Value user property is not defined, then a record is active if the value of its active flag is Y. If the value of the Active Value user property is N, then a record is active if the value of its active flag is N. For example, the value of the Active Field user property on the Quote business component is Active. The Active Value user property is not defined on Quote. Thus a record is active if its Active field value is Y. Alternatively, the value of the Active Field user property on the Fund business component is Locked Flag. The value of its Active Value user property is N. Thus a record is active if its Locked Flag field value is N—that is, when it is unlocked.
Usage	You can inactivate this user property or modify its values. You cannot create more than one instance of this user property for a business component.
Parent Object Type	Business Component
Functional Area	Various

Active Value

The Active Value and the Active Field user properties together determine whether a record is active. An active record can be updated. A record that is not active cannot be updated.

For a given business component, the Active Field user property specifies the field on the business component that is the active flag. The Active Value user property specifies how to interpret the value of the flag.

See also “Active Field” on page 74.

Value	Y or N. To use the Active Value user property, you must also set the Active Field user property on the business component to specify the field that contains the active flag. If the Active Value user property is also defined and its value is Y, or if the Active Value user property is not defined, then a record is active if the value of its active flag is Y. If the value of the Active Value user property is N, then a record is active if the value of its active flag is N. For example, the value of the Active Field user property on the Quote business component is Active. The Active Value user property is not defined on Quote. Thus a record is active only if its Active field value is Y. Alternatively, the default value of the Active Field user property on the Fund business component is Locked Flag. The value of its Active Value user property is N. Thus a record is active only if its Locked Flag field value is N—that is, when it is unlocked.
Usage	You can inactivate this user property or modify its values. You cannot create more than one instance of this user property for a business component.
Parent Object Type	Business Component
Functional Area	Various

Activity SearchSpec

This user property allows you to specify a search specification for the Action business component.

Value	The value for the Activity SearchSpec user property must be a valid search specification. For example, `[Status]= LookupValue('EVENT_STATUS', 'Open') AND [Class] = LookupValue('FS_ACTIVITY_CLASS', 'Sales Activity')`
Usage	You can inactivate or modify the value for this user property. However, you cannot create new instances this user property.
Parent Object Type	Business Component
Functional Area	CSSBCFINOppty

Admin Mode Field

The Admin Mode Field user property provides access to the current value of the business component’s Admin Mode Flag.

Although the Admin Mode Flag is a View property, the business component object has a corresponding internal Admin Mode Flag property. At runtime, the business component inherits the value of the Admin Mode Flag from the current view.

In Admin mode, updating, inserting, deleting, and merging records is permitted, independent of those property values on the business component.

For information on Admin Mode functionality, see *Security Guide for Siebel Business Applications* and *Configuring Siebel Business Applications*.

Value

The value of the Admin Mode Field user property must be the name of a field on the current business component, not enclosed in quotes. The named field should be an active calculated field of type DTYPE_BOOL with a value of "".

For example, for a given business component, create a calculated field, as described above, and name it IsAdminMode. This is the naming convention for such a field, although the name is not restricted. Add the Admin Mode Field user property on the business component with a value of IsAdminMode.

Usage

This user property is intended for use with script to identify whether a business component is currently in Admin mode. The following simple example displays Y if the business component is currently in Admin mode; otherwise, it displays N.

```
function BusComp_NewRecord ()
{
var isAdmin = this.GetFieldValue("IsAdminMode");
var WshShell = COMCreateObject("WScript.Shell");
WshShell.Popup(isAdmin);
}
```

You can inactivate this user property or modify its value. You can create new instances of this user property, but you should not create more than one instance for a business component.

Parent Object Type Business Component

Functional Area Scripting

Admin NoDelete

This user property prevents deleting records when a view, and thus its business component, are in Admin mode.

Administrative views often have their Admin Mode Flag property set to TRUE to allow administrators to insert, update, delete, and merge records on business components that are set to disallow some or all of these operations in typical views. Thus, setting the Admin NoDelete user property on a business component typically means that records cannot be deleted in any view.

For information on Admin Mode functionality, see *Security Guide for Siebel Business Applications* and *Configuring Siebel Business Applications*.

Value	Y or N.
	The value of this user property defaults to N if the user property is not defined on the business component.
Usage	You can inactivate this user property or modify its value. You can create new instances of this user property, but you should not create more than one instance for a business component.
Parent Object Type	Business Component
Functional Area	Various

Admin NoUpdate

This user property prevents updating of records when a view, and thus its business component, are in Admin mode.

Administrative views often have their Admin Mode Flag property set to TRUE to allow administrators to insert, update, delete, and merge records on business components that are set to disallow some or all of these operations in typical views. Thus, setting the Admin NoUpdate user property on a business component typically means that records cannot be updated in any view.

For information on Admin Mode functionality, see *Security Guide for Siebel Business Applications* and *Configuring Siebel Business Applications*.

Value	Y or N.
	The value of this user property defaults to N if the user property is not defined on the business component.
Usage	You can inactivate this user property or modify its value. You can create new instances of this user property, but you should not create more than one instance for a business component.
Parent Object Type	Business Component
Functional Area	Various

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.

Value	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 144). 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

This user property allows you to specify whether the Siebel application overrides the default sort specification.

Value	The value for the All Mode Sort user property must be one of 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.
Usage	Standard Siebel application behavior is to override the sort specification on views with certain visibility types to force the view to ORDER BY the standard user key. The All Mode Sort user property determines whether the Siebel application overrides the sort specification and, if so, determines the sort (if any) that is applied to the business component for the affected views. This user property affects views with visibility other than Personal or Sales Rep visibility, including Manager, All, Organization, Sub-Organization, Group, and Catalog views. **NOTE:** If you set up the default sort order so that it sorts on one of these views, be aware that this might 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 indexed. You should also conduct careful performance testing on the view itself and on reports run against the view.
Parent Object Type	Business Component
Functional Area	SSE Service Request, Action, and Activity Plan Action

Always Enable Child: *buscomp_name*

This user property allows you to specify whether a service request is frozen when it is closed.

Value	■ TRUE Specifies that service requests are not frozen when closed.
Usage	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. 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.
Parent Object Type	Business Component
Functional Area	SSE Service Request business component

ApplicationContextType

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

Value	■ 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.

Value	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 changes 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 becomes 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 becomes 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) is not changed when the user steps to a different record; otherwise, run-time behavior may confuse the end user. Improper configuration makes 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,* then the predefault value for this field becomes 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(), then the predefault value for the activity's planned start date and time becomes Timestamp().

Aspect user properties to use with a parent object type of Business Component are listed in Table 5.

Table 5. 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 6.

Table 6. 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 7.

Table 7. 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”*
View Aspect 1	*“View Name”, “Aspect Name”*
View Aspect 2	*“View Name”, “Aspect Name”*
Default Aspect	*Aspect Name*

Assignment Object

This user property provides the assignment object to which the Assignment Manager server component’s Assignment Object Name (AsgnObjectName) parameter is set when running interactive assignment.

Value The value of this user property is the name of an assignment object, not enclosed in quotes, that is a child of a workflow policy object. In the Siebel Tools Explorer, see Siebel Objects > Workflow Policy Objects > Assignment Objects for well-defined assignment objects.

For example, the Activity List View is based on the Action business component. By setting the Assignment Object user property on the Action business component to Activity, when a user chooses Menu > Assign while in a child form applet in the Activity List View, the activity, instead of some other object, is added back to the Assignment Manager queue for reassignment to a new owner.

Usage	You should inactivate this property only if you intend to turn off interactive assignment and you must also remove any related controls from the user interface. You can also modify the values for this user property. However, you cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Activities, service requests, opportunities

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.

Value	The value of the Associate: Completion Timeout (Client) user property is an integer greater than 0.
Usage	This user property is used during forecast association. When a subordinate forecast is being created, the application waits for the creation process to finish before associating the subordinate forecast. This user property specifies the amount of time for the application to wait before timing out. When the application times out, it throws an error indicating that some forecasts were unable to be associated. The user can subsequently associate the subordinate forecasts manually. This user property is used in synchronous mode. If you inactivate the Associate: Completion Timeout (Client) user property, the default value is used. 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

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.

Value	The value of the Associate: Completion Timeout (Server) user property is an integer greater than 0.

Usage	This user property is used during forecast association. When a subordinate forecast is being created, the application waits for the creation process to finish before associating the subordinate forecast. This user property specifies the amount of time for the application to wait before timing out. When the application times out, it throws an error indicating that some forecasts were unable to be associated. The user can subsequently associate the subordinate forecasts manually. This user property is used by the server component. If you inactivate the Associate: Completion Timeout (Server) user property, the default value is used. 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

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.

Value	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

This user property indicates that the integration component is an association business component.

Parent Object Type	Integration Component Field
Functional Area	eBusiness Application Integration Test

AssocFieldName [Field Name]

This user property denotes the name of the business component field as it appears on the MVG business component.

Parent Object Type	Integration Component Field
Functional Area	eBusiness Application Integration Test

AutoAssignSearch

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

Value 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.

Value

- 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

This user property allows you to specify the name of the BAPI adapter business services.

Value	The value for the BAPIAdapterService user property must be one of the following: ■ eAI ■ SAP ■ BAPI Adapter
Parent Object Type	Business Service
Functional Area	eBusiness Application Integration Business Services

BatchSize

This user property allows you to specify the number of IDOCs to read from SAP in a single call.

Value	The value for the BatchSize user property must be an integer.
Usage	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.
Parent Object Type	Business Service
Functional Area	SAP Integration

BC eAuto Sales Step

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

Value	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 value 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.

Value	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 value 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.

Value	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.

Value	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

This user property allows you to specify a field on the business component that determines whether individual records are read-only.

Value	The value for the BC Read Only Field user property must be the name of a field on the business component.
Usage	When the value of the field specified in this user property is TRUE, the current record is read-only. Setting the Admin Mode Flag property to TRUE overrides the BC Read Only Field user property. For information on the Admin Mode Flag, see "Admin Mode Field" on page 75.
Parent Object Type	Business Component
Functional Area	Data-Driven Access Control (Time Sheet)

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.

Value	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 value 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 to execute Actual Number calculation when a record is written.

Value	■ Y Indicates that the Actual Number is calculated when a record is written. ■ N Indicates that the Actual Number is not calculated.
Usage	You can inactivate or modify the value for this user property. You can also create new instances of this user property as needed.
Parent Object Type	Business Component
Functional Area	CSSBCFINOppty

ChargeBusinessService

This user property specifies the name of the business service that is used to calculate charges for service activities.

Value	The value for this user property is the name of a business service, not enclosed in quotes; for example, FS Service Charge.
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	Activity

ChargeBusinessServiceMethod*n*

This user property specifies the name of the method on the business service that is used to calculate charges for service activities.

Value	The value for this user property is the name of a method, not enclosed in quotes, on the business service specified by the ChargeBusinessService user property. For example, ChargeBusinessServiceMethod1 could have the value CreateServiceCharges, which is a method of the FS Service Charge business service.
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	Activity

CloseOutFlag

This user property allows you to specify the value of the CloseOut Flg field for the parent business component (typically Opportunity).

Value	The value for the CloseOutFlag user property must be Y or N.
Usage	You can inactivate or modify the value for this user property. However, you cannot create new instances of this user property.
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.

Value	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.

Value	The value for the Contact MVG PreDefault Expression user property uses the following syntax: *‘Parent Buscomp1 Name.Parent Buscomp1 Field Name’, ‘Parent Buscomp2 Name.Parent Buscomp2 Field Name’* You can list multiple business component-field pairs, enclosed in single quotes and separated by commas. Alternatively, you can prefix the list with `Parent:` at the beginning of the value as the preconfigured user property for the 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.

Value	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.

Value	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

This user property allows you specify whether to copy contact associations to a campaign when the status of the campaign is changed from Planned to Active.

Value	The value for the Copy Contact user property must be either TRUE or FALSE.
Usage	If this property is set to TRUE, then when the status of a campaign is changed from Planned to Active, all of a contact's associations for the planned campaign are copied to the active campaign. This user property is currently defined for DBM Campaign business component.
Parent Object Type	Business Component
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 does 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.

Value	■ 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).

Value	The value for the Credit Check Workflow user property must be the name of a workflow.
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

Currency Field *n*

This user property allows you to specify the name of a field that holds currency data.

Value	The value for the Currency Field *n* user property must be the name of a field on the business component.

Usage	When the currency code is changed, a currency exchange operation is performed on the data in the specified field and the value of the field is updated. You can inactivate and modify the values for this user property. You can also create new instances of this user property. See also "Setting Numbered Instances of a User Property" on page 73.
Parent Object Type	Business Component
Functional Area	CSSBCBase

DataCleansing Field *n*

This user property allows you to specify a correspondence between a field name in the Siebel Firstlogic Connector and a field name in the Siebel application.

Value	The value for the DataCleansing Field user property uses the following syntax:
	
	`“[Firstlogic Field Name]”, “[Siebel Field Name]”`
	
	For example, the following value specifies a correspondence between the Firm Location field in the Siebel Firstlogic Connector and the Primary Account Location field in the Siebel application.
	
	`"Firm Location", "Primary Account Location"`

Usage	Used for Siebel Data Quality, which performs address verification, name and address standardization, and duplicate record identification, in real-time and batch modes. **NOTE:** All data quality user properties require components from Firstlogic Corporation. See also "Setting Numbered Instances of a User Property" on page 73.
Parent Object Type	Business Component
Functional Area	Data Quality

DataCleansing Type

This user property allows you to specify to the Siebel Firstlogic Connector what kind of data is being validated in the Data Cleansing Field.

Value	The value for the DataCleansing Type user property must be one of the following: ■ 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.
Usage	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.
Parent Object Type	Business Component
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.

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

Usage	The specified business component is 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

Day Number: Arrival Date Field

This user property allows you to specify the Arrival Date field for the business component.

Value	The value for the Day Number: Arrival Date Field user property must be the name of a field in the business component.
Usage	The value of this user property specifies the name for the Arrival Date Field. It is used in the Day Number business service. When the value of this field is changed, the service propagates the change to the Date fields in the Function Space and Room Block business components. You can modify the value for this user property. However, you cannot inactivate or create new instances this user property.
Parent Object Type	Business Component
Functional Area	CSSBCFINOppty

Day Number: Function BC Name

This user property allows you to specify the associated Function Space business component.

Value	The value for the Day Number: Function BC Name user property must be the name of a business component.

Usage	The value of this user property specifies the name of the Function Space business component. It is used in conjunction with the Day Number: Arrival Date Field user property, to specify the business component that is updated by the Day Number business service when the Arrival Date field is changed. You can modify the value for this user property. However, you cannot inactivate or create new instances of this user property.
Parent Object Type	Business Component
Functional Area	CSSBCFINOppty

Day Number: Room Block BC Name

This user property allows you to specify the associated Function Space business component.

Value	The value for the Day Number: Room Block BC Name user property must be the name of a business component.
Usage	The value of this user property specifies the name of the Room Block business component. It is used in conjunction with the Day Number: Arrival Date Field user property, to specify the business component that is updated by the Day Number business service when the Arrival Date field is changed. You can modify the value for this user property. However, you cannot inactivate or create new instances of this user property.
Parent Object Type	Business Component
Functional Area	CSSBCFINOppty

DB2 Optimization Level

This user property allows you to change the optimization level of all of the SQL statements produced by the given business component.

Value	The value of the DB2 Optimization Level user property must be an integer.

Usage

The specified integer indicates the level of optimization to be used.

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.

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 that addresses the needs of both the where conditions on the driving tables of the query, and the 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 does not appear 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.

Parent Object Type

Business Component

DeDup Token Value

This user property allows you to specify the dedup token calculation expression for a business component.

Value The value for the DeDup Token Value user property consists of a calculation expression.

Usage The DeDup Token Value user property specifies 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. If the calculation value is changed, you must 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 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 ([Primary
  Account Name], 1), “_”) + IfNull (Left ([Last Name], 1), “_”)
  ```

Parent Object Type Business Component

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.

See also “Setting Numbered Instances of a User Property” on page 73.

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 Copying and Deleting

When you copy or delete a record, you can use the Deep user properties to propagate the change to child business components. For example, you can arrange to copy the detail records of a child business component from the original record to the new copied record of the parent business component.

For purposes of these user properties, the application checks the following sources in the order given to determine the parent/child link to use:

- If the parent and child are the same business component, then the relationship must be defined by the Recursive Link user property set on the business component. The link that the Recursive Link user property specifies is used to determine child records.

  See also "Recursive Link" on page 154.

- If the Deep Copy/Delete Link user property is set on the current (parent) business component, then the link that the Deep Copy/Delete Link user property specifies to the child business component is used.

  See also "Deep Copy/Delete Link" on page 102.

- If the parent and child business components are of the same Siebel object, and the parent is the primary business component in the business object, then the application looks up the link listed for the parent to the child, if one exists. If the link exists, then it is listed by choosing Siebel Tools Object Explorer > Business Object > Business Object Component. Under the applicable business object, the link displays in the Link column for the child business component in the Business Object Components list.

- If none of the sources mentioned in this list provides a link between the parent and child business components, then the application determines whether a link named *parent business component/child business component* exists (for example, Opportunity/Revenue). If such a link exists, then the application uses that link.

Deep Copy *n*

This user property allows you to specify a child business component that should be copied when a user selects the Copy option.

See also “Deep Copy/Delete Link” on page 102.

Value	The value for the Deep Copy *n* user property must be the name of a child business component for which a parent/child link is defined in one of the ways described in “Deep Copying and Deleting” on page 101.
Usage	The Deep Copy *n* user property allows child business components and their respective child business components to be copied automatically when selecting the Copy option. Normally, the 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 to avoid the SQL error “A Duplicate Record Exists” occurring.

Each business component in the Deep Copy chain takes care of its own children. The parent business component has Deep Copy properties for each of its direct children, and each child business component has 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.

See also “Setting Numbered Instances of a User Property” on page 73.

Parent Object Type	Business Component
Functional Area	Copying records

Deep Copy/Delete Link

The Link object defines a one-to-many parent/child relationship between two business components. The Deep Copy/Delete Link user property is set on the parent business component to specify the link to use for deep copies and deep deletes of the child business component named in Deep Copy/Delete Link.

See also

"Deep Copying and Deleting" on page 101

"Deep Copy n" on page 101

"Deep Delete n" on page 104

For information about the Link object type, see *Configuring Siebel Business Applications*.

Syntax	Deep Copy/Delete Link: *Buscomp*

Argument	Description
Buscomp	This parameter is the name of a business component for which: ■ The business component is the child in a link for which the current business component is the parent, and ■ Deep Copy or Deep Delete user properties are set on the current (parent) business component. This is an optional parameter. Use it when links must be specified for more than one business component.

Value	The value of this business component user property must be the name of a link between the current (parent) business component and the child business component specified by the *Buscomp* parameter.
Usage	For example, to specify a link to apply to deep copying or deep deleting of the revenues associated with an opportunity, add the user property Deep Copy/Delete Link: Revenue with value Opportunity/Revenue to the Opportunity business component. Opportunity is the parent and Revenue is the child business component of the Opportunity/Revenue link. You can inactivate or modify the values for this user property. You can also create new instances of this user property. **CAUTION:** Before you inactivate any predefined instances of this user property in a production environment, do thorough planning and testing to confirm that the child records that are to be no longer copied or deleted are those you intended, and that records of other business components related to those child records are not adversely affected.
Parent Object Type	Business Component
Functional Area	Copying and deleting records

Deep Delete *n*

This user property allows you to specify a child business component that should be deleted when a user selects the Delete option.

See also "Deep Copy/Delete Link" on page 102.

Value The value for the Deep Delete *n* user property must be the name of a child business component for which a parent/child link is defined in one of the ways described in "Deep Copying and Deleting" on page 101.

Usage 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.

To use this feature, do the following:

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:

   FALSE: allows deep delete for the child business component
   TRUE: does not allow deep delete for the 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, Deep Copy and Deep Delete are used together.

See also "Setting Numbered Instances of a User Property" on page 73.

Parent Object Type Business Component

Functional Area Deleting records

Default Applet Method

The Default Applet Method user property specifies the method that is executed when the user presses the Enter key in the applet.

Value The value of this user property must be the name of a method that is accessible from the applet.

Usage	The method specified in this user property is executed when the Enter key is pressed. You can inactivate and modify values for this user property. You can also create new instances of this user property, but only one such user property per applet.
Parent Object Type	Applet
Functional Area	CSSSWEFrame

Default Bookmark View

This user property specifies the default view in which to access a business component when a bookmark for this business component is created.

Value	The value of this user property must be the name of a view, not enclosed in quotes.
Usage	Communications Server uses this user property to create a bookmark when sending a package with the Attach Bookmark field checked. For example, this user property is preset to Opportunity List View for the Opportunity business component. If a bookmark to an Opportunity record is attached to an email message that is generated by Communications Server, then clicking on that bookmark takes the recipient to the linked record in the Opportunity List View, if the recipient has that view in his or her responsibility. For more information about creating and using Siebel bookmarks, see *Siebel Communications Server Administration Guide*. You can inactivate this user property or modify its value. You can create new instances of this user property, but you should not create more than one instance for a business component.
Parent Object Type	Business Component
Functional Area	Various

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#

Value	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

DefaultAppletFocus

This view user property sets the applet within a view that receives focus by default—that is, before a user interacts to dynamically change the applet with focus. This user property is provided to allow overriding the applet that receives default focus as determined by the view type of the view in Siebel Tools.

Value	The name of an applet in the view, not enclosed in quotes.
Usage	For example, the Account Screen Homepage View has the DefaultAppletFocus user property set with value Account Home Search Virtual Form Applet. When this view is first accessed, the applet with focus is the Account Home Search Virtual Form Applet. If the DefaultAppletFocus user property is not added to a view, then the applet with default focus is determined by the view type as specified in Object Explorer > Screen > Screen View. If the view type is Detail View, focus is placed on the second applet, if the view consists of two or more applets. If the view type is any other type, or there is only one applet on the view, focus is placed on the first applet. You can inactivate this user property or modify its value. You can create new instances of this user property, but you should not create more than one instance for a view. **CAUTION:** If you set default focus to an applet that is off the screen, one of two things may happen: 1) The user may not know where focus is, or 2) The application may try to adjust the vertical position of the view to try to show the in-focus applet. In either case, the behavior may be disruptive to end users.
Parent Object Type	View
Functional Area	Various

DefaultFocus User Properties

These applet user properties set the field or control within an applet that receives focus by default—that is, before a user interacts to dynamically change the field or control with focus. This user property is provided to allow overriding the field or control that receives default focus as determined by the applet's mode. The modes in which an applet can be deployed are defined in Object Explorer > Applets > Applet Web Templates.

CAUTION: If you set default focus to a field or control that is off the screen, one of two things may happen: 1) The user may not know where focus is, or 2) The application may try to adjust the vertical position of the view to try to show the in-focus field or control. In either case, the behavior may be disruptive to end users.

DefaultFocus_Edit

This applet user property sets the field or control within an applet that receives focus when the applet is in Base, Edit, or Edit List mode.

Value	The name of a field or control on the applet, not enclosed in quotes.
Usage	For example, this user property could be set to Last Name to give default focus to the Last Name field on an applet in Edit mode. If the DefaultFocus_Edit user property is not added to an applet, then no field or control typically gets default focus when the applet is in Base, Edit, or Edit List mode. You can inactivate this user property or modify its value. You can create new instances of this user property, but you should not create more than one instance for an applet.
Parent Object Type	Applet
Functional Area	Various

DefaultFocus_New

This applet user property sets the field or control within an applet that receives focus when the applet is in New mode.

Value	The name of a field or control on the applet, not enclosed in quotes.
Usage	For example, this user property could be set to NewRecord to give default focus to the New button on an applet being used to add a new record. If the DefaultFocus_New user property is not added to an applet, then the first field in the applet typically gets default focus when the applet is in New mode. You can inactivate this user property or modify its value. You can create new instances of this user property, but you should not create more than one instance for an applet.

Parent Object Type	Applet
Functional Area	Various

DefaultFocus_Query

This applet user property sets the field or control within an applet that receives focus when the applet is in Query mode.

Value	The name of a field or control on the applet, not enclosed in quotes.
Usage	For example, this user property could be set to First Name to give default focus to the First Name field on an applet in Query mode. If the DefaultFocus_Query user property is not added to an applet, then the first field in the applet typically gets default focus when the applet is in Query mode. You can inactivate this user property or modify its value. You can create new instances of this user property, but you should not create more than one instance for an applet.
Parent Object Type	Applet
Functional Area	Various

DefaultPrefix

This user property allows you to specify a common prefix for business component field names.

Value	The value for the DefaultPrefix user property is the string that is the prefix in names of the required fields for a file attachment business component. The value is not enclosed in quotes.

Usage	Each file attachment business component in the CSSBCFile class has a set of required fields. The names of those required fields are the same across file attachment business components, except for a prefix that is unique to each file attachment business component. See Table 4 on page 38. The value of the DefaultPrefix user property is the unique field name prefix of the file attachment business component. For example, the value of DefaultPrefix is “Accnt” for the Account Attachment business component because this business component’s required fields are named AccntDockStatus, AccntFileDate, AccntFileName, and so on. This user property is used by methods to construct the complete field names for required fields in any file attachment business component, in order to access those fields. By using the default prefix to build the required field names for a file attachment business component, a method can access those required fields without being provided the literal names of those fields. You can modify the value for this user property. You cannot create more than one instance of this user property, and you cannot inactivate this user property.
Parent Object Type	Business Component
Functional Area	CSSBCFile

DisableNewRecord

This user property allows you to prevent NewRecord from being invoked on the current applet in the specified Siebel application.

Value	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* disables 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

DisableSearch

This single-value field user property allows a Siebel developer to specify whether an end user can execute a wildcard query on a particular field.

Value	TRUE or FALSE
Usage	The intent of this field user property is to allow a Siebel developer to prevent users (and the Siebel query engine) from performing queries on non-indexed or text fields. If its value is TRUE, wildcard searching on the field is disabled, but exact match searching is allowed. If its value is FALSE or not specified, searching is allowed on the field. For example, if the [Name] field has the DisableSearch field user property set to TRUE, wildcard searches such as [Name] LIKE 'S*' are suppressed, and an error message is displayed. Exact searches such as [Name] = 'Siebel' are allowed. This user property is enforced when the following query options are exercised: query by example, Query Assistant, Search Center, queries initiated through programmatic or message-based interfaces, and pre-defined queries. You can inactivate and modify the values for this user property, and you can create a new instance of this user property if it is not already defined on the field.
Parent Object Type	Field
Functional Area	Search

DisableSort (Field User Property)

This single-value field user property allows a Siebel developer to specify whether an end user can sort a result set on a specific field of a business component.

Value	TRUE or FALSE
Usage	The intent of this business component field user property is to provide finer-grained control over field-level sorting. If its value is TRUE, all sorting capabilities on the field are disabled in all applets based on the business component. The sort icons and tool tips do not appear in the list column header, and the field is not displayed in the Advanced Sort window. If its value is FALSE or not specified, sorting is enabled on the field. For example, if the *Name* field for a particular business component has the DisableSort user property set to TRUE, the sort icons and (Sortable) tool tip do not appear in the *Name* list column, and the *Name* field is not displayed in the Advanced Sort window. If a user attempts to perform a sort on the *Name* field in any applet based on that business component, an error message displays. See also "DisableSort (Control User Property)" on page 111. You can inactivate and modify the values for this user property, and you can create a new instance of this user property if it is not already defined on the field.

Parent Object Type	Field
Functional Area	Sorting

DisableSort (Control User Property)

This control user property allows a Siebel developer to specify whether an end user can sort a result set on a specific field or list column control.

Value	TRUE or FALSE
Usage	The intent of this control user property is to allow a Siebel developer to prevent users (and the Siebel query engine) from sorting on nonindexed data. If its value is TRUE, all sorting capabilities on a particular field or list column are disabled. The sort icons and tool tip do not appear in the list column header, and the field is not displayed in the Advanced Sort window. If its value is FALSE or not specified, sorting is enabled by the control. For example, if the *Name* list column has the DisableSort user property set to TRUE, the sort icons and (Sortable) tool tip do not appear in the *Name* list column, and the *Name* field is not displayed in the Advanced Sort window. If a user attempts to perform a sort on the *Name* field, an error message displays. See also "DisableSort (Field User Property)" on page 110. You can inactivate and modify the values for this user property, and you can create a new instance of this user property if it is not already defined on the control.
Parent Object Type	Control
Functional Area	Sorting

Display Mask Char

This field user property allows you to display a masked version of secure data, typically a credit card number or account number.

Usage	This user property is used with the Encrypt Source Field user property to display only the last 4 digits of a credit card number or account number, such as xxxxxxxxxxxx9999. This user property is set on a separate calculated field that is displayed in the UI instead of the field containing the entire credit card number or account number. For example, in the Quote business component, the Credit Card Number field is the encrypted field that stores the credit card number. The Credit Card Number - Display field is a calculated field on which the following user properties are set: ■ Display Mask Char with a value of x ■ Encrypt Source Field with a value of Credit Card Number The applet field that displays the masked credit card number should reference the Credit Card Number - Display field. See also "Encrypt Source Field" on page 127.
Parent Object Type	Field
Functional Area	Encryption

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.

Value	The value for the DocumentContextType user property is a string that defines a proposal template category. The value must exactly match a value in PROPOSAL_TEMPLATE_TYPE LOV.
Usage	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* filters proposal templates so that only proposal templates with the Account Proposal defined for the Category field are available for the user to choose in the Template picklist in this applet. 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

This user property allows you to define the visibility applied to the new view during a single record drill down.

Usage	This user property is currently used in the preconfigured Siebel product. However, its function has been superseded by the Visibility Type attribute of the drilldown object.
Parent Object Type	Applet

Duplicate Elimination

If a query on a business component is executed in ForwardOnly mode, the same record may repeat in the result set of a query. The Duplicate Elimination user property is used to omit duplicate records from the result sets of queries on the business component that are executed in the ForwardOnly mode.

For information about executing queries in ForwardOnly mode, see *Siebel Object Interfaces Reference*.

Value TRUE or FALSE

If set to TRUE, duplicates of the same record are not returned in the result sets of queries executed on the business component in ForwardOnly mode. Aggregation is done in memory on the set of unique records only. The aggregation is not performed as part of the SQL.

If this user property is set to FALSE or it is not defined on the business component, duplicates of the same record are included in the result sets of queries executed on the business component in ForwardOnly mode. In general, the aggregation is done as part of the SQL, that is, the aggregation is done in the database layer. Aggregation in the database layer is preferred for performance considerations. However, the aggregation is done instead by the Siebel Object Manager after all rows are read from the database if any of the following exceptions apply:

- The Database Aggregation Flag parameter in the Server Datasource named subsystem is FALSE (in server mode), or the parameter DBAggregation parameter is defined and set to FALSE in the application's .cfg file on the Mobile Web client.

  For information on setting application object manager named subsystem parameters, and for information on setting .cfg file parameters on the Mobile Web client, see *Siebel System Administration Guide*.

- A search specification has clauses that cannot be evaluated by the database, thus requiring evaluation in memory.
- Aggregation in the database fails for some reason.

NOTE: The Duplicate Elimination user property only determines how aggregation is done. To enable aggregation on a list column or for a multi-value link requires that you first make several settings in Siebel Tools.

For information on configuring a list applet to display totals, see *Configuring Siebel Business Applications*.

For information on showing totals in a separate applet for a multi-value link, see *Configuring Siebel Business Applications*.

Usage	This user property allows the developer to configure whether duplicates are included in ForwardOnly queries on the business component. If a query is executed in ForwardOnly mode, the same record may repeat in the result set for one of the following reasons: ■ The intersection table has duplicate rows. For example, Start Date is part of the association, and associations with different start dates are logically unique. ■ The join is to one or more destination columns that are non-unique, and there are no join constraints or run-time search specifications applied. ■ Some cases exist for which the application logic may require denormalizing the relationship and viewing all the associations in the context of the parent record. Even if there are no duplicate associations, the same parent row repeats in the result set as associations to different child records in the intersection table. In this case, it may be preferable to not reject duplicates. You can inactivate this user property or modify its value. You can create new instances of this user property, but you should not create more than one instance for a business component.
Parent Object Type	Business Component
Functional Area	Query

DynHierarchy User Properties

This group of user properties is used exclusively with Global Accounts. Global accounts are hierarchies of accounts. These properties define relationships that control visibility in various Global Accounts views in the Accounts screen.

For information about creating account hierarchies, see *Applications Administration Guide*.

DynHierarchy Hierarchy Id Field

This user property specifies the field on the current business component that defines a join to account hierarchies. This relationship determines which records of the business component are visible in the flat list associated with a particular account hierarchy in Global Accounts views.

Value	The value for this user property is the name of a field on the current business component, not enclosed in quotes. For example, the value of this user property on the Global Account Action business component is Dynamic Hierarchy Id. The content of the Dynamic Hierarchy Id field on the Global Account Action business component is the Id of a record on the table that defines account hierarchies. Thus, a Global Account Action record is associated with the account hierarchy to which the activity's direct account belongs. The record appears in the flat list of activities for its parent account and for any ancestor account in the hierarchy.

Usage	Default values for this user property are provided for the business components that underlie subaccounts, contacts, activities, opportunities, and sales teams in Global Account views. This user property can be inactivated if you are not implementing global accounts. You can modify values for this user property, but do so only if you are well-justified. You cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Global account-associated subaccounts, activities, contacts, opportunities, and account teams

DynHierarchy Visibility Organization Id Field

This user property specifies the field on the current business component that defines the join between accounts and their organizations. This relationship determines which records are visible in the flat list of the business component in Global Accounts views for All Global Accounts visibility and All Global Accounts Across Organizations visibility.

Value	The value for this user property is the name of a field on the business component, not enclosed in quotes. For example, the value of this user property on the Global Account Contact business component is DynHierarchy Visibility Organization Id. The default value of the DynHierarchy Visibility Organization Id field is the alias for the join of accounts to organizations. Thus, a Global Account Contact record is associated with the organization to which its account is associated. When visibility is set to All Global Accounts, hierarchies display only accounts that have the same organization as the user's current position. Only the contact records in accounts in that same organization appear in the flat list of contacts for a hierarchy.
Usage	Default values for this user property are provided for the business components that underlie subaccounts, contacts, activities, opportunities, and sales teams in Global Account views. This user property can be inactivated if you are not implementing global accounts. You can modify values for this user property, but do so only if you are well-justified. You cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Global account-associated subaccounts, activities, contacts, opportunities, and account teams

DynHierarchy Visibility Position Id Field

This user property specifies the field on the current business component that specifies the join to positions. This relationship defines which records are visible in the flat list of the business component in Global Accounts views for My Global Accounts visibility.

Value	The value for this user property is the name of a field on the business component, not enclosed in quotes. For example, the value of this user property on the Global Account Opportunity business component is DynHierarchy Visibility Position Id. The default value of the DynHierarchy Visibility Position Id field is the alias for the join of positions, or team members, to accounts. Thus, a Global Account Opportunity record is associated with the team members on its account. When visibility is set to My Global Accounts, hierarchies display only those accounts for which the position of the user is on the account team. Only the opportunity records in accounts for which the current user is on the team appear in the flat list of opportunities for a hierarchy.
Usage	Default values for this user property are provided for the business components that underlie subaccounts, contacts, activities, opportunities, and sales teams in Global Account views. This user property can be inactivated if you are not implementing global accounts. You can modify values for this user property, but do so only if you are well-justified. You cannot create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Global account-associated subaccounts, activities, contacts, opportunities, and account teams

eAuto Enable Create Sales Step

This user property specifies whether to populate Opportunity Sales Steps for a Siebel eAutomotive application.

Value	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* has a value of Y or N indicating whether 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.

Value	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 8 contains the user properties for eGanttChart Busy Free Time Applet:

Table 8. 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 8. 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 the field in the X business component on which you can drill down. A corresponding Drilldown Object must be 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.
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 defined here, the Y business component is constrained by a link in a parent/child relationship.

Table 8. eGanttChart Busy Free Time Applet User Properties

Name	Value	Comments
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	SetGridBeginEndDateTime	Maps to an internal Z business component method to calculate the datetime range for the Z business component data. You should not modify this.
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.

Value	The value for the Email Activity Accepted Status Code user property must be a language-independent code listed in the EVENT_STATUS_LOV, for example: `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 or unprocessed outbound email activity.

Value	The value for the Email Activity New Status Code user property must be a language-independent code listed in the EVENT_STATUS_LOV, for example: `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.

Value	The value for the Email Activity Rejected Status Code user property must be a language-independent code listed in the EVENT_STATUS_LOV, for example: `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.

Value	The value for the Email Activity Sent Status Code user property must be a language-independent code listed in the EVENT_STATUS_LOV, for example: `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 to use Mail Manager to send emails.

Value	■ Y use Mail Manager to send emails
Usage	If the value is Y, then the business component uses Mail Manager to send 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

Employee Link

This user property allows you to restrict data visibility to the data associated with the user's login.

Value	The value of this user property is the name, not enclosed in quotes, of a defined link whose parent business component is Employee and whose child business component is the current business component—for example, Employee/My Competitor.
Usage	Typically, the business component on which you set this user property is used expressly to provide a "My" view that restricts data by login (username) instead of position. For example, the My Competitor business component exists primarily to provide the SI Com Tracked Competitors View in Siebel Briefings. This view is labeled as "My Tracked Competitors" in the UI, and it lists competitors that the user enters in his or her individual list. The Employee Link user property on the My Competitor business component is set to a value of Employee/My Competitor. Competitor records that are associated with the user by the Employee/My Competitor link are the only records listed in the view. You can inactivate this user property or modify its values. You should not create more than one instance of this user property for a business component.
Parent Object Type	Business Component
Functional Area	Access control

Enable Dispatch Board

This user property enables the Dispatch Board views based on the business component for which this user property is set.

For more information on the Dispatch Board, see *Siebel Field Service Guide*.

Value	TRUE or FALSE. A value of TRUE enables the Dispatch Board views based on this business component. A value of FALSE or unspecified provides no Dispatch Board.
Usage	You can modify this user property. You cannot inactivate or create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Activity

Encryption User Properties

Encryption of business component fields can be controlled using the following field user properties, which are described in subsequent topics:

- Encrypted
- Encrypt Key Field
- Encrypt Service Name
- Encrypt ReadOnly Field
- Encrypt Source Field (used with Display Mask Char)

For more information on setting up and upgrading encryption, see *Security Guide for Siebel Business Applications*.

For more information on encryption keys and how they are managed, see *Siebel System Administration Guide*.

A field is encrypted by setting the encryption flag, identifying the encryption service, and specifying the encryption key to be used. Siebel applications come preconfigured with two business services that you can use to encrypt data fields: the Advanced Encryption Standard (AES) Encryptor and the RC2 Encryptor, based on RSA encryption.

NOTE: You must run upgrade scripts to change the encryption of a field by any of the following: use the RSA or AES encryptor service on a field that was previously unencrypted or that was encrypted using the Standard Encryptor (no longer supported); use a stronger version of RC2 encryption than was previously used on the field. For more information, see the upgrade guide for the operating system you are using.

When encryption is turned on, data written to the field is in the encrypted format and data read from the field is 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.

For information about turning on field level encryption, see *Security Guide for Siebel Business Applications*.

You can turn off encryption on a field by setting the field's Encrypted user property to N.

NOTE: Credit Card Number fields are commonly encrypted. However, in the Order Entry -- Orders, Quote, and Agreements business components screens, it may be desirable to turn off the encryption in particular credit card number fields so that the user can see what was typed.

Encrypted

This user property allows you to specify whether a field is encrypted.

Value	The value of the Encrypted user property must be either Y or N.
Usage	Turn on encryption on the field by setting this user property to Y and by setting the Encrypt Service Name and Encrypt Key Field user properties. See also *Security Guide for Siebel Business Applications*. You can turn off encryption on the field by setting this user property to N.

Parent Object Type	Field
Functional Area	Encryption

Encrypt Key Field

This user property allows you to specify which encryption key to use.

Value	The value of this user property is the name of the field on the business component that contains the encryption key index.
Usage	The keyfile.bin file in the \Siebel_Root\Admin directory contains indexed encryption keys. The Encrypt Key Field user property specifies the field on the business component that contains the numbered encryption key index to use to decrypt the parent field. For example, in the Quote business component, the Credit Card Number field is an encrypted field that contains credit card numbers. The Credit Card Number Key Index field contains the index of the encryption key that is used to decrypt the Credit Card Number field. Thus on the Credit Card Number field the Encrypt Key Field user property is set with a value of Credit Card Number Key Index.
Parent Object Type	Field
Functional Area	Encryption

Encrypt Service Name

This user property allows you to specify the encryption service name.

Value	■ RC2 Encryptor ■ AES Encryptor
Usage	Set this user property on an encrypted field to specify which embedded encryption service to apply.
Parent Object Type	Field
Functional Area	Encryption

Encrypt ReadOnly Field

This field user property allows you to set an encrypted field to read-only if its decryption fails.

Value	The value of this user property is the name of a calculated field on the business component whose Calculated Value property is left blank.

Usage	The calculated field that is specified by this user property determines whether the data in the encrypted field is set to read-only. Preserving the data in read-only form may allow someone to recover it later without the data being further modified. The calculated field can assume the following values: ■ Y if decryption fails on the encrypted field. The encrypted field is automatically set to read-only. ■ N if decryption succeeds on the encrypted field. The encrypted field is editable. For example, in the Quote business component, the encrypted Credit Card Number field has the Encrypt ReadOnly Field user property set to the calculated field Credit Card Number - Read Only.
Parent Object Type	Field
Functional Area	Encryption

Encrypt Source Field

This field user property allows you to display a masked version of secure data, typically a credit card number or account number.

Value	The value of the Encrypt Source Field user property is the name of a field on the business component that contains the encrypted credit card number or account number.
Usage	This user property is used with the Display Mask Char user property to display only the last 4 digits of a credit card number or account number, such as xxxxxxxxxxxx9999. This user property is set on a separate calculated field that is displayed in the UI instead of the field containing the entire credit card number or account number. For example, in the Quote business component, the Credit Card Number field is the encrypted field that stores the credit card number. The Credit Card Number - Display field is a calculated field on which the following user properties are set: ■ Display Mask Char with a value of x ■ Encrypt Source Field with a value of Credit Card Number The applet field that displays the masked credit card number should reference the Credit Card Number - Display field. See also "Display Mask Char" on page 112.
Parent Object Type	Field
Functional Area	Encryption

Extended Quantity Field

This user property is defined in the line item business component of a quote (for example, Quote Item).

Value	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 Read Only Field: *fieldname*

This user property sets a specific field in a business component to be read-only.

Value	The value of this user property is the name of a field that contains a Boolean value.
Usage	The name of the Field Read Only Field user property must specify the name of a field (*fieldname*) in the business component. The value of Field Read Only Field is typically a second field, which is Boolean. When the field specified by the value evaluates to TRUE, the field specified by *fieldname* in the current record is set to read-only. For example, if Field Read Only Field: Sales Rep has a value of Calculated Primary Flag, then the Sales Rep field is set to read-only when Calculated Primary Flag is TRUE.
Parent Object Type	Business Component
Functional Area	Data Driven Access

FileMustExist

This user property allows you to specify whether the user can enter the name of a file to be provided later.

Value	The value for the FileMustExist user property must be either TRUE or FALSE.
Usage	This user property is typically set to TRUE, indicating that the file must already exist in order to add it as an attachment.
Parent Object Type	Business Component
Functional Area	Asset Management

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.

Value	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 inactivate or modify the values for this user property. See also "Setting Numbered Instances of a User Property" on page 73.
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.

Value	The value of the Forecast Analysis BC user property is a business component name (for example, Forecast 2000 – Forecast Item Detail Flat).
Usage	This user property is used on the Forecast Analysis Views. These views allow users to select multiple forecasts and see the aggregate summary records in the lower applet. This user property gives the name of the business component that is used in the lower applet. You can modify the values for this user property. However, you cannot inactivate or create new instances of this user property. **NOTE:** While you can change the value of this user property to another business component, it is recommended that you do not change its value unless you are creating new business components for forecasting.
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.

Value	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')` During forecast rollup, only the detail records that satisfy the search specification are rolled up into the summary record. 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. **NOTE:** While you can change the value of this user property, it is recommended that you do not change its value unless you intend to use forecasting in a different manner.
Parent Object Type	Business Component
Functional Area	Forecast

Group Visibility

This user property allows you to specify that "group + team" visibility is applied to the campaign.

Value	The value for the Group Visibility user property is either TRUE or FALSE.
Usage	When this user property is set to TRUE, "group + team" visibility is applied.
Parent Object Type	Business Component
Functional Area	Campaign

Group Visibility Only

This user property allows you to specify that only group visibility is applied to the campaign.

Value	The value for the Group Visibility Only user property is either TRUE or FALSE.

Usage	When this user property is set to TRUE, only group visibility is applied. If you set this user property to TRUE, you should set the Group Visibility user property to FALSE. You might also want to inactivate the Buscomp View Mode user property, so that it does not perform an inner join to the S_SRC_POSTN table.
Parent Object Type	Business Component
Functional Area	Campaign

Inner Join Extension Table *n*

For a business component based on the S_PARTY table, this user property specifies an extension table to S_PARTY for which the join to the extension table is an inner join.

Value	The value of this user property is the name of a table, not enclosed in quotes. The table specified should be an extension table of S_PARTY.
Usage	Many business components, among them organization, account, and position, are based on the Siebel Party Model and are used to configure access control and visibility. These business components are based on one or more inner joins to the base S_PARTY table. For a given business component, the Inner Join Extension Table user property specifies a table that is inner-joined to the S_PARTY table. Inner Join Extension Table must be used if you create a new business component that is based on the Siebel Party Model. You may need to create such a business component in order to configure visibility for a group that does not fit any of the existing Siebel Party Model business components. You would then use this user property to define one or more tables that are implicitly joined to the S_PARTY base table. This user property can also be specified with a number appended. For a given business component, the value associated with the property that has the lowest number is the primary extension table. For example, for the Employee business component, Inner Join Extension Table 1 has value S_CONTACT, Inner Join Extension Table 2 has value S_USER, and Inner Join Extension Table 3 has value S_EMP_PER. Three extension tables are specified, of which S_CONTACT is the primary extension table. You must not inactivate this user property or modify its value. You can create new instances of this user property, if necessary. **CAUTION:** To understand the implications of using the S_PARTY table to define parties other than those provided with Siebel Business Applications, please see details on the Party model in *Configuring Siebel Business Applications*. See also "Setting Numbered Instances of a User Property" on page 73.
Parent Object Type	Business Component

Functional Area	Party-related business components

Logical Message Type

This user property allows you to specify the logical message type corresponding to the IDOC.

Value	The value for the UserPropName user property must be a logical message type corresponding to the IDOC (for example, DEBMAS or MATMAS).
Usage	When used with the SAP product, the Logical Message Type 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.
Parent Object Type	Integration Object
Functional Area	

Maintain Master Account

This user property allows you to specify whether the Master Account Id in an Account hierarchy should be maintained.

Value	■ Y Indicates that the Master Account Id is maintained. ■ N Indicates that the Master Account Id is not maintained.
Usage	If you change the value of this user property from its default (Y), the Master Account Id in the Account hierarchy is not maintained. You cannot inactivate or create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Account

Manager List Mode

This user property allows you to specify whether records related to only subordinate primary team members are displayed or records related to all team members reporting to a manager are displayed in a manager view.

Value	The value for the Manager List Mode user property must be either Primary or Team.

Usage

When the Manager List Mode user property is set to Primary (the default value), the records for the subordinate primaries are displayed in the manager view.

When the Manager List Mode user property is set to Team, the records for all people who report to a given manager are visible in a manager view, rather than just the primaries. When set to Team, it performs a subquery for the My Team's views to retrieve and display the accounts, opportunities, and so on, for all members of the team, not just the primary.

In Team mode, performance is slower but yields more data.

When you set the Manager List Mode user property to Team, 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.

Parent Object Type Business Component

Functional Area eBusiness Application Integration Test

Master Account Field

This user property specifies the name of the field that stores the Master Account Id.

Value 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

This user property allows you to specify that the integration component is a MVG business component.

Value The value for the MVG user property must be either Y or N.

Usage When used with the SAP product, you can set the value of the MVG user property to Y to indicate that the integration component is a MVG business component.

Parent Object Type	Integration Component
Functional Area	eBusiness Application Integration Test

MVG Set Primary Restricted: *visibility_mvlink_name*

This user property allows you to disable the restriction that only Siebel Administrators and Managers have the ability to change the Primary team member on opportunities, accounts, and contacts.

Value

The value for the MVG Set Primary Restricted: *visibility_mvlink_name* user property is FALSE.

Usage

The *visibility_mvlink_name* specified in the name of the user property indicates the name of the multivalue link in the BusComp View Mode child object of the business component for which you want to allow or restrict setting the primary.

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. Setting this user property to FALSE allows the Primary team member to be altered by someone other than the Manager or Siebel Administrator.

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.

Parent Object Type	Business Component
Functional Area	Access control

Named Method *n*

This user property allows you to invoke a method from a business component or business service, or set a field value.

Value

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 9 on page 136.

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 9 on page 136.

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”).

See also “Setting Numbered Instances of a User Property” on page 73.

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, Named Method 1, then Named Method 2, and so on). If there is only one such user property, then no number is required.

Parent Object Type

Business Component

Functional Area

CSSBCBase

Table 9. 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 that the revenues returned by the search are within the Forecast Date range.

Value	The value of the Named Search: Forecast Series Date Range must be a valid search specification.

Usage	This user property is used when querying the revenue records to pull into the current forecast. The default is to pull in all records in the range specified by the forecast. 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. **NOTE:** If you inactivate the Named Search: Forecast Series Date Range user property, you can make sure that 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.
Parent Object Type	Business Component
Functional Area	Forecast

No Change Field *n*

This user property disallows changing a field’s value after the record is committed.

Value	The value of this user property must be the name of a field in the business component, not enclosed in quotes.
Usage	This property can be specified with or without the numeric suffix. You should append the numeric suffix to differentiate between multiple instances on a business component. For example, add No Change Field 1 and No Change Field 2 user properties to a business component to specify two different fields which cannot be changed after a record is committed. You can inactivate or modify the values for this user property. You can also create new instances of this user property as needed. See also “Setting Numbered Instances of a User Property” on page 73.
Parent Object Type	Business Component
Functional Area	Various

No Clear Field *n*

This user property disallows setting a field's value to NULL.

Value	The value of this user property must be the name of a field in the business component, not enclosed in quotes.
Usage	This property can be specified with or without the numeric suffix. You should append the numeric suffix to differentiate between multiple instances on a business component. For example, add No Clear Field 1 and No Clear Field 2 user properties to a business component to specify two different fields whose values cannot be set to NULL. You can inactivate or modify the values for this user property. You can also create new instances of this user property as needed. See also "Setting Numbered Instances of a User Property" on page 73.
Parent Object Type	Business Component
Functional Area	Various

NoDataHide

This user property hides the applet when it contains no data.

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
Functional Area	

NoDelete

This user property allows you to restrict the Siebel eAI connector from performing deletes on the corresponding business component.

Value	The value for the NoDelete user property is Y.
Usage	Setting this user property to Y instructs the Siebel eAI connector to *not* perform deletes on the business component that the integration component represents.
Parent Object Type	Integration Component
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.

Value	The value of this user property must be the name of a field in the business component.
Usage	When you specify a field in this user property, the business component does not allow records to be deleted that have a value of Y in the specified field. For example, a record on the Contact business component cannot be deleted if NoDelete Field has a value of Protect Internal Employee Flag and the value of the Protect Internal Employee Flag field in the record is Y. You can inactivate or modify the values for this user property. You can also create new instances of this user property as needed, but you cannot create more than one instance for a business component.
Parent Object Type	Business Component
Functional Area	CSSBCBase

NoInsert

This user property allows you to restrict the Siebel eAI connector from performing deletes on the corresponding business component.

Value	The value for the NoInsert user property is Y.
Usage	Setting this user property to Y instructs the Siebel eAI connector to *not* perform inserts on the business component that the integration component represents.
Parent Object Type	Integration Component
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.

Value	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 only displays 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

This user property allows you to restrict the Siebel eAI connector from performing updates on the corresponding field.

Value	The value for the NoUpdate user property is Y.
Usage	Setting this user property to Y instructs the Siebel eAI connector to *not* perform updates on the field that the integration component field represents.
Parent Object Type	Integration Component Field
Functional Area	

On Condition Set Field Value

This user property allows you to specify the value of a field to be set under a specified condition.

Value	The value for the On Condition Set Field Value user property consists of three quoted parameters, separated by a comma and a space, as follows: "*Condition*", "*FieldName*", "*FieldValue*" *Condition* specifies the condition to be evaluated. *FieldName* specifies the name of the field on the business component. *FieldValue* specifies the value. For example: "[Primary Held Position Id] is not null and [Primary Held Position Id] <> ""No Match Row Id""", "Employee Flag", "Y"
Usage	When the specified condition evaluates to TRUE, the specified field (*FieldName*) is set to the specified value (*FieldValue*). In the example above, the Employee Flag field is set to Y when the condition is TRUE. You can modify the value for this user property and create new instances of it. You can also inactivate this user property.
Parent Object Type	Business Component
Functional Area	CSSBCUser

On Field Update Invoke *n*

This user property allows you to invoke the specified business component method when a field is updated.

Value	The value of the On Field Update Invoke user property 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. See also "Setting Numbered Instances of a User Property" on page 73.

Parent Object Type	Business Component
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.

Value

The value of the On Field Update Set user property consists of three quoted parameters separated by a comma and a space, as follows:

"*FieldToCheck*", "*FieldToSet*", "*Value*", "*Condition*"

where *Value* and *Condition* are optional parameters.

The field *FieldToSet* is set to *Value* when the field *FieldToCheck* is updated. If the *Value* parameter is not defined, *FieldToSet* is set to the value of *FieldToCheck*. If the *Condition* parameter is defined, then *FieldToSet* is updated only if *Condition* evaluates to TRUE.

Use the following guidelines when using this user property:

- Do not use this user property to set a multi-value or calculated field. That is, if *FieldToSet* is a multi-value or calculated field, it will not update when *FieldToCheck* is updated.
- Do not define *FieldToCheck* as a field on a multi-value group. This user property does not recognize changes to a multi-value group field, including changing the primary field by changing the primary record of a multi-value group.

  For example, if On Field Update Set has the value
  `"Primary Address Id", "Email Address", "my@siebel.com"`
  on the Contact business component, it fails to update `Email Address` when the primary on the multi-value group `Street Address` is changed.

Usage

The *Value* parameter may be an expression. In the following example, the Done field is set using an expression when the Done Flag field is updated:

```
"Done Flag", "Done", "IIF ([Done Flag] = "Y", Today (), "")"
```

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 the value of the field.

```
"Agreement Start Date", "Name", "ToChar([Agreement Start Date]) +
[Agreement Type]"
```

The following example shows how the *Condition* parameter is used. The Revenue field of the Opportunity business component is set when the Primary Revenue Amount field is updated, but only when the IsParentBCRevn field has a value of N:

```
"Primary Revenue Amount", "Revenue", "[Primary Revenue Amount]",
"[IsParentBCRevn] = 'N'"
```

Various address business components, such as Business Address, populate their Address Name field with a concatenation of street address, city, and state. This field is updated, or not, by using a few On Update Field Set instances and the value of a calculated field whenever the street address, city, or state are updated. For example, when the city is updated, an On Update Field Set user property with the following value is used:

```
"City", "Address Name",  "IIF( [Address Name Locked Flag] = ""N"",
[Calculated Address Name], [Address Name])"
```

Similar numbered instances of the user property are used to update the Address Name field when the street address or state are updated.

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.

See also “Setting Numbered Instances of a User Property” on page 73.

Parent Object Type Business Component

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.

Value

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.

Value	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.

Value	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.

Value	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

This user property allows you to specify a TRUE/FALSE test on a 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.

Value	The value for the Parent Read Only Field user property consists of a pair of period-separated parameters, as follows: *buscompname.fieldname* *buscompname* specifies the name of the business component, and *fieldname* specifies the name of the field in the business component.

Usage

When the value of the specified field evaluates to TRUE, the current business component becomes read-only.

The business component to be conditionally restricted is the one to which you add the user property as a child object definition. The business component containing the test field must be a parent or grandparent of the restricted business component by way of a link or series of link relationships.

Parent Read Only Field is used primarily to restrict the detail records in a multi-value group. It could also be used to restrict the detail records in a master/detail view, but in that case you need to make sure that the restricted business component is not also used in the context of some other business object than the intended one.

NOTE: When using the Parent Read Only Field user property, the test field must have its Link Specification property set to TRUE. Otherwise, the dynamic read-only functionality does not work. However, if the child record is displayed in the multi-value field in the parent business component, it is not necessary to have the Link Specification property of the field set to TRUE.

For more information on the Parent Read Only Field user property, see *Configuring Siebel Business Applications*.

Parent Object Type

Business Component

Functional Area

Data Driven Access

Picklist Pre Default Field *n*

Within a view based on a parent business component, such as Action, the user may be able to create a new record of a child business component, such as Opportunity, through a picklist for the child business component. The Picklist Pre Default Field user property is used to default fields on the new record of the child business component to field values from the parent business component record.

Value

"*field*", "'*buscomp1.field1*','*buscomp2.field2*',' . . .'"

where

- *field* is a field on the current business component
- *buscompn.fieldn* is a field name on a parent business component

NOTE: The list of *buscompn.fieldn* entries is enclosed in double quotes, and each *buscompn.fieldn* entry is enclosed in single quotes.

Usage	This property can be specified with or without the numeric suffix. You should append the numeric suffix to differentiate between multiple instances on a business component. For example, add Picklist Pre Default Field 1 and Picklist Pre Default Field 2 user properties to a business component to specify two different fields that should assume default field values from the parent business component. For example, a new opportunity can be created through an Opportunity picklist in each of the following applets: ■ Activity Form Applet, based on the Action business component ■ Comm Outbound Item Form Applet, based on the Comm Outbound Email business component When a new opportunity is created from the picklist in either context, the opportunity's Account and Account Id fields can be defaulted to the corresponding field values on the parent record by adding the following user properties to the Opportunity business component: ■ Picklist Pre Default Field 1 with value "Account", "'Action.Account Name', 'Comm Outbound Email.Account Name'" ■ Picklist Pre Default Field 2 with value "Account Id", "'Action.Account Id', 'Comm Outbound Email.Account Id'" You can inactivate or modify the values for this user property. You can also create new instances of this user property as needed. See also "Setting Numbered Instances of a User Property" on page 73.
Parent Object Type	Business Component
Functional Area	Picklist generation

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.

Value	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.

Value	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 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.

Value	■ 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.
Parent Object Type	Business Component
Functional Area	Service Request

PostMainViewData

This control user property is used to prevent unsaved data from being lost when a control changes the current record or the active applet in the main view.

Value	TRUE or FALSE
Usage	This control user property is intended to prevent data loss when a control invokes a method that changes the main view before new data in the current record is saved. ■ If set to TRUE in a high interactivity (HI) view, and if the current record contains modified data, this user property causes the current record to be saved before the control changes the focus. ■ If set to TRUE in a standard interactivity (SI) view, and if the current record contains modified data, this user property causes a message to display that warns that unsaved data may be lost. **NOTE:** For SI applications, the warning message does not display unless you also add EnableSIDataLossWarning = TRUE to the [SWE] section of the application's .cfg file. ■ If set to FALSE or if this user property is not set on the control, then the control's method is invoked and no unsaved data is written to the current record. For example, if the main view is the All Service Requests view, you can click the Search button before the current record is saved. After you execute a search, subsequently clicking the Attach button for a record in the Results list drills down into the active record, thereby changing the applet in the main view. The new data in the unsaved service request record can then be lost. The PostMainViewData user property should be set on controls such as the Attach button. You can inactivate and modify the values for this user property, and you can create a new instance of this user property if it is not already defined on the control.
Parent Object Type	Control
Functional Area	Various

Primary Position Modification

This user property allows you to specify whether Sales Method for an Opportunity can be modified only by the primary position.

Value	■ 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.
Parent Object Type	Business Component
Functional Area	Opportunity

Private Activity Search Spec

This user property provides a search specification to apply to non-Calendar activities. It is typically used to restrict the return activities to those for which the Private flag is not set and those for which the logged-in user is the primary owner. However, it can be used to provide other restrictions to the results set.

Value	The value of this user property is a valid search specification. A typical example is `[Private] = 'N' OR [Private] IS NULL OR [Primary Owner Id] = LoginId ()`
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	Activity

Protect Seed Data

This user property allows you to prevent seed data records of a business component from being modified.

Value

Y or N, not enclosed in quotes.

If this user property is set to Y, then seed data records of the business component cannot be deleted or modified when Siebel Business Applications are launched by the standard means.

If this user property is not defined on the business component or its value is N, then seed data records of the business component can be modified or deleted when Siebel Business Applications are launched by the standard means.

The Protect Seed Data user property can be overridden, and modification of seed data allowed, by launching Siebel Business Applications with the /editseeddata parameter. The /editseeddata parameter can be activated in one of the following ways:

- Append /editseeddata to the UNIX command line that launches the application.
- Append /editseeddata to the command line that launches the application, as defined in the properties dialog box for the application’s shortcut in Windows.

Usage	For example, add the Protect Seed Data user property with value Y to the Responsibility business component. Then, seed data responsibility records—those with record numbers starting with zero (0)—cannot be deleted or modified. You cannot inactivate this user property or modify its values. You can create new instances of this property.
Parent Object Type	Business Component
Functional Area	Seed data

RBFields

This user property allows you to specify the fields in the Room Block business component that represent types of rooms, and the sum of the values in the associated fields.

Value	The value for the RBFields user property consists of the names of fields in the Room Block business component, separated by commas, and terminated by an integer that indicates the sum of the values for the specified fields. For example, RB Single, RB Double, RB Triple, RB Quad, 100
Usage	The integer that indicates the sum of the values of the fields is typically 100. You can modify the value for this user property and create new instances of it. However, you cannot inactivate this user property.
Parent Object Type	Business Component
Functional Area	CSSBCFINOppty

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.

The following recipient communications user properties are described in subsequent topics:

- "Recipient First Name Field" on page 152
- "Recipient Last Name Field" on page 152
- "Recipient Email Address Field" on page 153
- "Recipient Fax Address Field" on page 153
- "Recipient Preferred Medium Field" on page 153
- "Recipient Id Field n" on page 153

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, and 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, and 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 on driver parameters for Internet SMTP/POP3 Server, see *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

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. See also "Setting Numbered Instances of a User Property" on page 73.
Functional Area	Send Email, fax

Recursive Link

The Deep Copy, Deep Delete, and Update Foreign Key Field user properties are used to copy or delete records of child business components when a record of the current (parent) business component is copied or deleted. If the parent and child are the same business component, then the link that defines the parent/child relationship must be specified by the Recursive Link user property.

See also

"Deep Copying and Deleting" on page 101

"Update Parent BC" on page 170

For information about the Link object type, see *Configuring Siebel Business Applications*.

Value	The value of this business component user property must be the name of an existing link between the current (parent) business component and itself.
Usage	For example, to delete catalog subcategories when a category is deleted, you could use Deep Delete with the Recursive Link user property set to Catalog Category/Catalog Category. **NOTE:** The convention for naming links is *parent business component/child business component*, but a link name does not have to follow this convention. Thus the name of a link specified by Recursive Link may have different names on either side of the slash; for example, Action/Action - Deep. The requirement that must be met is that the parent and child business components of the link must be the same business component. You can inactivate or modify the values for this user property. You can also create new instances of this user property as needed, but you should not create more than one instance of this user property on a business component.
Parent Object Type	Business Component
Functional Area	Copying and deleting records

Remote Source

This user property allows you to specify an external data source used by the business service.

Value	The value for the UserPropName user property is the name of the external data source, for example: `DSN=EXCELABCCust`
Usage	You can inactivate and modify the value for this user property.
Parent Object Type	Business Component
Functional Area	

Required

This user property allows you to make the parent field a required field under certain conditions.

Value	The value of the Required user property is an expression.
Usage	You specify the condition by defining a calculated expression for the value of the user property. When the expression evaluates to Y, the field is required.
Parent Object Type	Field
Functional Area	CSSBCBase 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.

Value

The value for the Required Position MVField user property uses the following syntax:

```
"[Employee Flag]", "[Position MVField]"
```

Quotes are required.

- [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

This user property allows you to specify whether to create a response type “Call Back” when a new email or Web offer is created.

Value	The value for the Response Type Call Back user property is Y.
Usage	If this user property is set to Y, it creates a response type “Call Back” when you create a new email or Web offer. This user property is defined for Offer, Email Offer, and Web Offer business components.
Parent Object Type	Business Component
Functional Area	Offer, Email Offer, and Web Offer business components

Response Type More Info

This user property allows you to specify whether to create a response type “More Info” when a new email or Web offer is created.

Value	The value for the Response Type More Info user property is Y.
Usage	If this user property is set to Y, it creates a response type “More Info” when you create a new email or Web offer. This user property is defined for Offer, Email Offer, and Web Offer business components.
Parent Object Type	Business Component
Functional Area	Offer, Email Offer, and Web Offer business components

Response Type Unsubscribe

This user property allows you to specify whether to create a response type “Unsubscribe” when a new email or Web offer is created.

Value	The value for the Response Type Unsubscribe user property is Y.
Usage	If this user property is set to Y, it creates a response type “Unsubscribe” when you create a new email or Web offer. This user property is defined for Offer, Email Offer, and Web Offer business components.
Parent Object Type	Business Component
Functional Area	Offer, Email Offer, and Web Offer business components

Revenue Aggregation Field *n*

This user property specifies a field in the business component that is rolled up into the summary record from the detail records.

Value	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	Details and summaries are on the same business component. A summary record sums one or more fields of the detail records for the summary date range. 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. See also “Setting Numbered Instances of a User Property” on page 73.
Parent Object Type	Business Component
Functional Area	Forecast

Revenue Associate List

This user property allows you to specify whether to use an associate list when a detail is created by clicking the New button in a Forecast 2000 -- Forecast Item business component.

Value	The value of the Revenue Associate List user property is either Y or N.
Usage	This user property is used during detail creation. When this user property is set to Y, an associate list is used when a detail is created by clicking the New button. When the New button is clicked, a pop-up list displays the Revenues that fit the Associate Search Spec for the current forecast Series. The user must associate an existing Revenue to the forecast, rather than adding free text data. You can modify the value for this user property. However, you cannot inactivate or create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Forecast

Revenue Field Map: *fieldname*

This user property allows you to specify a field to be copied from the Revenue business component to the Forecast 2000 -- Forecast Item (or Forecast 2000 -- Forecast Item Detail) business component during detail creation.

Value	The value of the Revenue Field Map: *fieldname* user property is the name of a Revenue business component field.
Usage	The *fieldname* specified in the name of the user property is a Forecast 2000 -- Forecast Item field. This user property is used during detail creation. When a detail is created in the current Forecast Item business component, the value of the field in the Revenue business component (specified in the value of the user property) is copied into the field in the Forecast Item business component specified in the name of the user property. For example, when the user property with the following values is defined for the Forecast 2000 -- Forecast Item business component, the value for the `Organization Id` field is copied from the `Sales Rep Organization Id` field in the Revenue business component when a detail is created. Name: Revenue Field Map: Organization Id Value: Sales Rep Organization Id You can inactivate and modify the values for this user property. You can also create new instances of this user property, as needed, for each field that needs to be copied.

Parent Object Type	Business Component
Functional Area	Forecast

Revision Copy Field *n*

When the Revise method is used to create a new record as a copy of an existing record, this user property causes the value in a particular field of the existing record to be copied to the new record.

The Revision Copy Field user property can be used in conjunction with the Revision Field user property to produce numbered revisions of such things as quotes, orders, agreements, and so on.

See also "Revision Field" on page 159 and "Revise" on page 26.

Value	The value of this user property must be the name of a field in the business component.
Usage	This property can be specified with or without the numeric suffix. You should append the numeric suffix to differentiate between multiple instances on a business component. A typical example of the use of the Revision Copy Field and Revision Field user properties is to create an updated revision of a quote. On the Quote business component, add several Revision Copy Field user properties, including Revision Copy Field 1 with value Quote Number and Revision Copy Field 2 with value Credit Card Number. Add the Revision Field user property with value Revision. When a new quote record is created by using the Revise method to copy an existing record, the Revision field contains the next number in the revision sequence (one more than the maximum value in the existing records). The Quote Number and Credit Card Number fields contain the same values as in the copied existing record. You can inactivate or modify the values for this user property. You can also create new instances of this user property as needed. See also "Setting Numbered Instances of a User Property" on page 73.
Parent Object Type	Business Component
Functional Area	Revisions of various kinds of records

Revision Field

When the Revise method is used to create a new record as a copy of an existing record, this user property causes a field to prepopulate with the next value in an integer sequence. It is typically used to generate a version, or revision, number.

The Revision Copy Field user property can be used in conjunction with the Revision Field user property to produce numbered revisions of such things as quotes, orders, agreements, and so on.

See also "Revision Copy Field n" on page 159 and "Revise" on page 26.

Value	The value of this user property must be the name of a field in the business component.
Usage	A typical example of the use of the Revision Copy Field and Revision Field user properties is to create an updated revision of a quote. On the Quote business component, add several Revision Copy Field user properties, including Revision Copy Field 1 with value Quote Number and Revision Copy Field 2 with value Credit Card Number. Add the Revision Field user property with value Revision. When a new quote record is created by using the Revise method to copy an existing record, the Revision field contains the next number in the revision sequence (one more than the maximum value in the existing records). The Quote Number and Credit Card Number fields contain the same values as in the copied existing record. 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	Revisions of various kinds of records

Sequence Field

This user property allows you to define a sequence field for a business component.

Value	The value for the Sequence Field user property must be the name of the field (typically Line Number or Sequence Number) in the business component that corresponds to the sequence number column in the underlying table.
Usage	This user property is used to configure 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. For new record and copy record events, this user property specifies a field on the business component whose value is a number in a sequence that is auto-generated. A sequence business component must also be defined in the business object. **NOTE:** When defining a Sequence Field user property, set the Insert Position property to LAST for the applets that display records from the numbered detail business component. Leaving the Insert Position property blank can cause unexpected behavior in the line numbers generated in the applet. **NOTE:** Configuring a sequence field on a business component requires several tasks to be completed. For detailed information on creating sequence fields, see *Configuring Siebel Business Applications*. For information on how a new record with a sequenced field is numbered, see also "Sequence Use Max" on page 161.

Parent Object Type	Business Component
Functional Area	Record sequencing

Sequence Use Max

This user property allows you to specify whether to generate sequence numbers for the business component based on the maximum sequence number.

See also "Sequence Field" on page 160.

Value

- A value of Y causes the sequence number of a new or copied record to be generated as the maximum of the existing sequence numbers + 1.
- If this user property is not defined on the business component or if this user property has a value of N, then the sequence number of a new or copied record is generated as the sequence number of the current record + 1. Other records are renumbered, if necessary.

Usage

This user property determines whether the sequence number generated is based upon the current record position or the maximum sequence number.

For example, to define a field Line Number as a sequence field for which new and copied records are assigned a the maximum of the existing sequence values + 1, you would have to add the following user properties to the business component:

- Sequence Field with value Line Number
- Sequence Use Max with value Y

NOTE: Configuring a sequence field on a business component requires several tasks to be completed. For detailed information on creating sequence fields, see *Configuring Siebel Business Applications*.

Parent Object Type	Business Component
Functional Area	Record sequencing

Service Name

This user property allows you to specify a business service that is used by a virtual business component.

Value	The value for the Service Name user property must be the name of the business service.
Usage	
Parent Object Type	Business Component
Functional Area	

Service Parameters

This user property allows you to specify parameters for a business service that is used by the business component.

Value	The value for the Service Parameters user property consists of one or more *name=value* pairs delimited by semicolons. `ParamName1=ParamValue2;ParamName2=ParamValue2;...` *ParamName* specifies the name of the parameter, *ParamValue* specifies the value for the parameter that is passed to the business service (for example, DLLName=VirtualBusCompODBC.dll).
Usage	This user property names a text string of parameters parsed typically by the Pre_Invoke method and used by the virtual business component. For more information about business services in workflow processes, see the *Siebel Business Process Designer Administration Guide*.
Parent Object Type	Business Component
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 be assigned to this Primary Sales Rep.

Value	■ 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.

Value	■ 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

This user property allows you to specify whether to hide an employee's home phone number in the Contact List and Employee List views.

Value	The value for the Share Home Phone Flag Field user property is either Y or N.
Usage	When this user property is set to Y, the employee's home phone number is hidden in the Contact List and Employee List views. 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. 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

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.

Value	The value of the Show Required *n* 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. See also "Setting Numbered Instances of a User Property" on page 73.
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.

Value	■ If Y, then if a forecast exists for the current user for the current date, then the date is not available for the user to pick. ■ If Y, then dates are not available to pick for which a forecast already exists for the user. ■ If N, then there is no restriction on picking the Forecast Date.
Usage	The Skip Existing Forecast Series Date user property is used for the Forecast Series Date picklist on the Forecast screen. If the value is Y, only the forecast dates that have not yet been used in a forecast are displayed in the picklist. You can modify the value for this user property. However, you cannot inactivate or create new instances of this user property.
Parent Object Type	Business Component
Functional Area	Forecast

SleepTime

This user property allows you to specify the time out interval, in seconds, on receive requests.

Value	The value for the SleepTime user property must be an integer.
Usage	This user property sets the time out interval, in seconds, on receive requests. The default value is 20 seconds. **CAUTION:** 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.
Parent Object Type	Business Service
Functional Area	eBusiness Application Integration Business Services

Sort Field Map *n*

Several opportunity-related business components, such as Opportunity and Global Account Opportunity, have a one-to-many relationship with the Revenue business component. These relationships enable master/detail views that display the revenue records associated with an opportunity.

Opportunities typically have several multivalue fields that reference fields in the Revenue business component. For example, the Revenue and Close Date fields on the Opportunity business component reference the Revenue and Date fields on the Revenue business component, respectively.

Existing predefined queries on opportunities may include sort specifications based on one or more of these multivalue revenue fields. The Sort Field Map user property is used at run time to redirect such sorts on revenue multivalue fields to corresponding single-value fields on the opportunity. For example, a sort specification on the Revenue multivalue field on the Opportunity business component is typically redirected to the Primary Revenue Amount field, which is the revenue amount for the opportunity's primary revenue record.

NOTE: It is unlikely that you need to use this user property to redirect new predefined queries. Instead, write the sort specification to reference a single-value field, as described in the Usage section for this topic.

For information on sort specifications in predefined queries, see "Sorting Through Predefined Queries" on page 441.

For information on the primary revenue record for an opportunity, see *Applications Administration Guide*.

Value	"*field*", "*redirect field*" where ■ *field* is the name of a multivalue field on the current opportunity-related business component, such as Close Date or Revenue, that maps to a field on the Revenue business component. ■ *redirect field* is the name of the single-value field on the current opportunity-related business component that maps to the same revenue field on the primary revenue record for the opportunity—for example, Primary Revenue Close Date or Primary Revenue Amount.
Usage	This property can be specified with or without the numeric suffix. However, typically several instances are added to specify redirection of several revenue fields, so the numeric suffix differentiates the instances. For example, define the following user properties on the Opportunity business component, as well as similar ones for other revenue fields: ■ Sort Field Map 1 with value “Revenue”,“Primary Revenue Amount” ■ Sort Field Map 2 with value “Close Date”,“Primary Revenue Close Date” For the sort specification 'Opportunity'.Sort = "Revenue (Descending)"in a predefined query, the result set would be sorted in descending order of the revenue amounts in the primary revenue records of the opportunities in the result set. Similarly, the sort specification 'Opportunity'.Sort = "Close Date "in a predefined query would base the sort on the revenue close dates in the primary revenue records. **NOTE:** For new predefined queries, do not set sort specifications to reference multivalue fields. Instead of necessitating the redirection by the Sort Field Map user property, define the sort specification with the corresponding Primary Revenue field initially. For example, define the preceding sort specification as 'Opportunity'.Sort = "Primary Revenue Close Date". You should not inactivate or modify the values for this user property. You can create new instances of this user property. See also “Setting Numbered Instances of a User Property” on page 73.
Parent Object Type	Business Component
Functional Area	Predefined queries on opportunity-related business components

State Model

This user property allows you to make additional business components accessible to the State Model business component Multi-Value Group applet.

Value The value for the State Model user property is either Y or N.

Usage Setting the State Model user property to Y adds the business component 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 Siebel Tools, go to the business component (for example, Account) that you want to add to the state model 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. All of the business components you added in Siebel Tools should appear in the business component Name Multi-Value Group.
6. When you select the appropriate one and navigate to Field Name, the MVG applet that pops up includes 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 can create the State Model rules that Siebel Workflow enforces.

For more information about the State Model, see the *Applications Administration Guide.*

Parent Object Type Business Component

Functional Area Siebel Workflow

TargetProp *n*

This user property allows you to specify a target list for a business component.

NOTE: For a complete description of how to configure a target list, see the “Global Target List Management” section of the *Applications Administration Guide*.

Value

The value for the TargetProp user property consists of three quoted parameters separated by a comma and a space, as follows:

"*EntityDisplayName*", "*MVFName*", "*ListCategory*"

EntityDisplayName is a Language-Independent Code value defined for the SLM_FIELD_DISPLAY LOV. *MVFName* is the name of the multi-value field that stores the target list. *ListCategory* is one of the display values defined for the SLM_LST_CATEGORY LOV (Account, Contact, Employee, Position, or Prospect).

For example:

"Accounts", "List Mgmt List Id", "Accounts"

Usage

You can create new instances of this user property, as needed, for each target list. If you have more than one instance of this user property in a business component, they are executed sequentially by number (for example, TargetProp 1, then TargetProp 2, and so on). If there is only one such user property, then no number is required.

You can also inactivate or modify the value for this user property.

See also “Setting Numbered Instances of a User Property” on page 73.

Parent Object Type

Business Component

Functional Area

CSSBCAccountSIS

Text Length Override

This user property allows you to specify that the text length of the field, rather than that of the database column, defines the maximum field length.

Value

The value for the Text Length Override user property is either TRUE or FALSE.

Usage

When this user property is set to TRUE, the Text Length property of the field determines the maximum field length. If this user property is not defined, the size of the database column determines the maximum field length. Use only for Text type fields.

This user property replaces the Field Length user property in older versions of Siebel applications.

Parent Object Type

Field

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.

Value	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.

Value	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.

Value	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 is 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.

Value	■ 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 are updated are specified using the Update Status To Synchronized Types user property.

Value	■ 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 have their Status field changed to Synchronized during handheld synchronization.

Value

- ,*type1*,*type2*, The Status of *type1* and *type2* Activities are 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,
```

then the Status fields for Account Call, Professional Call, and Attendee Call are 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

UseExistsForSubQuery

When a user enters an EXISTS search specification, Siebel applications by default generate SQL that contains an IN clause. The UseExistsForSubQuery business component user property is used to cause the EXISTS search specification on the business component to generate an SQL EXISTS clause instead.

Value	TRUE or FALSE If set to TRUE, the SQL EXISTS clause is used when the user enters an EXISTS search specification. Is set to FALSE or if the user property is not defined on the business component, the SQL IN clause is used when the user enters an EXISTS search specification.
Usage	In some cases, use of the SQL EXISTS clause instead of the IN clause can improve query performance, notably on IBM DB2 390. You can inactivate this user property or modify its value. You can create new instances of this user property, but you should not create more than one instance for a business component.
Parent Object Type	Field
Functional Area	Search

Use Literals For Like

This business component field user property allows you to specify that literals, instead of bind variables, be generated as criteria for LIKE predicates in the SQL for queries on the field.

Value	TRUE or FALSE. If the value of this user property is TRUE, then literals, instead of bind variables, are generated as criteria for LIKE predicates in the SQL for queries on the field. If the value of this user property is FALSE or the user property is not defined on a field, then bind variables are used in criteria in the SQL for queries on the field.

Usage

Use of literals in certain queries with LIKE predicates can improve the speed of those queries on DB2 UDB by providing the database optimizer with better information with which to choose appropriate indexes for the query.

A predicate of the form “LIKE ?” is generated whenever a wildcard (* or ?) is included in a search string. A wildcard may be included in a search string in either of the following ways:

- The user includes the wildcard explicitly in the search string; for example “ABC*” or “*ABC?”.
- A trailing wildcard (*) is automatically appended to a search string when all of the following apply:
  - The search string does not contain any wildcards, such as just “ABC”.
  - The search string is not preceded with the = sign.
  - The AutomaticTrailingWildcards parameter in the SWE section of the application CFG file (such as uagent.cfg) is set to TRUE or is not present.

NOTE: Adding the AutomaticTrailingWildcards parameter with a value of FALSE in the SWE section of the application CFG file eliminates automatic trailing wildcards and the LIKE predicates that are generated as a result. Typically, implementations should have the AutomaticTrailingWildcards parameter set to FALSE, thereby removing much of the need for the Use Literals for Like user property.

When using a bind variable, criteria containing wildcards are treated as LIKE ? in the SQL. The Use Literals for Like user property provides a further conversion of the SQL to, for example, LIKE “ABC%”, thus specifying the exact location of wildcards with the % sign.

Use Literals for Like can provide greater efficiency than using bind variables for searches on some criteria, but it does not improve performance for other criteria. For example:

- Searches are typically more efficient for criteria with wildcards in trailing positions only, such as “ABC*” because an appropriate index is more easily chosen by the optimizer to do a matching scan.
- Searches of large tables on criteria with wildcards in leading positions, such as “*ABC”, typically impact performance. Use Literals for Like does not improve performance for such searches because it does not significantly reduce the number of rows scanned.

Follow these other guidelines for using Use Literals for Like:

- Try this user property with fields for which you have a strong indication from performance and from DB2 UDB utilities that inefficient indexes are being chosen by the optimizer. Deploy this solution to production if you see a significant improvement in performance with the user property.
- This property provides no benefit for case-insensitive queries.

CAUTION: Widespread use of this user property can drive up CPU use because more SQL parsing and prepares are required, and can result in package cache overflows. Use this user property sparingly, on fields for which you have confirmed the benefit by testing.

You can inactivate this user property or modify its value. You can create new instances of this user property, but you should not create more than one instance for a field.

Parent Object Type	Field
Functional Area	Querying

Use Literals For Merge

This user property enables the Merge Records operation to generate literals in SQL statements with predicates on columns known to have low cardinality.

Syntax Use Literals For Merge:*table_name*

Argument	Description
table_name	This parameter is the name of a table for which the current business component has a field that is a foreign key column to the named table.

Value TRUE or FALSE.

If the value of this user property is TRUE, then during the Merge Records process, literals, instead of bind variables, are used as criteria in SQL statements for which the predicate is on a column that is a foreign key to *table_name*.

If the value of this user property is FALSE or the user property is not defined on a business component, then during the Merge Records process, bind variables are used in criteria in all SQL statements.

Usage During the Merge Records process, bind variables are used for all predicates. In some cases, when the cardinality of the column is low, DB2 does table scans when using indexes is more efficient.

You can enable the Use Literals for Merge user property to direct the Merge Records process to generate literals in SQL statements for which the predicate is on a column known to have low cardinality. You should add this user property to the business component on which Merge Records is run.

For example, add Use Literals For Merge:S_BU and set its value to TRUE. During the Merge Records process, literals are used instead of bind variables in SQL statements whose predicates are on a column that is a foreign key to S_BU.

You can inactivate this user property or modify its value. You can also create new instances of this user property.

Parent Object Type Business Component

Functional Area Merge

Validate Parent Account

This user property indicates whether to validate the Parent Account Id during Account hierarchy changes.

Value

- 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

ViewMode Sort

This user property is used with the All Mode Sort user property to allow you to specify a custom sort specification on the business component when the business component is in a particular view mode.

See also "All Mode Sort" on page 78.

Syntax View Mode Sort: *mode_num*

Argument	Description
mode_num	This parameter is an integer representing a view mode. The valid integer parameter values in this context are: ■ 1 for ManagerView ■ 3 for AllView ■ 5 for OrganizationView ■ 7 for GroupView ■ 8 for CatalogView ■ 9 for SubOrganizationView For information about integers representing view modes, see *Siebel Object Interfaces Reference*.

Value The value of this user property is one or more names of fields on the business component, separated by commas and not enclosed in quotes. The priorities of the fields in the sort are given by the order of the fields in the list.

Usage You can override sorting specifications on the business component by using the All Mode Sort user property. If All Mode Sort is set to FALSE for a business component, effectively disregarding all other sort specifications, then you can set the ViewMode Sort user property to a particular sort specification when the business component is in a particular view mode.

For example, to sort a given business component by its Last Name field, and then by First Name, when the business component is in the AllView view mode, add the following business component user properties:

- All Mode Sort with value FALSE
- ViewMode Sort:3 with value Last Name, First Name

Use the ViewMode Sort user property to sort in views with visibility other than Personal or Sales Rep visibility, including Manager, All, Organization, Sub-Organization, Group, and Catalog views.

You can inactivate this user property or modify its value. You can also create new instances of this user property.

CAUTION: Sorts, including sorts defined by ViewMode Sort, must be based on indexed columns and in the sequential orders defined by the indexes to minimize any performance implications.

Parent Object Type	Business Component
Functional Area	Sort specification

WebGotoPlayerErrorPage

This user property allows you to specify the Web page that is displayed when an error occurs while playing a SmartScript in Siebel eMarketing.

Value	The value for the WebGotoPlayerErrorPage user property must be the name of a Web page.
Usage	When an error occurs while playing a SmartScript in Siebel eMarketing, the Web page specified in this user property is displayed. The error page has a control to return to the eSmartScript page that caused the error.
Parent Object Type	Applet
Functional Area	Web applets

WebGotoView

This user property allows you to specify the view to go to if either the FINISH or the CANCEL event is invoked in the Siebel SmartScript player applet while playing a SmartScript.

Value	The value for the WebGotoView user property must be the name of a view.
Usage	When the FINISH or CANCEL event is invoked (the corresponding button is pressed) in the Siebel SmartScript player applet while playing a SmartScript in Siebel eMarketing, the view specified in this user property is displayed.
Parent Object Type	Applet
Functional Area	Web applets

WorkFlow Behaviour

This user property specifies whether 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.

Value

- 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

5 SWE Tags

Siebel tags are special tags that you insert into Web template files. They specify how objects defined in the repository should be laid out and formatted in the final HTML page in the user's Web browser.

For information about configuring Siebel Business Applications Web pages and using Siebel tags, see *Configuring Siebel Business Applications*.

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 shows 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 shows the same page again with the error message displayed within the page. For errors that occur outside of a form submission, SWE continues to use the application's Error Page.

This error message display is intended for showing error messages within a form.

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 are shown in plain text, but each error message is 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 automatically generates an error node (a CSSSWEErrorSWX instance). This automatically-generated error node is 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: Use the `<swe:error>` tag inside a `<swe:form>` tag in an applet template only in standard interactivity applications. In high interactivity applications, these errors are shown using a pop-up window.

You should also use the <swe:error> 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 no longer supported.

An example of the use of the <swe:error> 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 are 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 starts the iteration by assigning this value to the iterator, and increments the iterator 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 catalog-like layout, the master and detail applets are configured as list applets. The master applet is referred to 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 have position values like 1, 2, and so on. For the applet with position 1, its immediate child applets are assigned position values 1.1, 1.2, and so on. It is possible to define a third-level applet 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 object definitions for this view are listed below.

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 has 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 has 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 iterates 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 is 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 is 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 subcategory values, configure the appropriate drilldown objects.

The catalog style layout forces a view to be shown in standard interactivity mode, 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. The term “tree item” refers to a tree element regardless of its type. A repository tree node is called 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 button, 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 button 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 determines 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 is printed if the tree item is a tree node. Otherwise, the field value is generated. The expand/collapse button, the item's icon, and the links are also parts of a tree item. Depending on the configuration file settings, the expand/collapse button is shown as either a GIF image or text. The expand/collapse button is only shown for tree items with child items. There are two links associated with each item. There is a link for the expand/collapse button to expand or collapse the item and a 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. Nonselected nodes are in an “Inactive” state.

type. Set to “DisplayName” or “FieldValue.” Outputs the repository tree node's Display Name if “DisplayName.” Otherwise, the type 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 depends 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 iterates 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 are 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 do 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 refreshes 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 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 retrieves 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.

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.
Page	In a container page template, specifies that the frame contains a Web page. These frames are not 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 is 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 generates standard names for the frames.

SWE supports nested framesets. In this case the `<swe:frame>` tag contains 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, use 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 a DHTML-based menu in an IE 5.0 browser, 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; hence, this attribute is 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 and 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 is 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 is displayed, based on the current view layout mode and layout preferences.

Usage

```
<swe:layout viewDisplayMode="xxx" appletDisplayMode="xxx" appletDisplaySize="xxx" />
```

Attributes

viewDisplayMode. (Optional) This attribute can have the value Layout or Show. If viewDisplayMode is Layout, the tag is shown only if the user is in Edit View Layout mode. If viewDisplayMode is Show, the tag is shown only if the user is not in Edit View Layout mode. If viewDisplayMode is not specified, the tag is shown whether the user is in Edit View Layout mode or not.

appletDisplayMode. (Optional) This attribute can have the value Show or Hide. If appletDisplayMode is Show, the tag is shown only if the applet is visible. If appletDisplayMode is Hide, the tag is shown only if the applet is hidden. If appletDisplayMode is not specified, the tag is shown regardless of applet visibility.

appletDisplaySize. (Optional) This attribute can have the value Min or Max. If appletDisplaySize is Min, the tag is shown only if the applet is minimized. If appletDisplaySize is Max, the tag is shown only if the applet is maximized. If appletDisplaySize is not specified, the tag is 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. (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, then the tag shows 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 view bar and view. A `<swe:pageitem>` in the view bar 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 screen bar. This tag is used to mark the beginning and end of the screen bar 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. (Optional) Can have the values Active or Inactive. If state is Active, then this tag is used only if the current view name being rendered is the currently active view. If state is Inactive, then this tag is used only if the current view name being rendered is not the currently active view. If not specified, the tag is used for all views.

Property. (Optional) Can have only one value, FormattedHtml, which generates the HTML for creating a link to navigate to the screen. If this attribute is not specified, then no output is generated.

htmlAttr. (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 displays numbers for list items. It should only be used for WML apps.

swe:scripts

Purpose

Specifies the location to place any JavaScript that may be generated for the page.

Usage

`<swe:scripts/>`

Attributes

None.

Restrictions

The 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 the Siebel Web Engine where to place the JavaScript. By default, if this tag is not specified, the 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, repositions it if needed, and removes any superfluous `<swe:scripts>` tags.

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. 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 is automatically selected. Users can unselect this record 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, multirecord selection is not enabled for list applets. To enable multirecord selection on a list applet, in Siebel Tools set to TRUE the HTML Multi Row Select attribute of the applet's List object.

To render the check boxes to select multiple rows in list applet templates, use the <`swe:select-row`> tag. With this tag, you can create a generic list applet template that can be used with list applets that support multirecord selection and list applets 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 with the <`swe:this`> tag to conditionally put in a <td> that contains the check box.

NOTE: You need to place your list applet controls and list columns within a <`swe:form`> tag when you enable the multiselect feature, as any invoke method on the applet requires the form that contains the row selection check boxes to be submitted.

The HTML Client framework marks 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 are not disabled. This is because there is no server call when selecting multiple records. Instead, when the control is activated the user gets a message 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 check box is rendered if the applet is enabled for multirecord selection in Siebel Tools. When <swe:select-row> tag is used without the property attribute, the tag acts as a conditional tag to show its body if the applet is enabled for multirecord 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 subview bar is rendered as an 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 has no 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 subviews.

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>&nbsp;&nbsp;&nbsp;</td>

         <td valign="top"><swe:pageitem id="2" property="DisplayName" /></td>

         <td ><swe:this property="FormattedHtml" /></td>

         <td width="100%">&nbsp;</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>&nbsp;<swe:viewname/>&nbsp;</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>&nbsp;<swe:viewname/>&nbsp;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">&nbsp;</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"/>:&nbsp; </td>

   <td> <swe:this property="FormattedHtml"/>&nbsp;</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 leads the user to a previous page. The thread buttons are separated by separators. In the preceding example, the right-angle bracket (>) is the separator.

For thread buttons that include a hyperlink, 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 is 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 is 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 is created when the user clicks the thread button and brings back a bookmarked view. The bookmark ID for the new view is 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 keeps the 20 most recently created 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 are deleted.

The HTML thread bar is based on the same configuration that was used previously to display the dedicated client thread bar. The behavior of the thread bar also remains unchanged, and is summarized below:

- When a new screen is requested, a new thread is created to replace the current thread.
- When a view button is clicked, the last thread step is replaced by that of the new view requested.
- When the user follows a drill-down link, a new step is appended to the thread bar for the view requested.
- When a thread button is clicked, all the thread buttons to the right of it are deleted.
- Some views may not have a thread applet or thread field defined. Showing these views does not cause the thread button to be updated.

When a thread button is clicked, the thread is 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:stepseparator. 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:stepseparator

Purpose

Specifies the symbol used to separate thread buttons. Include at the beginning or the end of the `<swe:threadbar>` block.

Usage

`<swe:stepseparator>` *`separator_symbol`* `</swe:stepseparator>`

Attributes

None.

NOTE: The `swe:threadlink` and `swe:stepseparator` 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">&nbsp;&gt;&nbsp;</span>
        </swe:stepseparator>
        ... HTML ...
    </swe:for-each-thread>
    ... HTML ...
</swe:threadbar>
```

This creates 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 view bar 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 is rendered as an HTML Select control showing the set of applets to which the user can toggle. 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 (it has no 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 is 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. (Optional) Can have value Active or Inactive. If state is Active, this tag is used only if the current applet title being rendered is the currently active applet. If state is Inactive, this tag is used only if the current applet title being rendered is not the currently active applet. If not specified, the tag is shown for all applets.

Property. (Optional) Can have only one value, FormattedHtml, which generates the HTML for creating a link to toggle to the applet. If this attribute is not specified, then no output is 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 give users the means 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 those 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 invokes the ShellUIInit method on the command target service when it tries to initialize. It invokes ShellUIExit when it exits. There is a set of communication protocols defined for the communication between the Java applet and the service.

The toolbar is 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 view bar (Level 2 or 3) for a given screen. There is always 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 or 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 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 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 contains a frameset that has 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 defines 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 does not generate any errors, but does allow showing multiple views in the application.

To show additional views in the content area, 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, for example “CCSMainAndSearchView.swt”, and load it using the LoadContentContainer method. CCSMainAndSearchView.swt contains 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 is used to replace this view with another view.

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, you want to be able to navigate to other views. In other words, there are multiple view zones now. In the above example, SWE navigates 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 causes a navigation to another view within the same view zone. You can also cause the main view to navigate to another view using BuildViewAsync(). Be sure to 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 view bar 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, Select. If the type is set to Select, the view bar is 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 are shown. If the value is NonContext, only the noncontext views are 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 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. (Optional) Can have value Active or Inactive. If the state is Active, this tag is used only if the current view name being rendered is the currently active view. If the state is Inactive, this tag is used only if the current view name being rendered is not the currently active view. If not specified, the tag is shown for all views.

Property. (Optional) Can have only one value, FormattedHtml, which generates the HTML for creating a link to navigate to the view. If this attribute is not specified, then no output is generated.

htmlAttr. (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, the SWE performs XSLT processing on the XML output and sends the transformed document as the response back to the client. When set to embed, the SWE inserts an XML processing instruction in the beginning of the XML document for external XSLT processing.

6 Siebel Templates for Employee Applications

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 for Employee Applications" on page 223
- "Applet Visual Reference" on page 224
- "Grid Form Layouts" on page 232
- "Non-Grid Form Layouts" on page 233
- "Applet Template Descriptions" on page 236
- "View Layouts" on page 282
- "View Template Descriptions" on page 283
- "Page Container Templates" on page 309
- "Specialized Applet Templates" on page 311
- "Specialized Views" on page 324

Overview of UI Elements for Employee Applications

Table 10 gives an overview of user interface elements in employee applications.

Table 10. Description of UI Elements

UI Elements	Description
Application menu	Application-level menu items such as File, Edit, View, Navigate, Tools, and Help.
Branding area	The area in which the Siebel Systems, Inc. logo appears.
Application toolbar	Contains toolbar icons for items such as Site Map, Customer Dashboard, and iHelp items.
Applet 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 the primary record changes.
First-level navigation	Set of tabs that allow users to navigate to screens.

Table 10. Description of UI Elements

UI Elements	Description
Second-level navigation	Links that allow users to navigate to views. These links can appear in the area directly under the screen tabs, or in the Visibility filter.
Third-level navigation	View tabs displayed below the first applet on a view. This is also referred to as the view bar.
Fourth-level navigation	Depending on the Web template being used, fourth-level navigation items can appear as links underneath the third-level tabs (most common), view tabs on a grandchild applet, or links in a drop-down list.
Record navigation	Area that displays position within the record set as well as controls to move forward and backward within the record set.
Search	Integrated Search and Global Find launch button.
Favorites	List of saved and predefined queries for the view.
iHelp frame	Area that displays links for navigating through steps in a task.

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.

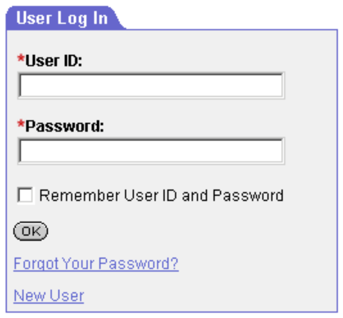
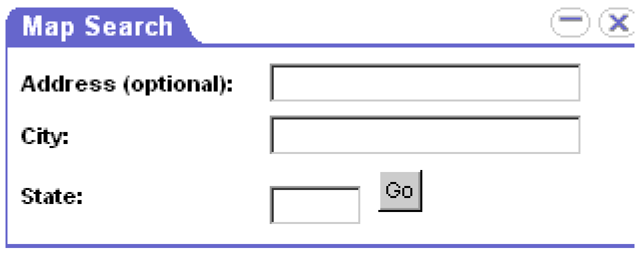


Figure 1. CCAppletForm1Col_B_E_N.swt

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 shown below.



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

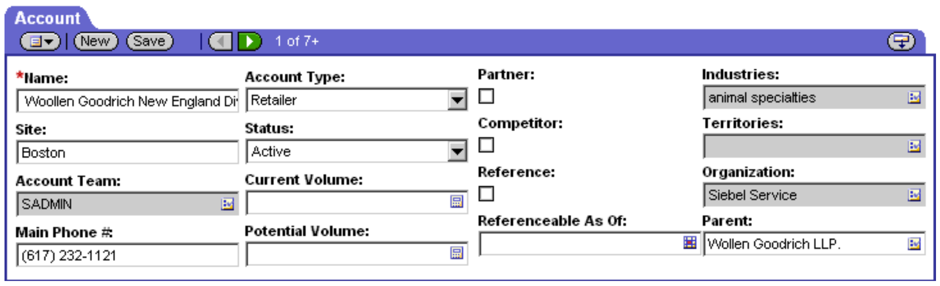
Applet Form 4-Col (Edit / New)

CCAppletForm4Col_E_N.swt

Applet List (Edit / New / Query)

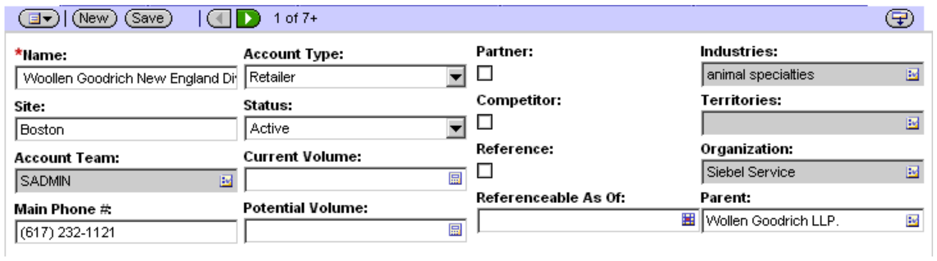
CCAppletList_E_N_Q.swt

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.)



*Name:	Account Type:	Partner:	Industries:
Woollen Goodrich New England Di	Retailer	☐	animal specialties
Site:	Status:	Competitor:	Territories:
Boston	Active	☐	
Account Team:	Current Volume:	Reference:	Organization:
SADMIN		☐	Siebel Service
Main Phone #:	Potential Volume:	Referenceable As Of:	Parent:
(617) 232-1121			Wollen Goodrich LLP.

Figure 2. Parent Style



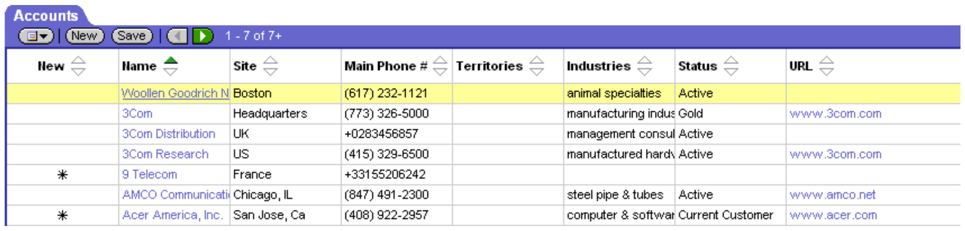
*Name:	Account Type:	Partner:	Industries:
Woollen Goodrich New England Di	Retailer	☐	animal specialties
Site:	Status:	Competitor:	Territories:
Boston	Active	☐	
Account Team:	Current Volume:	Reference:	Organization:
SADMIN		☐	Siebel Service
Main Phone #:	Potential Volume:	Referenceable As Of:	Parent:
(617) 232-1121			Wollen Goodrich LLP.

Figure 3. Child Style

Applet List (Base/EditList)

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.



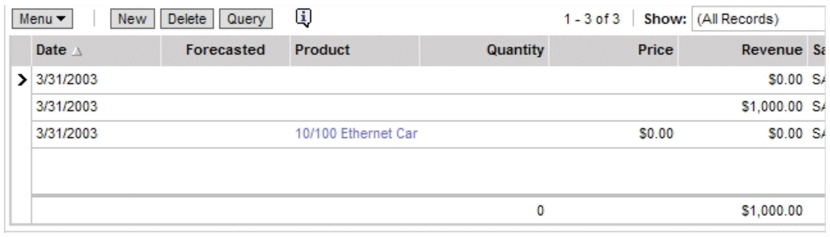
New	Name	Site	Main Phone #	Territories	Industries	Status	URL
	Woollen Goodrich N	Boston	(617) 232-1121		animal specialties	Active	
	3Com	Headquarters	(773) 326-5000		manufacturing indus	Gold	www.3com.com
	3Com Distribution	UK	+0283456857		management consul	Active	
	3Com Research	US	(415) 329-6500		manufactured hardv	Active	www.3com.com
*	9 Telecom	France	+33155206242				
	AMCO Communicati	Chicago, IL	(847) 491-2300		steel pipe & tubes	Active	www.amco.net
*	Acer America, Inc.	San Jose, Ca	(408) 922-2957		computer & softwar	Current Customer	www.acer.com

Figure 4. Applet List (Base/EditList)

Applet List Totals (Base/EditList)

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 a 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.



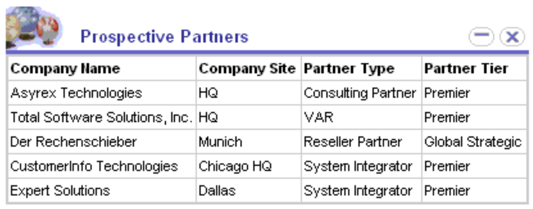
Date	Forecasted	Product	Quantity	Price	Revenue	Sa
3/31/2003					$0.00	S
3/31/2003					$1,000.00	S
3/31/2003		10/100 Ethernet Car		$0.00	$0.00	S
			0		$1,000.00	

Figure 5. Applet List Totals (Base/EditList)

List Portal (Graphical) Applet

CCAppletListPortalGraphical.swt

An example of the List Portal (Graphical) applet is shown in Figure 6.



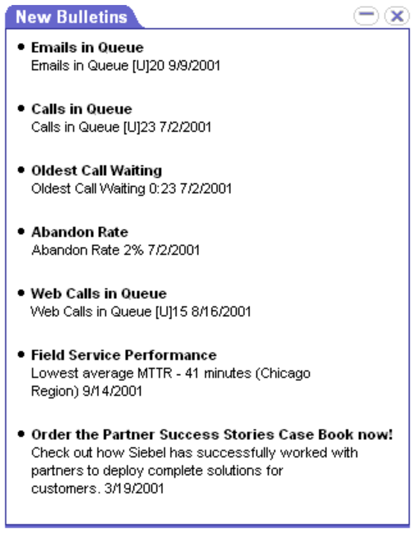
Company Name	Company Site	Partner Type	Partner Tier
Asyrex Technologies	HQ	Consulting Partner	Premier
Total Software Solutions, Inc.	HQ	VAR	Premier
Der Rechenschieber	Munich	Reseller Partner	Global Strategic
CustomerInfo Technologies	Chicago HQ	System Integrator	Premier
Expert Solutions	Dallas	System Integrator	Premier

Figure 6. List Portal (Graphical)

Applet List Message

CCAppletListMessage.swt

An example of the List Messages Applet is shown below.



- **Emails in Queue**
  Emails in Queue [U]20 9/9/2001
- **Calls in Queue**
  Calls in Queue [U]23 7/2/2001
- **Oldest Call Waiting**
  Oldest Call Waiting 0:23 7/2/2001
- **Abandon Rate**
  Abandon Rate 2% 7/2/2001
- **Web Calls in Queue**
  Web Calls in Queue [U]15 8/16/2001
- **Field Service Performance**
  Lowest average MTTR - 41 minutes (Chicago Region) 9/14/2001
- **Order the Partner Success Stories Case Book now!**
  Check out how Siebel has successfully worked with partners to deploy complete solutions for customers. 3/19/2001

Popup List, Popup Query, Popup Form

The following three templates are grouped together:

Popup List

CCPopupList.swt

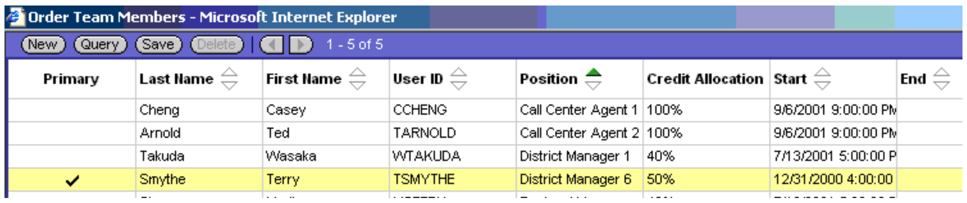
Popup Query

CCPopupQuery.swt

Popup Form

CCPopupForm.swt

Popup List and Popup Query applets are shown in Figure 7 and Figure 8.

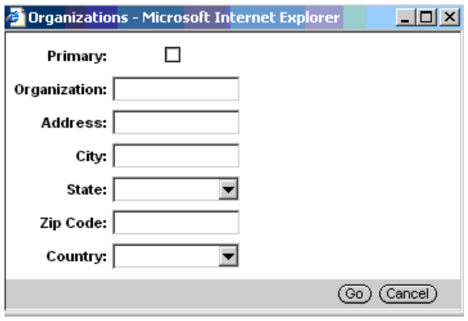


Order Team Members - Microsoft Internet Explorer

New Query Save Delete 1 - 5 of 5

Primary	Last Name	First Name	User ID	Position	Credit Allocation	Start	End
	Cheng	Casey	CCHENG	Call Center Agent 1	100%	9/6/2001 9:00:00 P	
	Arnold	Ted	TARNOLD	Call Center Agent 2	100%	9/6/2001 9:00:00 P	
	Takuda	Wasaka	WTAKUDA	District Manager 1	40%	7/13/2001 5:00:00 P	
✓	Smythe	Terry	TSMYTHE	District Manager 6	50%	12/31/2000 4:00:00	

Figure 7. Popup List



Organizations - Microsoft Internet Explorer

Primary:	☐
Organization:	
Address:	
City:	
State:	
Zip Code:	
Country:	

Go Cancel

Figure 8. Popup Query

Calendar Monthly, Calendar Weekly, Calendar Daily

The following three templates are grouped together:

Calendar Monthly

CCAppletCalendarMonthly.swt

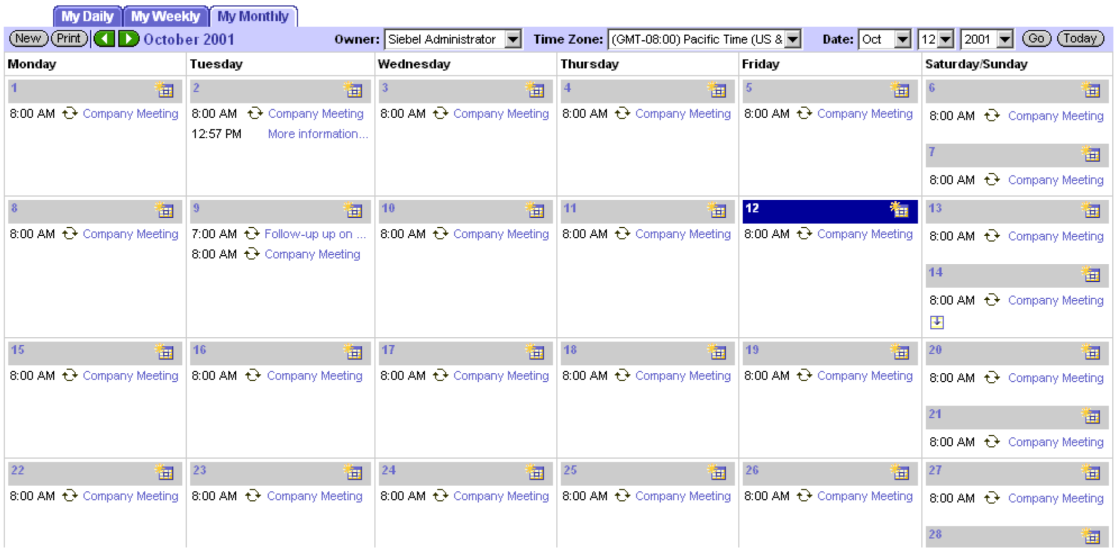
Calendar Weekly

CCAppletCalendarWeekly.swt

Calendar Daily

CCAppletCalendarDaily.swt

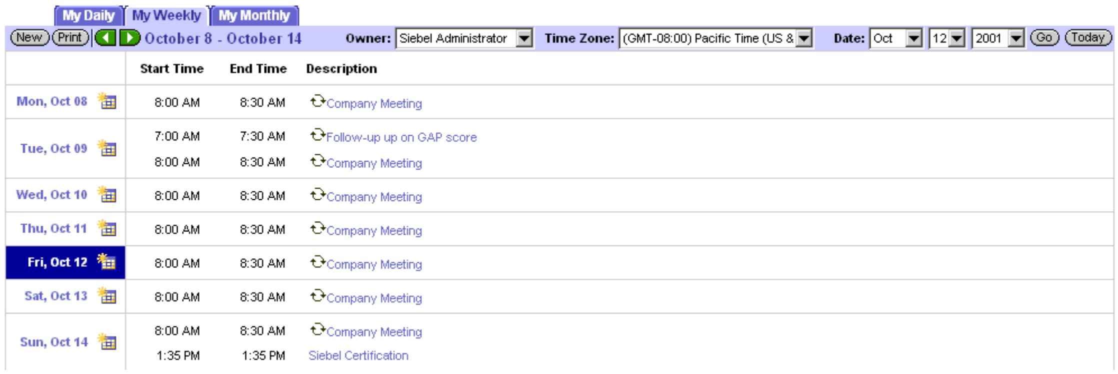
Each of the calendar templates supports the parent, child, and grandchild styles. (Parent style is shown in Figure 9, Figure 10, and Figure 11.)



October 2001 **Owner:** Siebel Administrator **Time Zone:** (GMT-08:00) Pacific Time (US & **Date:** Oct 12 2001

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday/Sunday
1 8:00 AM Company Meeting	2 8:00 AM Company Meeting 12:57 PM More information...	3 8:00 AM Company Meeting	4 8:00 AM Company Meeting	5 8:00 AM Company Meeting	6 8:00 AM Company Meeting 7 8:00 AM Company Meeting
8 8:00 AM Company Meeting	9 7:00 AM Follow-up up on ... 8:00 AM Company Meeting	10 8:00 AM Company Meeting	11 8:00 AM Company Meeting	12 8:00 AM Company Meeting	13 8:00 AM Company Meeting 14 8:00 AM Company Meeting
15 8:00 AM Company Meeting	16 8:00 AM Company Meeting	17 8:00 AM Company Meeting	18 8:00 AM Company Meeting	19 8:00 AM Company Meeting	20 8:00 AM Company Meeting 21 8:00 AM Company Meeting
22 8:00 AM Company Meeting	23 8:00 AM Company Meeting	24 8:00 AM Company Meeting	25 8:00 AM Company Meeting	26 8:00 AM Company Meeting	27 8:00 AM Company Meeting 28

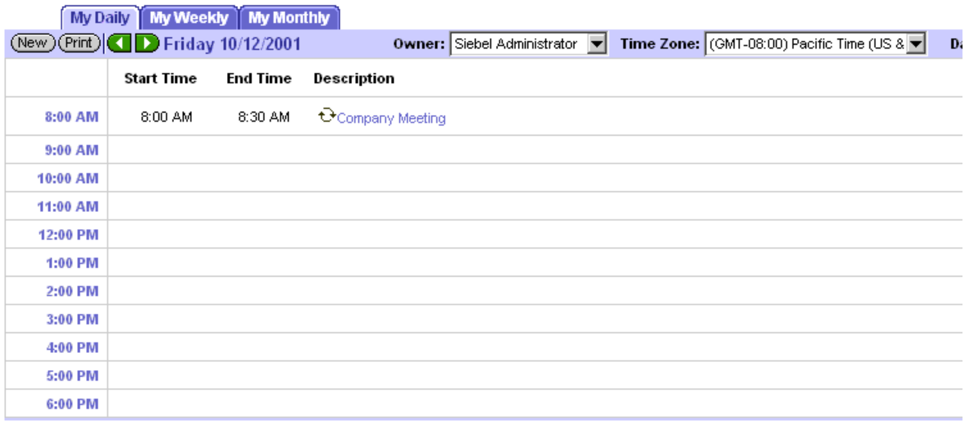
Figure 9. Monthly Calendar Applet



October 8 - October 14 **Owner:** Siebel Administrator **Time Zone:** (GMT-08:00) Pacific Time (US & **Date:** Oct 12 2001

	Start Time	End Time	Description
Mon, Oct 08	8:00 AM	8:30 AM	Company Meeting
Tue, Oct 09	7:00 AM	7:30 AM	Follow-up up on GAP score
	8:00 AM	8:30 AM	Company Meeting
Wed, Oct 10	8:00 AM	8:30 AM	Company Meeting
Thu, Oct 11	8:00 AM	8:30 AM	Company Meeting
Fri, Oct 12	8:00 AM	8:30 AM	Company Meeting
Sat, Oct 13	8:00 AM	8:30 AM	Company Meeting
Sun, Oct 14	8:00 AM	8:30 AM	Company Meeting
	1:35 PM	1:35 PM	Siebel Certification

Figure 10. Weekly Calendar Applet



Friday 10/12/2001

Owner: Siebel Administrator

Time Zone: (GMT-08:00) Pacific Time (US &

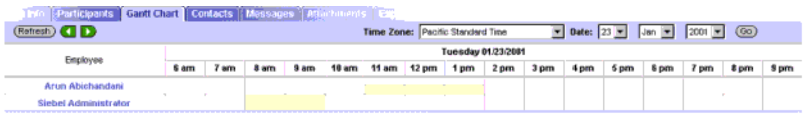
	Start Time	End Time	Description
8:00 AM	8:00 AM	8:30 AM	Company Meeting
9:00 AM			
10:00 AM			
11:00 AM			
12:00 PM			
1:00 PM			
2:00 PM			
3:00 PM			
4:00 PM			
5:00 PM			
6:00 PM			

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.



Time Zone: Pacific Standard Time

Date: 23 Jan 2001

Tuesday 01/23/2001

Employee	6 am	7 am	8 am	9 am	10 am	11 am	12 pm	1 pm	2 pm	3 pm	4 pm	5 pm	6 pm	7 pm	8 pm	9 pm
Arun Abichandani																
Siebel Administrator																

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.)

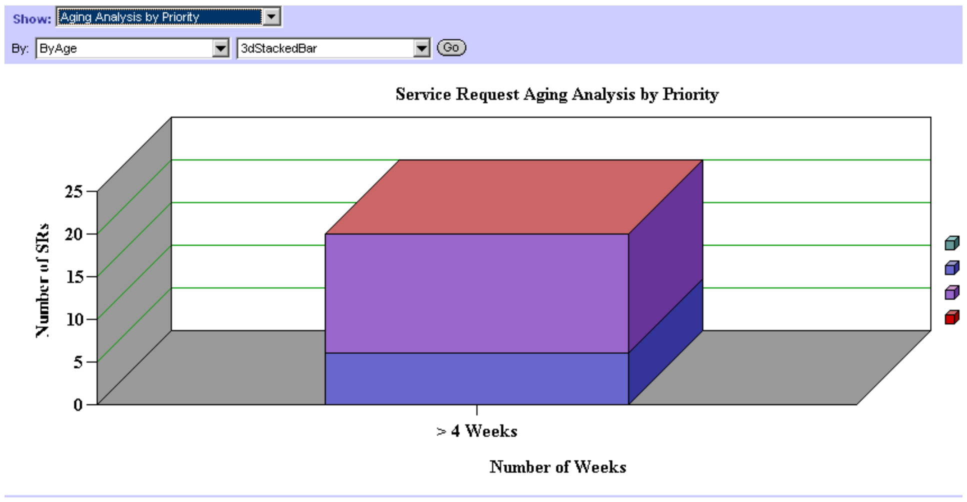

Show: Aging Analysis by Priority
By: ByAge
3dStackedBar
Go
Service Request Aging Analysis by Priority
Number of SRs
0
5
10
15
20
25
> 4 Weeks
Number of Weeks

Figure 13. Child Style

Grid Form Layouts

Grid-based templates allow you to modify form layout using the Applet Layout Editor in Siebel Tools, without having to modify the associated Web templates. Grid layout templates do not use placeholder tags as non-grid based layout templates do. Instead, grid layout templates use two Siebel tags (<swe:form-applet-layout> and </swe:form-applet-layout>) that serve as a single container for all controls in the main body of the form.

Grid layout templates consist of a body region and a header or footer. The body region is defined by <swe:form-applet-layout> tag and contains no placeholder tags. However, the header and footer regions do use placeholder tags for items such as buttons. You cannot edit the layout of header and footer regions using the Applet Layout Editor.

The following list summarizes how grid-based applet Web templates differ from non-grid applet Web templates:

- With grid-based templates, you can modify the layout of the form using Siebel Tools without having to modify the Web template itself.
- With grid-based templates, Labels and Controls behave as separate items in the Web Layout Editor. This allows you to place them independently in the applet layout. However, Labels and Controls are really a single object in the repository with one set of shared properties.
- Grid-based templates do not automatically compress empty space in a column. The browser compresses horizontal space as much as possible without changing the size of any fields on the form applet.

For examples of grid layout templates, see "Applet Form Grid Layout" on page 237 and "Applet Popup Form Grid Layout" on page 238.

Non-Grid Form Layouts

Non-grid form applets were used in releases prior to release 7.7 for the Edit mode of list applets in SI mode. Most form applets now use the grid layouts as described in the previous section. This section describes non-grid form layouts, which are still available, but are no longer used in preconfigured Siebel applications.

The four-column form templates define a set of layout regions. A region can hold one or more label/field pairs. Controls are dimensioned in Siebel 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.

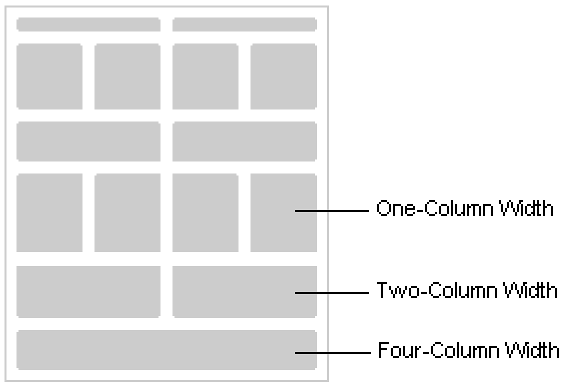


Figure 14. Master Template

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.

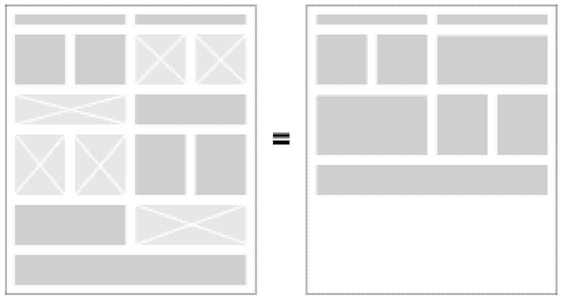


Figure 15. Layout Derived from the Master Template

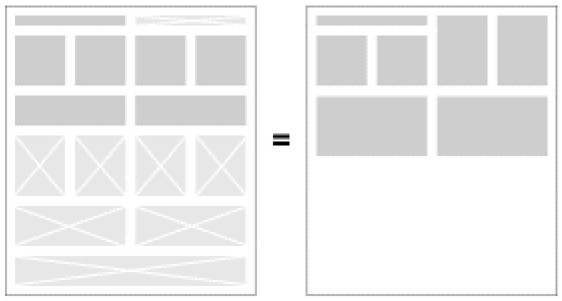


Figure 16. Layout Derived from the Master Template

Controls IDs Per Region for Non-Grid Form Templates

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.

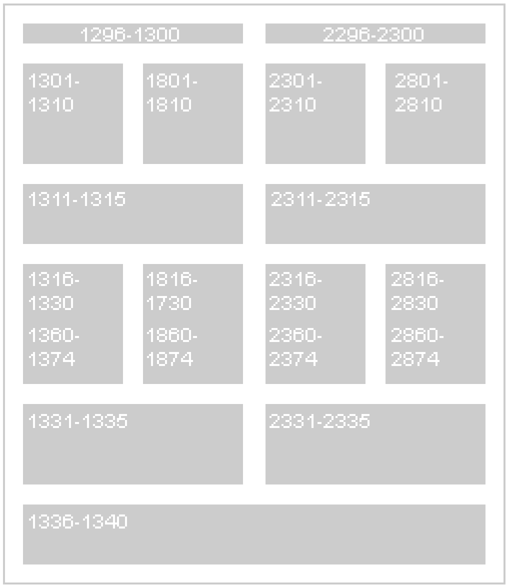


Figure 17. Control IDs for Each Region

Considerations for Using Applet Templates

Mapping a MiniButton

You should never map a native HTML button to a list or form. The Siebel Business UI standard is to use the custom control called *MiniButton*. 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 control.

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.

Vertical Alignment of Fields in a Non-Grid 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. To maximize the potential for vertical alignment, place taller fields (such as text area fields) at the bottom of forms.

Mapping a FormSection in a Form

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 Siebel Tools, find FormSection 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 expands to fit the region in which you place it. To set it apart, the label appears against the FormSection color defined in CSS.

NOTE: In non-grid forms, the control might not appear to expand to fit within the layout editor, but it is rendered correctly in the running application.

Applet Template Descriptions

- "Applet Tree Marketing" on page 272
- "Smart Script Player Applet (Tree Only)" on page 273
- "Applet Calendar Daily (Portal)" on page 274
- "eCalendar Daily Applet" on page 276
- "eCalendar Monthly Applet" on page 277
- "eCalendar Weekly Applet" on page 279
- "Service Calendar Applet" on page 280
- "Applet Chart" on page 281

Applet Form Grid Layout

SWT filename: CCAppletFormGridLayout.swt

This is the applet Web template that supports grid-based layout of controls on form applets, as shown in Figure 18. 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.

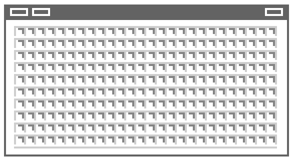


Figure 18. Applet Form Grid Layout

Includes Tree

```
CCAppletFormGridLayout.swt
    CCApplet_NamedSpacer.swt
    CCTitle_Named.swt
    CCFormButtonsTop.swt
        CCButtons.swt
```

Table 11 lists the controls that can be mapped to the header region of the Applet Form Grid Layout template.

Table 11. Mappable Items

ID	Description
2	...
91	Inside Applet Help Text

Table 11. Mappable Items

ID	Description
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
580	New - shows only in SI
599	Save - shows only in HI

Applet Popup Form Grid Layout

SWT filename: CCAppletPopupFormGridLayout.swt

This is the applet Web template that supports grid-based layout of controls for pop-up applets, as shown in Figure 19. 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.

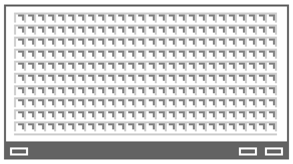


Figure 19. Applet Popup Form Grid Layout

Includes Tree

```
CCAppletPopupFormGridLayout.swt

    CCApplet_NamedSpacer.swt

    CCButtons.swt

    CCButtons_Popup.swt
```

The items listed in Table 12 can be mapped to the footer region of the Applet Popup Form Grid Layout template.

Table 12. 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
580	New - shows only in SI
599	Save - shows only in HI

Applet List (Base/EditList)

SWT Filename: CCAppletList_B_EL.swt

This is the standard list template for lists in base or editlist modes, as shown in Figure 20. 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 80 fields, but this is done so that you may mark the majority as available but hidden. Fields marked as such do not appear by default in the list but appear in the columns displayed dialog.

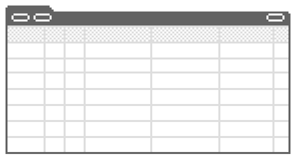


Figure 20. Applet List (Base/EditList)

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
```

Table 13 lists the mappable items for this template.

Table 13. Mappable Items

ID	Description
2	...
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
121	First

Table 13. Mappable Items

ID	Description
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
160-164	Control
501-540	Field
580	New - shows only in SI
598	Save
599	Save - shows only in HI
611-650	Field
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 21 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.

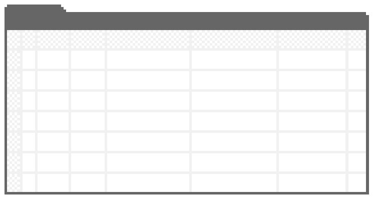


Figure 21. Applet List Inverted

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
```

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

Table 14. Mappable Items

ID	Description
123	Next
124	Last
131	New
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
580	New - shows only in SI
599	Save - shows only in HI
1100	Outside Applet Help Text
1500	Required Legend

Applet List Message

SWT Filename: CCAppletListMessage.swt

Applet List Message, shown in Figure 22, 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 22. 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 15 lists the mappable items for this template.

Table 15. Mappable Items

ID	Description
2	Back
90	Title
106	Query

Table 15. Mappable Items

ID	Description
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
212	HideApplet
501	Field
502-511	Field
555	Label
580	New - shows only in SI
599	Save - shows only in HI
1100	Outside Applet Help Text

Applet List Portal

SWT Filename: CCAppletListPortal.swt

This is the standard list template, shown in Figure 23. 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.

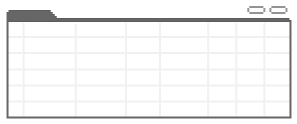


Figure 23. Applet List Portal

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 16 lists the mappable items for this template.

Table 16. Mappable Items

ID	Description
2	Back
90	Title
106	Query
107	Go (ExecuteQuery)

Table 16. Mappable Items

ID	Description
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
184	Drilldown Title
203	MinimizeApplet
204	MaximizeApplet
207	MoveAppletUp
208	MoveAppletDown
211	ShowApplet
212	HideApplet
501-540	Field
555	Label
580	New - shows only in SI
599	Save - shows only in HI
1100	Outside Applet Help Text

Applet List Portal (Graphical)

SWT Filename: CCAppletListPortalGraphical.swt

This list template, shown in Figure 24, 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 drilling down 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.

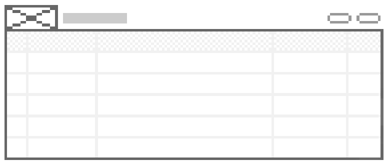


Figure 24. Applet List Portal (Graphical)

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 17 lists the mappable items for this template.

Table 17. 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
184	Drilldown Title
203	MinimizeApplet
204	MaximizeApplet
207	MoveAppletUp
208	MoveAppletDown
211	ShowApplet
212	HideApplet

Table 17. Mappable Items

ID	Description
501-540	Field
555	Label
580	New - shows only in SI
599	Save - shows only in HI
1100	Outside Applet Help Text

Applet List Search Results

SWT Filename: CCAppletListSearchResults.swt

This applet, shown in Figure 25, 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 25. Applet List Search Results

Includes Tree

CCAppletListSearchResults.swt

    CCButtons.swt

    CCRecordNav.swt

    CCListHeader.swt

    CCListBodySearchResults.swt

Table 18 lists the mappable items for this template.

Table 18. Mappable Items

ID	Description
2	Back
106	Query
107	Go (ExecuteQuery)

Table 18. Mappable Items

ID	Description
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	Select
145	Control
146	Control
147	...
501-540	...
580	New - shows only in SI
598	...
599	Save - shows only in HI

Applet List Totals (Base/EditList)

SWT Filename: CCAppletListTotals_B_EL.swt

This is the standard list template for lists in base or editlist modes. See Figure 26 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 do not appear by default in the list but appear in the columns displayed dialog.

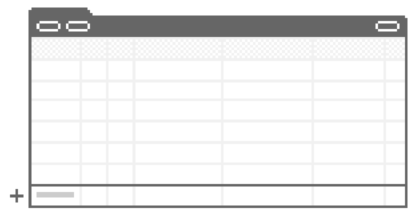


Figure 26. Applet List Totals

Includes Tree

```
CCAppletListTotals_B_EL.swt
    CCApplet_NamedSpacer.swt
    CCTitle_Named.swt
        CCTitle.swt
    CCListButtonsTop.swt
        CCButtons.swt
        CCRecordNav.swt
        CCTogglebar_drop.swt
        CCListButtonsTopRight.swt
    CCListHeaderTotals.swt
    CCListBodyTotals.swt
```

Table 19 lists the mappable items for this template.

Table 19. Mappable Items

ID	Description
2	...
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)

Table 19. Mappable Items

ID	Description
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
160-164	Control
199	Totals Label
501-520	...
580	New - shows only in SI
599	Save - shows only in HI
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 27 for an example.

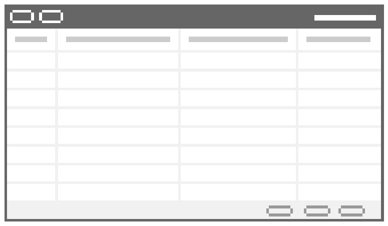


Figure 27. Popup List

Includes Tree

```
CCPopupList.swt
    CCListButtonsTop.swt
        CCButtons.swt
        CCRecordNav.swt
        CCTogglebar_drop.swt
        CCListButtonsTopRight.swt
    CCListHeader.swt
    CCListBody.swt
    CCButtons_Popup.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
152	OK
153	Cancel
154-158	Control
160-164	Control
501-540	Field
580	New - shows only in SI
598	Save
599	Save - shows only in HI

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 28. 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.

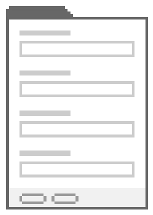


Figure 28. Applet Form 1 Column Light (Base/Edit/New)

Includes Tree

```
CCAppletForm1ColLight_B_E_N.swt

    CCApplet_NamedSpacer.swt

    CCTitle_Named.swt

        CCTitle.swt

    CCForm1ColBodyLight.swt

        dCCFormButtonsBottom.swt

        dCCButtons_Form.swt
```

Table 21 lists the mappable items for this template.

Table 21. 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

Table 21. Mappable Items

ID	Description
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 29. 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.

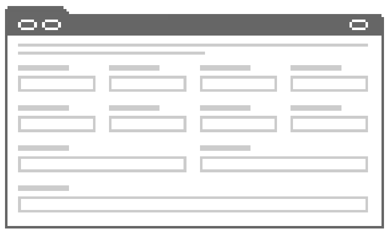


Figure 29. Applet Form 4 Column (Base)

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
```

Table 22 lists the mappable items for this template.

Table 22. 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
580	New - shows only in SI
599	Save - shows only in HI
1001-1009	FormSection
1020	FormSection

Table 22. 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
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 30. 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.

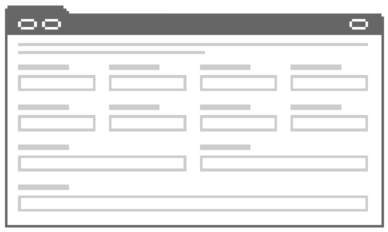


Figure 30. Applet Form 4 Column (Edit/New)

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
```

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)

Table 23. Mappable Items

ID	Description
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
580	New - shows only in SI
599	Save - shows only in HI
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

Table 23. Mappable Items

ID	Description
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 31. 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.

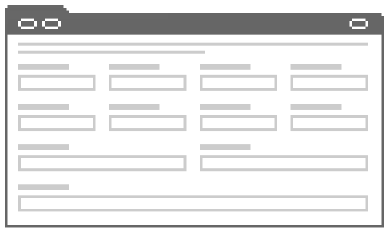


Figure 31. Applet Form 4-Col (No Record Nav)

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
```

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
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
150-152	Control
160-164	Control
580	New - shows only in SI
599	Save - shows only in HI
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

Table 24. Mappable Items

ID	Description
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 32 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.

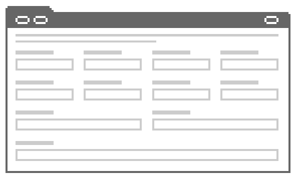


Figure 32. Applet List Edit (Edit/New/Query)

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
```

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
121	First
122	Previous
123	Next
124	Last
131	New
132	Edit

Table 25. Mappable Items

ID	Description
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
150-152	Control
160-164	Control
580	New - shows only in SI
599	Save - shows only in HI
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
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

Table 25. Mappable Items

ID	Description
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 33 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.

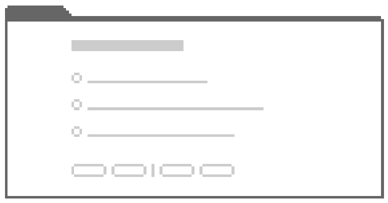


Figure 33. Applet Wizard

Includes Tree

```
CCAppletFormWizard.swt

    CCTitle_Mapped.swt

    CCForm1ColBody.swt

    CCButtons.swt
```

Table 26 lists the mappable items for this template.

Table 26. Mappable Items

ID	Description
2	Back
90	Title
91	Inside Applet Help Text
106	Query
107	Go (ExecuteQuery)

Table 26. Mappable Items

ID	Description
108	Cancel (Query)
109-111	Control
131	New
132	Edit
133	Delete
134	Reset
135	Cancel
136	Save
139-143	Control
184	Title
580	New - shows only in SI
599	Save - shows only in HI
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

    CCHTMLHeader.swt

    CCBottomApplet.swt

    CCHTMLFooter.swt
```

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 34 for an example.

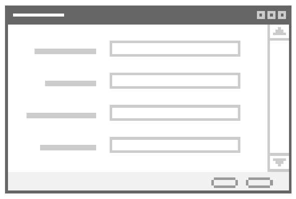


Figure 34. Popup Form

Includes Tree

```
CCPopupForm.swt

    CCStylesChoice.swt

    CCButtons.swt

    CCButtons_Popup.swt
```

Table 27 lists the mappable items for this template.

Table 27. 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

Table 27. Mappable Items

ID	Description
154-158	Control
580	New - shows only in SI
599	Save - shows only in HI
1001-1009	FormSection
1090-1099	Required; Label; field
1100	Label With Rule
1101-1110	Required; Label; Field
1111-1115	Required; Label; Field
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 35 for an example.

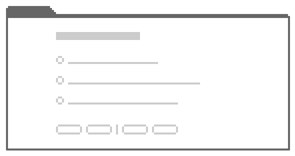


Figure 35. SmartScript Player Applet (Player Only)

NOTE: To encourage form completion, these applets display buttons at the bottom of the form.

Includes Tree

CCSmartScriptPlayerApplet.swt

Table 28 lists the mappable items for this template.

Table 28. 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 is a standard tree template that displays applet tab and border treatment. See Figure 36 for an example.

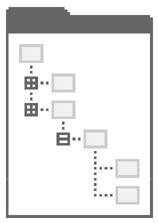


Figure 36. Applet Tree 2

Includes Tree

```
CCAppletTree2.swt

    CCApplet_NamedSpacer.swt

    CCTitle_Named.swt

        CCTitle.swt

    CCListButtonsTopNoRecNav.swt

    CCButtons.swt

    CCListButtonsTopRight.swt
```

Table 29 lists the mappable items for this template.

Table 29. 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
580	New - shows only in SI
599	Save - shows only in HI
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 37 for an example.

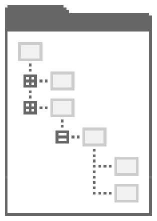


Figure 37. Applet Tree Marketing

Includes Tree

CCAppletTreeMarketing.swt

    CCTitle.swt

Table 30 lists the mappable items for this template.

Table 30. 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 38 for an example.

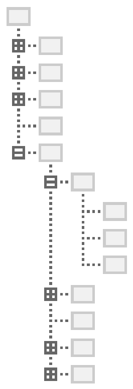


Figure 38. Smart Script Player Applet (Tree Only)

Includes Tree

CCAppletTree.swt

Table 31 lists the mappable items for this template.

Table 31. 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 39 for an example.

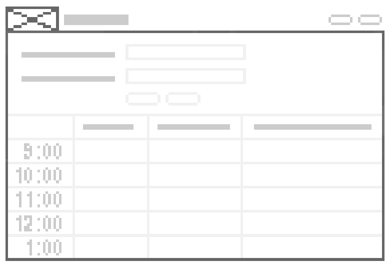


Figure 39. Applet Calendar Daily (Portal)

Includes Tree

```
CCAppletCalendarDailyPortal.swt
    CCLayoutTitlePortal.swt
        CCLayoutButtons.swt
        CCBottomApplet.swt
    CCCalendarAppletTitleGraphical.swt
        CCLayoutButtons.swt
```

Table 32 lists the mappable items for this template.

Table 32. Mappable Items

ID	Description
89	Image
90	Title
101	Label
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

Table 32. Mappable Items

ID	Description
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 40 for an example.

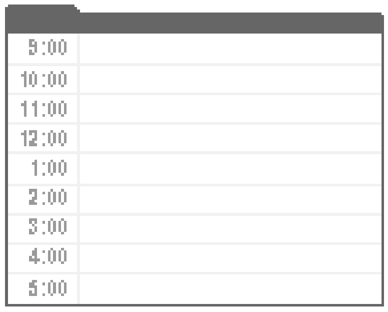


Figure 40. eCalendar Daily Applet

Includes Tree

```
CCAppletCalendarDaily.swt

    CCApplet_NamedSpacer.swt

    CCTitle_Named.swt

        CCTitle.swt

    CCBottomApplet.swt
```

Table 33 lists the mappable items for this template.

Table 33. Mappable Items

ID	Description
101	Label
102	Field
103	...
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 41 for an example.

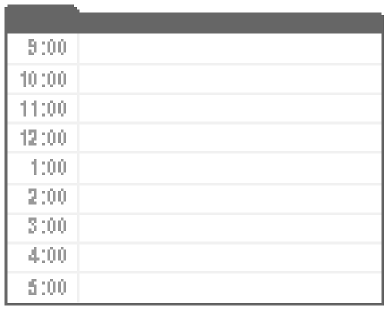


Figure 41. eCalendar Monthly Applet

Includes Tree

```
CCAppletCalendarMontly.swt

    CCApplet_NamedSpacer.swt

    CCTitle_Named.swt

        CCTitle.swt

    CCCalendarMonthly_weekday.swt

    CCCalendarMonthly_weekend.swt

    CCBottomApplet.swt
```

Table 34 lists the mappable items for this template.

Table 34. Mappable Items

ID	Description
101	Label
102	Field
103	...
104	...
105	...
106	...
107	Label
108	Field
130	...

Table 34. Mappable Items

ID	Description
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 42 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.

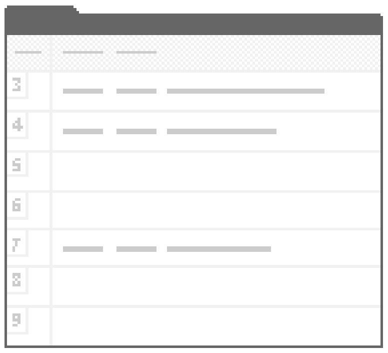


Figure 42. eCalendar Weekly Applet

Includes Tree

```
CCAppletCalendarWeekly.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	...
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 43 for an example.

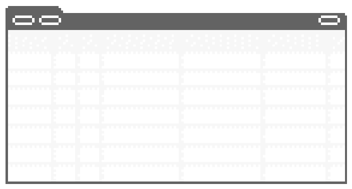


Figure 43. Service Calendar Applet

Includes Tree

```
CCAppletCalendarService.swt
    CCApplet_NamedSpacer.swt
    CCTitle_Named.swt
        CCTitle.swt
    CCBottomApplet.swt
```

Table 36 lists the mappable items for this template.

Table 36. 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 44 for an example.

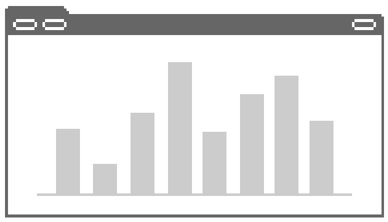


Figure 44. Applet Chart

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 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
```

Table 37 lists the mappable items for this template.

Table 37. Mappable Items

ID	Description
2	...
501-520	...
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 an example, see "View Template Descriptions" on page 283). 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.

NOTE: In Siebel Tools, style declarations are not evaluated. Therefore, color schemes and applet titles may display differently in Siebel Tools than they do in the running application.

View Issues

- Mapping tree applet maps onto views other than the 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.

- Subframe views

  View 25 – 75 (Framed) demonstrates subframe 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 run time and determines which color scheme the applet displays and whether its title is displayed.

  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” on page 284
- “View 25 - 50 – 25” on page 285
- “View 25 – 75” on page 286
- “View 25 – 75 Framed” on page 287
- “View 25 75 Framed 2” on page 288
- “View 50 – 50” on page 289
- “View 66 – 33” on page 290
- “View Admin 1” on page 291
- “View Admin 1 (Grandchild Indented)” on page 292
- “View Basic” on page 293
- “View Catalog Admin” on page 294
- “View Detail” on page 295
- “View Detail (Grandchild Indented)” on page 296
- “View Detail 2” on page 298
- “View Detail 2 (Grandfather Indent)” on page 299
- “View Detail 3” on page 300
- “View Detail 3 (Grandchild Indented)” on page 302
- “View Detail 3 Multi Child” on page 303
- “View Detail Multi-Child” on page 304
- “View Search” on page 305
- “View Tree” on page 306
- “View Tree 2” on page 307

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 45 for an example.

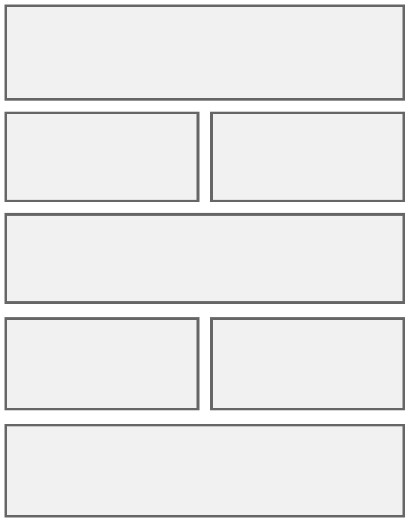


Figure 45. 1 Over 2 Over 1

Includes Tree

```
CCView_1Over2Over1.swt

    CCHTMLHeader.swt

        CCStylesChoice.swt

    CCThreadbar.swt

    CCHTMLFooter.swt
```

Table 38 lists the mappable items for this template.

Table 38. Mappable Items

ID	Description
101	Salutation Applet
102-106	Applet
201	Mini-Applet
202-206	Applet
302-306	Applet

Table 38. Mappable Items

ID	Description
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 46 for an example.

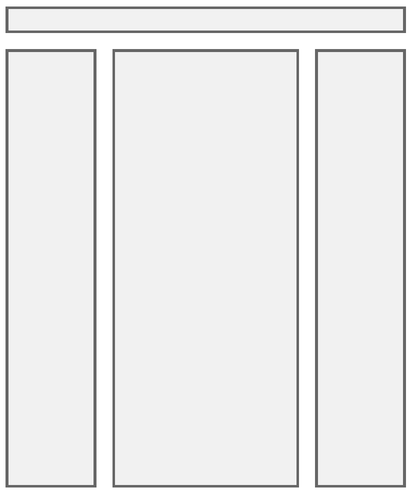


Figure 46. View 25 - 50 - 25

Includes Tree

```
CCView_25_50_25.swt
    CCHTMLHeader.swt
        CCStylesChoice.swt
    CCThreadbar.swt
    CCHTMLFooter.swt
```

Table 39 lists the mappable items for this template.

Table 39. 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 47 for an example.

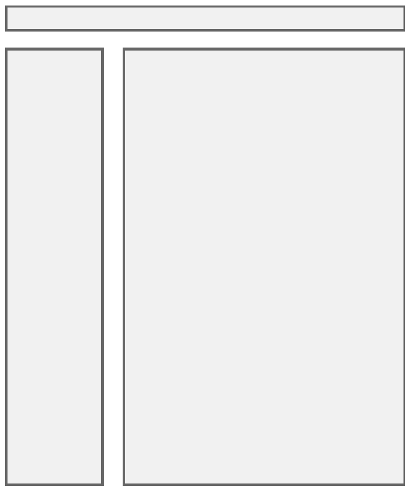


Figure 47. View 25 - 75

Includes Tree

```
CCView_25_75.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
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 48 for an example.

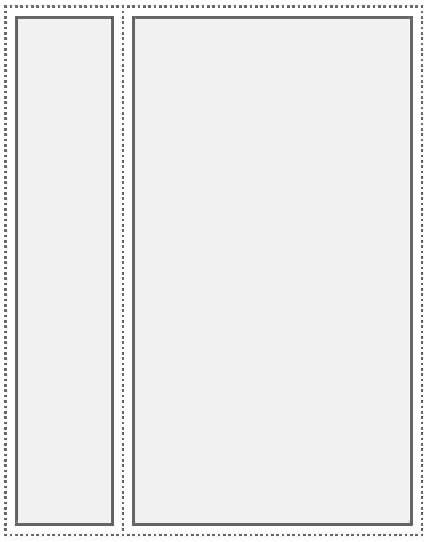


Figure 48. View 25 - 75 (Framed)

Includes Tree

```
CCView_25_75_Framed.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
102-106	Applet
108-112	Applet
201	Mini-Applet
202-211	Applet

View 25 75 Framed 2

SWT Filename: CCView_25_75_Framed2.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 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 49 for an example.

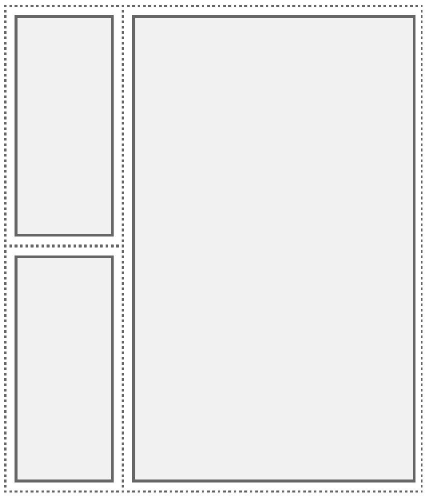


Figure 49. View 25 75 Framed 2

Includes Tree

```
CCView_25_75_Framed2.swt
```

```
    CCHTMLHeader.swt
        CCStylesChoice.swt
    CCThreadbar.swt
    CCHTMLFooter.swt
```

xTable 42 lists the mappable items for this template.

Table 42. 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 50 for an example.

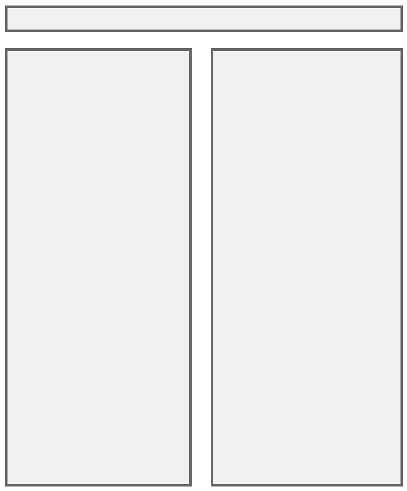


Figure 50. View 50 - 50

Includes Tree

```
CCView_50_50.swt
    CCHTMLHeader.swt
```

```
        CCStylesChoice.swt
    CCThreadbar.swt
    CCHTMLFooter.swt
```

Table 43 lists the mappable items for this template.

Table 43. 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 51 for an example.

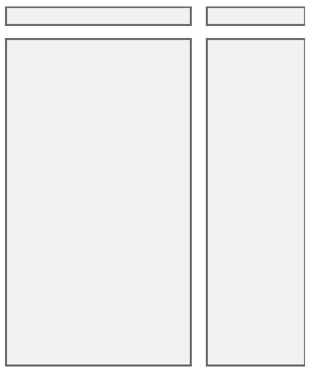


Figure 51. View 66 - 33

Includes Tree

```
CCView_66_33.swt
    CCHTMLHeader.swt
        CCStylesChoice.swt
    CCThreadbar.swt
    CCHTMLFooter.swt
```

Table 44 lists the mappable items for this template.

Table 44. Mappable Items

ID	Description
101	Salutation Applet
102-121	Applet
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 52 for an example. It is useful for admin views that need to display nonrelated views that are not easily categorized.

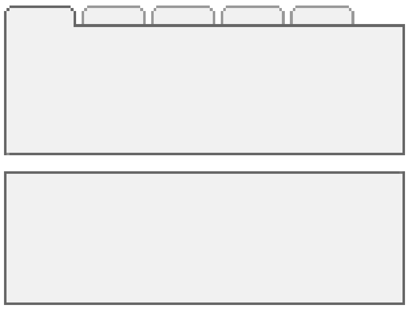


Figure 52. View Admin 1

Includes Tree

```
CCViewAdmin1.swt
    CCHTMLHeader.swt
        CCStylesChoice.swt
    CCThreadbar.swt
    CCViewbar_Tabs.swt
    CCSubViewbar_Tabs.swt
    CCHTMLFooter.swt
```

Table 45 lists the mappable items for this template.

Table 45. 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 53 for an example.

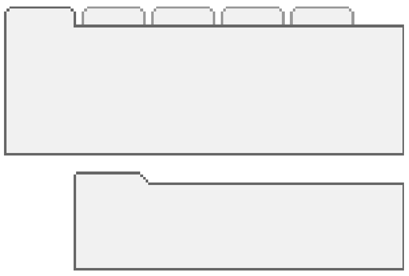


Figure 53. View Admin 1 (Grandchild Indented)

Includes Tree

```
CCViewAdmin1_GrndchldIndnt.swt
    CCHTMLHeader.swt
        CCStylesChoice.swt
    CCThreadbar.swt
    CCSubViewbar_Tabs.swt
        CCApplet_Spacer.swt
    CCHTMLFooter.swt
```

Table 46 lists the mappable items for this template.

Table 46. Mappable Items

ID	Description
5	Child Applet With Pointer
6	Child Applet
7	Grandchild Applet With Pointer
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 54 for an example.

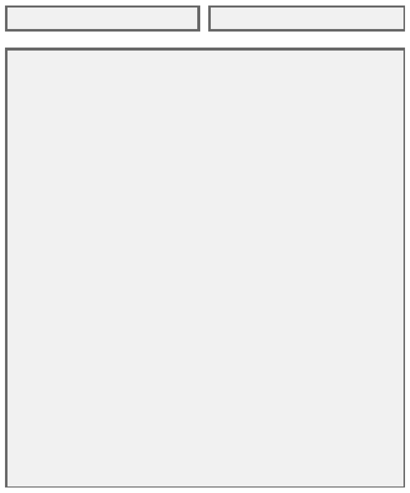


Figure 54. View Basic

Includes Tree

```
CCViewBasic.swt

    CCHTMLHeader.swt

        CCStylesChoice.swt
    CCThreadbar.swt

    CCHTMLFooter.swt
```

Table 47 lists the mappable items for this template.

Table 47. 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 55 for an example.

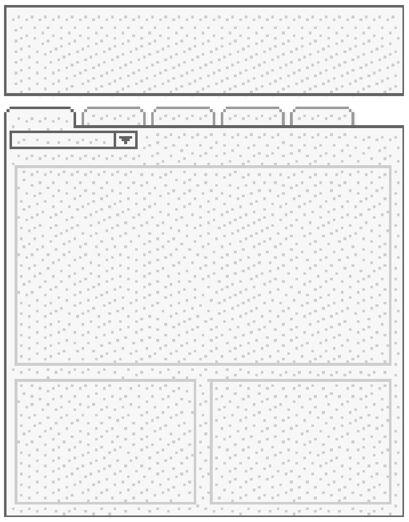


Figure 55. View Catalog Admin

Includes Tree

```
CCViewCatalog.swt

    CCHTMLHeader.swt

        CCStylesChoice.swt

    CCThreadbar.swt
    CCViewbar_Tabs_DropList.swt

    CCHTMLFooter.swt
```

Table 48 lists the mappable items for this template.

Table 48. 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 a parent applet, noncontext views presented as tabs, categorized subviews presented in a drop-down list, child applet, and multiple grandchild applets. See Figure 56 for an example.

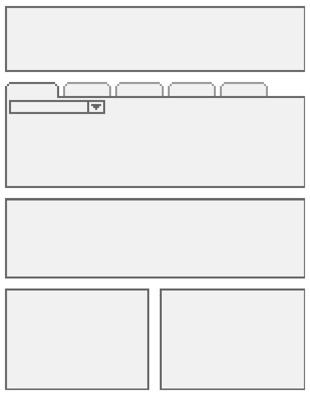


Figure 56. View Detail

Includes Tree

```
CCViewDetail.swt

    CCHTMLHeader.swt
```

```
        CCStylesChoice.swt
    CCThreadbar.swt
    CCViewbar_Tabs_DropList.swt
    CCHTMLFooter.swt
```

Table 49 lists the mappable items for this template.

Table 49. 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 a parent applet, noncontext views presented as tabs, a child applet, categorized subviews presented as a drop-down list, and multiple grandchild applets. See Figure 57 for an example.

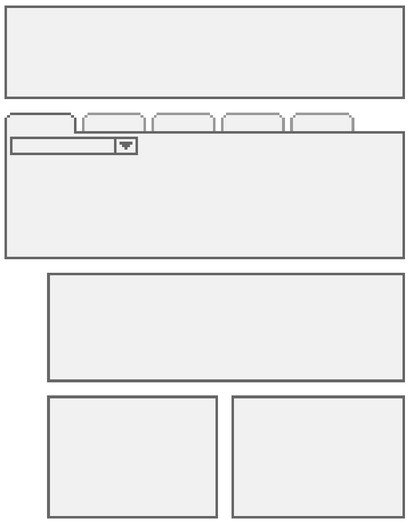


Figure 57. View Detail (Grandchild with Indent)

NOTE: Grandchild applets appear indented to convey hierarchy.

Includes Tree

```
CCViewDetail_GrndchldIndnt.swt
    CCHTMLHeader.swt
        CCStylesChoice.swt
    CCThreadbar.swt
    CCViewbar_Tabs_DropList.swt
    CCHTMLFooter.swt
```

Table 50 lists the mappable items for this template.

Table 50. Mappable Items

ID	Description
1	Parent Applet
2	...
3-5	Child or Grandchild Applet
6-7	Grandchild Applet

Table 50. Mappable Items

ID	Description
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 58 for an example.

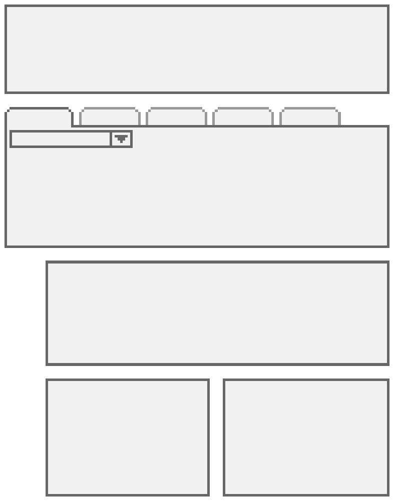


Figure 58. View Detail 2

Includes Tree

```
CCViewDetail2.swt

    CCHTMLHeader.swt

        CCStylesChoice.swt

    CCThreadbar.swt

    CCViewbar_Tabs.swt

    CCSubViewbar_Tabs.swt

    CCHTMLFooter.swt
```

Table 51 lists the mappable items for this template.

Table 51. 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 a parent applet, noncontext views presented as tabs, a child applet, categorized subviews presented as tabs, and multiple grandchild applets. See Figure 59 for an example.

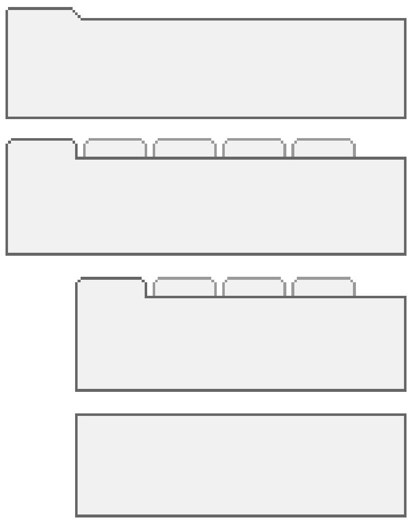


Figure 59. View Detail 2 (Grandfather Indent)

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 52 lists the mappable items for this template.

Table 52. 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 60 for an example.

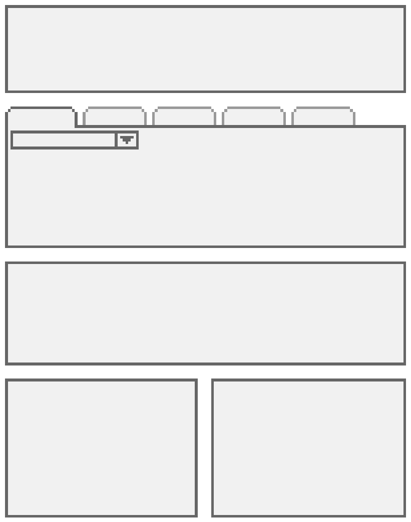


Figure 60. View Detail 3

Includes Tree

```
CCViewDetail3.swt
    CCHTMLHeader.swt
        CCStylesChoice.swt
    CCThreadbar.swt
    CCViewbarAll_Tabs.swt
    CCSubViewbar_Drop.swt
    CCHTMLFooter.swt
```

Table 53 lists the mappable items for this template.

Table 53. Mappable Items

ID	Description
1	Parent Applet
2	...
3-5	Grandchild Applet
6-7	Grandchild Applet

Table 53. Mappable Items

ID	Description
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 61 for an example.

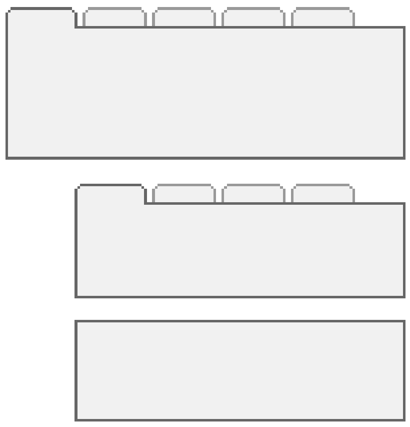


Figure 61. View Detail 3 (Grandchild Indented)

NOTE: Child and grandchild applets appear indented to convey hierarchy.

Includes Tree

```
CCViewDetail3_GrndchldIndnt.swt

    CCHTMLHeader.swt

        CCStylesChoice.swt

    CCThreadbar.swt

    CCViewbarAll_Tabs_DropList.swt

    CCHTMLFooter.swt
```

Table 54 lists the mappable items for this template.

Table 54. 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 62 for an example of the View Detail 3 Multi Child template.

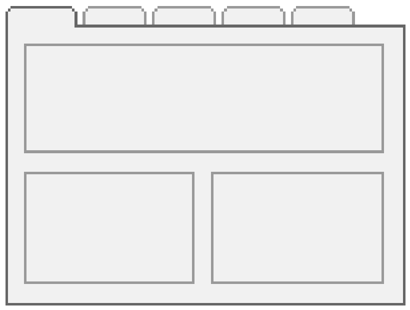


Figure 62. View Detail 3 Multi Child

Includes Tree

```
CCViewDetail3MultiChild.swt

    CCHTMLHeader.swt

        CCStylesChoice.swt

    CCThreadbar.swt

    CCViewbarAll_Tabs_DropList.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	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 list, a child applet, and multiple grandchild applets. See Figure 63 for an example.

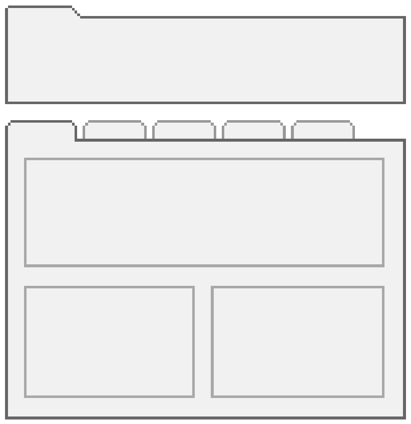


Figure 63. View Detail Multi-Child

NOTE: The child and grandchild applets appear inside a bounding box.

Includes Tree

```
CCViewDetailMultiChild.swt
```

```
    CCHTMLHeader.swt

        CCStylesChoice.swt

    CCThreadbar.swt

    CCViewbarAll_Tabs_DropList.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
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 64 for an example.

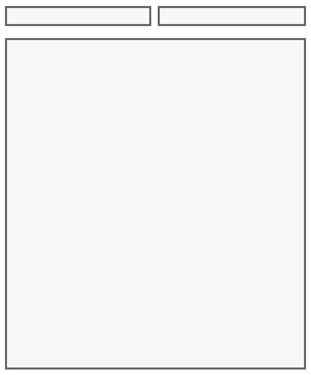


Figure 64. View Search

Includes Tree

```
CCView_Search.swt

    CCStylesChoice.swt

    CCHTMLFooter.swt
```

Table 57 lists the mappable items for this template.

Table 57. 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 65 for an example.

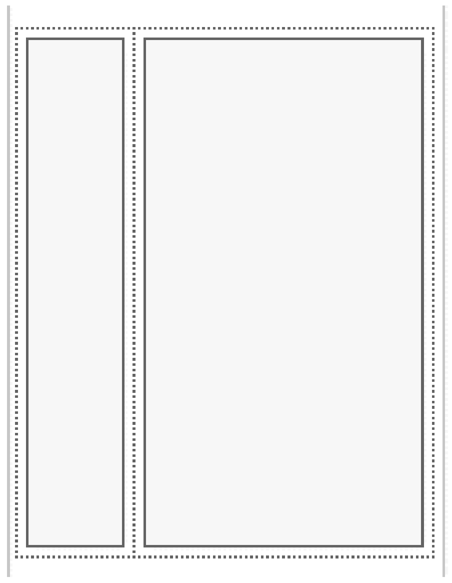


Figure 65. View Tree

The view also supports noncontext views as tabs.

Includes Tree

```
CCViewTree.swt
   CCHTMLHeader.swt
      CCStylesChoice.swt
   CCThreadbar.swt
   CCViewbarAll_Tabs_DropList.swt
   CCHTMLFooter.swt
```

Table 58 lists the mappable items for this template.

Table 58. 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 66 for an example.

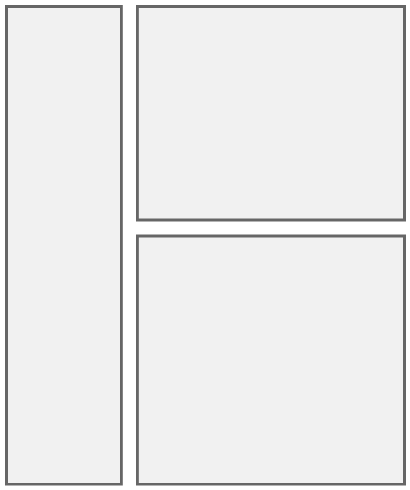


Figure 66. View Tree 2

The view also supports noncontext views as tabs.

Includes Tree

```
CCViewTree2.swt

    CCHTMLHeader.swt

        CCStylesChoice.swt

    CCThreadbar.swt

    CCHTMLFooter.swt
```

Table 59 lists the mappable items for this template.

Table 59. 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, a message bar, the search center, the persistent customer dashboard, and the iHelp pane. To understand how the frames are organized and what conditions control their display, see the following template called CCPageContainer.swt.

- "Page Container" on page 309
- "CC Container Page Logic" on page 310

Page Container

SWT Filename: CCPageContainer.swt

The page container template defines the setup for all frames within the application. Standard frames include banner, screen bar, view bar, and content. Optional frames include toolbar, message bar, and dashboard. Display of optional frames is evaluated at run time 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 Siebel Tools expects them to be mapped to the page container.

Includes Tree

```
CCPageContainer.swt

    CCStylesChoice.swt

    CCFrameBanner.swt

    CCFrameViewbar.swt

        CCStylesChoice.swt

    CCFrameToolbar.swt

        CCStylesChoice.swt

    CCFrameThreadbar.swt

        CCStylesChoice.swt

        CCThreadbar.swt

    CCFrameScreenbar.swt

        CCStylesChoice.swt

        CCScreenbar_Tabs.swt

    CCFrameContentHI.swt

    CCFrameMsgbar.swt
```

```
        CCStylesChoice.swt
```

Table 60 lists the mappable items for this template.

Table 60. 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

```
CCFrameContent_Logic.swt

    CCFrameContent_VSDT.swt

    CCFrameContent_VSD.swt

    CCFrameContent_VST.swt

    CCFrameContent_VS.swt

    CCFrameContent_VDT.swt

    CCFrameContent_VD.swt

    CCFrameContent_VT.swt

    CCFrameContent_V.swt
```

Specialized Applet Templates

Applet Advanced Search

SWT Filename: CCAppletSearchAdvanced.swt

Includes Tree:

```
CCAppletSearchAdvanced.swt

    CCFormSearch.swt
```

Table 61. Mappable Items

ID	Description
91	Inside Applet Help Text
132	Menu
133	New
141-142	Control

Table 61. Mappable Items

ID	Description
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 closes the dashboard. See Figure 67 for an example of the Applet Dashboard template.

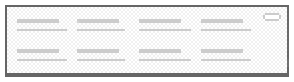


Figure 67. Applet Dashboard

Includes Tree

`CCAppletDashboard.swt`

Table 62 lists the mappable items for this template.

Table 62. 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

Table 62. Mappable Items

ID	Description
1802	Field
2200	Label
2201	Label
2202	Label
2300	Field
2301	Field
2302	Field
2700	Label
2701	Label
2702	Label
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 63 lists the mappable items for this template.

Table 63. Mappable Items

ID	Description
91	Inside Applet Help Text
132	Menu
133	New

Table 63. Mappable Items

ID	Description
141-142	Control
143	Control
1101-1130	Label; Field

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 list 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 64 lists the mappable items for this template.

Table 64. Mappable Items

ID	Description
90	...
91	Inside Applet Help Text
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 box, 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
```

Table 65 lists the mappable items for this template.

Table 65. 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
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	Error Message
1100	Outside Applet Help Text

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 68 for an example.

Figure 68. Applet Salutation

Includes Tree

```
CCAppletSalutation.swt

    CCApplet_Spacer.swt
```

Table 66 lists the mappable items for this template.

Table 66. 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 69 for an example.

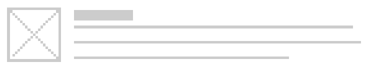


Figure 69. Applet Salutation (Graphical)

Includes Tree

```
CCAppletSalutationGraphical.swt

    CCApplet_Spacer.swt
```

Table 67 lists the mappable items for this template.

Table 67. 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 68 lists the mappable items for this template.

Table 68. 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

Table 68. Mappable Items

ID	Description
150-152	Control
160-164	Control
580	New - shows only in SI
599	Save - shows only in HI
1100	Group Label
1101-1120	Link
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.swt

This is a specialized applet for creating the send mail pop-up list.

Includes Tree

```
CCAppletSendMail.swt

    CCButtons.swt

    CCPopupButtonsBottom.swt
```

Table 69 lists the mappable items for this template.

Table 69. 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
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

    CCButtons.swt

    CCPopupButtonsBottom.swt
```

Table 70 lists the mappable items for this template.

Table 70. 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
598	Save
1200	Label

eActivityGanttChart Applet

SWT Filename: CCAppletActivityGanttChart.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
```

Table 71 lists the mappable items for this template.

Table 71. 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
```

Table 72 lists the mappable items for this template.

Table 72. 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 73 lists the mappable items for this template.

Table 73. 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 73. Mappable Items

ID	Description
306-314	Label; Control
405	Control
555	Label
2101-2130	...

Search Applet

SWT Filename: CCAppletSearchBasic.swt

This applet is used in Search Center. It defines the bottom portion of the Search Center.

Includes Tree

```
CCAppletSearchBasic.swt

    CCFormSearch.swt
```

Table 74 lists the mappable items for this template.

Table 74. 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 75 lists the mappable items for this template.

Table 75. 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
155	Control
156	Control
1201	Replacement Word Label
1211	Suggested Word Label
1221	Dictionary Label
1300	Text Segment

Specialized Views

- "View Dashboard" on page 325

- "View SME Segment Detail" on page 325

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

        CCStylesChoice.swt

    CCHTMLFooter.swt
```

View SME Segment Detail

SWT Filename: CCViewSegmentDetail.swt

This is a specialized view used in tree and expression builder. See Figure 70 for an example.

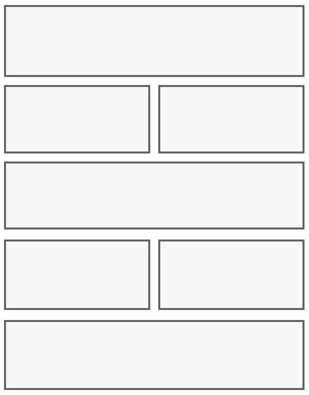


Figure 70. View SME Segment Detail

Includes Tree:

```
CCViewSegmentDetail.swt

    CCHTMLHeader.swt

        CCStylesChoice.swt

    CCThreadbar.swt
```

```
CCViewbar_Tabs.swt

CCHTMLFooter.swt
```

Table 76 lists the mappable items for this template.

Table 76. Mappable Items

ID	Description
1	Parent Applet
2	Tree Applet
3	Java Applet

7 Siebel Templates for Customer Applications

Most of the customer applications (such as Siebel eSales and Siebel eService) use a set of customer application (standard interactivity) templates along with a limited subset of employee (high interactivity) templates for messages and greetings.

NOTE: There are two types of applications for partner relationship management: PRM Portal and PRM Manager. PRM Manager can use high interactivity, and all templates shown through it adapt to the employee application look and feel as described in Chapter 6, "Siebel Templates for Employee Applications." PRM Portal uses standard interactivity and adapts to the customer application color scheme as described in this chapter.

The links below provide access to topics with examples of the applet and view templates used in customer applications:

- "Overview of UI Elements" on page 327
- "Applet Template Visual Reference" on page 328
- "Applet Templates" on page 338
- "View Templates" on page 381
- "Page Containers" on page 392
- "Specialized Applets" on page 395

Overview of UI Elements

Table 77 gives an overview of user interface elements in customer applications.

Table 77. UI Elements in Customer Applications

Element	Description
Header	The header is composed of the following elements: the banner, the screen bar, and the view bar. 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.

Table 77. UI Elements in Customer Applications

Element	Description
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.*
Screen bar	The area that displays the primary tabs, which provide access to key areas of the applications.
View bar	The bottom frame of the header that is used to display the simple search applet in most customer applications. In eChannel, the view bar 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 view bar 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.
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 thread bar on pages deeper within an application's hierarchical structure.
Thread bar	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.

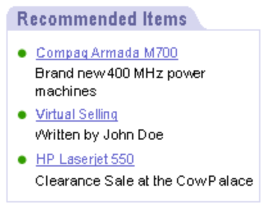
- "List Brief/Bullet" on page 329
- "List Brief/Bullet/Border" on page 329
- "List Brief/Bullet/Shaded" on page 330
- "List Brief/Image Bullet" on page 330
- "List Brief/Image Bullet/Border" on page 331
- "List Brief/Image Bullet/Shaded" on page 331
- "List Detailed/Image Bullet" on page 331

- "List Detailed/Image Bullet/Record Navigation" on page 332
- "List Detailed/Image Bullet/Record Navigation 2" on page 333
- "Form/Title Only" on page 334
- "List/Categorized/Bulleted/Tabbed" on page 334
- "List/Categorized/Bulleted" on page 334
- "Form/Item Detail 1" on page 334
- "Form/1-Column/Basic" on page 335
- "List/Light" on page 336
- "Form/Totals" on page 336
- "List Tabbed" on page 336
- "Form/4 Column" on page 336
- "Form/1-Column" on page 337
- "List/Horizontal" on page 337
- "Real-Time Shopping Cart" on page 338
- "Go To View List" on page 338

List Brief/Bullet

dCCAppletListBriefBullet.swt

For an example of the List Brief/Bullet applet, see Figure 71.



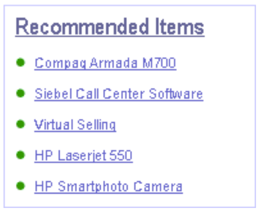
- Compaq Armada M700
  Brand new 400 MHz power machines
- Virtual Selling
  Written by John Doe
- HP Laserjet 550
  Clearance Sale at the Cow Palace

Figure 71. List Brief/Bullet Applet

List Brief/Bullet/Border

dCCAppletListBriefBulletBorder.swt

For an example of the List Brief/Bullet/Border applet, see Figure 72.



Recommended Items

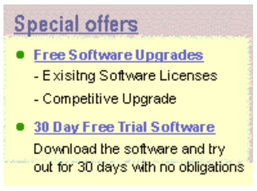
- Compaq Armada M700
- Siebel Call Center Software
- Virtual Selling
- HP Laserjet 550
- HP Smartphoto Camera

Figure 72. List Brief/Bullet/Border Applet

List Brief/Bullet/Shaded

dCCAppletListBriefBulletShade.swt

For an example of the List Brief/Bullet/Shaded applet, see Figure 73.



Special offers

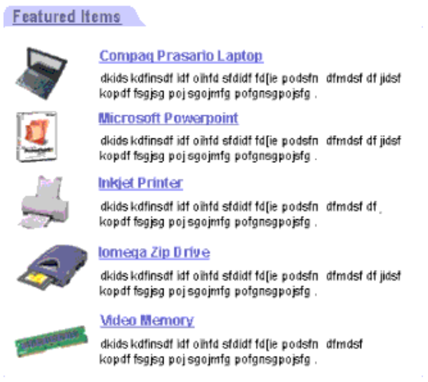
- Free Software Upgrades
  - Exisitng Software Licenses
  - Competitive Upgrade
- 30 Day Free Trial Software
  Download the software and try out for 30 days with no obligations

Figure 73. List Brief/Bullet/Shaded Applet

List Brief/Image Bullet

dCCAppletListBriefImgBullet.swt

For an example of the List Brief/Image Bullet applet, see Figure 74.



Featured Items

Compaq Prasario Laptop
dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdsf df jidsf kopdf fsgjsg poj sgojmfg pofgnsgpojsfg .

Microsoft Powerpoint
dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdsf df jidsf kopdf fsgjsg poj sgojmfg pofgnsgpojsfg .

Inkjet Printer
dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdsf df kopdf fsgjsg poj sgojmfg pofgnsgpojsfg .

Iomega Zip Drive
dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdsf df jidsf kopdf fsgjsg poj sgojmfg pofgnsgpojsfg .

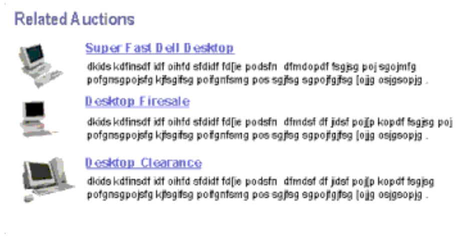
Video Memory
dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdsf kopdf fsgjsg poj sgojmfg pofgnsgpojsfg .

Figure 74. 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 75.



Related Auctions

Super Fast Dell Desktop

dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdopdf fsgjsg poj sgojmfg pofgnsgpojsfg kjfsgifsg poifgnfsmg pos sgjfsg sgpojfgjfsg [ojjg osjgsopjg .

Desktop Firesale

dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdsf df jidsf pojjp kopdf fsgjsg poj pofgnsgpojsfg kjfsgifsg poifgnfsmg pos sgjfsg sgpojfgjfsg [ojjg osjgsopjg .

Desktop Clearance

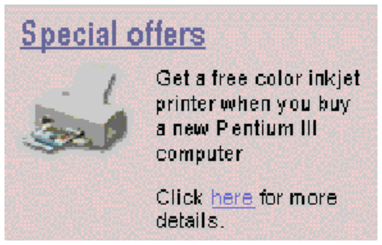
dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdsf df jidsf pojjp kopdf fsgjsg pofgnsgpojsfg kjfsgifsg poifgnfsmg pos sgjfsg sgpojfgjfsg [ojjg osjgsopjg .

Figure 75. List Brief/Image Bullet/Border Applet

List Brief / Image Bullet / Shaded

dCCAppletListBriefImgBulletShade.swt

For an example of the List/Image Bullet applet, see Figure 76.



Special offers

Get a free color inkjet printer when you buy a new Pentium III computer

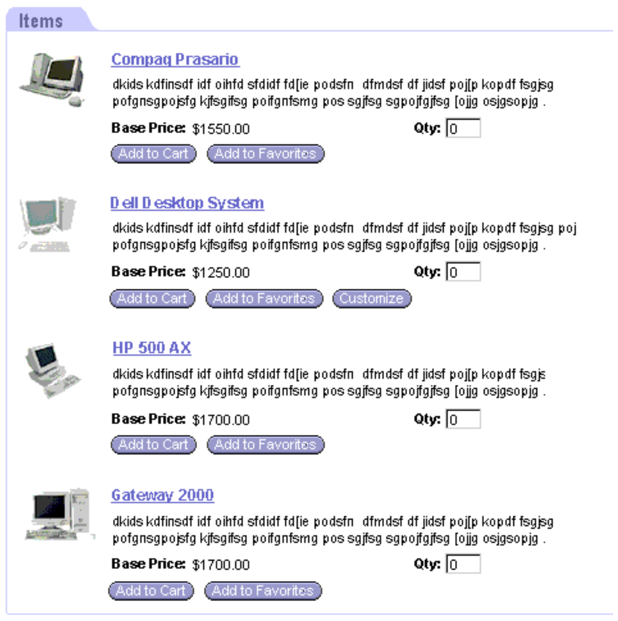
Click here for more details.

Figure 76. List Brief/Image Bullet/Shaded Applet

List Detailed / Image Bullet

dCCAppletListDetailedImgBullet.swt

For an example of the List Detailed/Image Bullet applet, see Figure 77.



Compaq Prasario

dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdsf df jidsf poj[p kopdf fsgjsg pofgnsgpojsfg kjfsgifsg poifgnfsmg pos sgjfsg sgpojfgjfsg [ojjg osjgsopjg .

Base Price: $1550.00 **Qty:** 0

Dell Desktop System

dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdsf df jidsf poj[p kopdf fsgjsg poj pofgnsgpojsfg kjfsgifsg poifgnfsmg pos sgjfsg sgpojfgjfsg [ojjg osjgsopjg .

Base Price: $1250.00 **Qty:** 0

HP 500 AX

dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdsf df jidsf poj[p kopdf fsgjs pofgnsgpojsfg kjfsgifsg poifgnfsmg pos sgjfsg sgpojfgjfsg [ojjg osjgsopjg .

Base Price: $1700.00 **Qty:** 0

Gateway 2000

dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdsf df jidsf poj[p kopdf fsgjsg pofgnsgpojsfg kjfsgifsg poifgnfsmg pos sgjfsg sgpojfgjfsg [ojjg osjgsopjg .

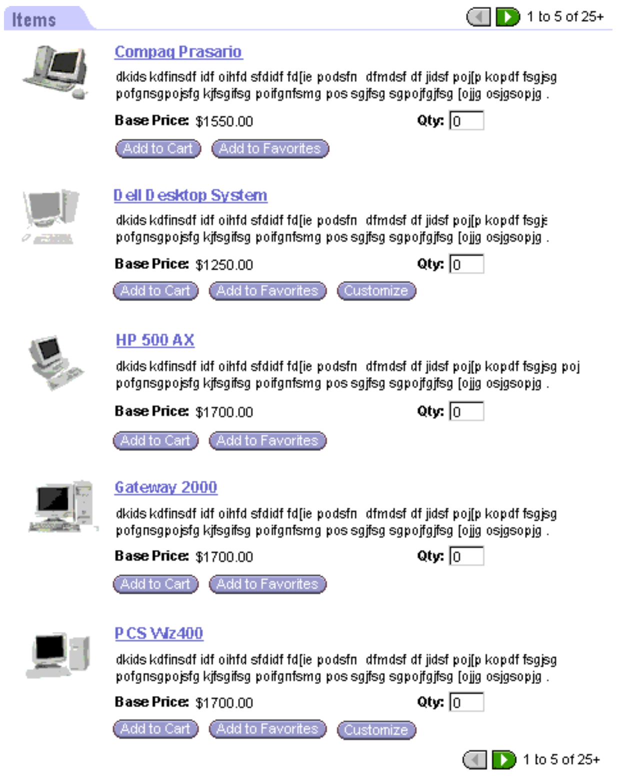
Base Price: $1700.00 **Qty:** 0

Figure 77. 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 78.



Compaq Prasario
dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdsf df jidsf poj[p kopdf fsgjsg pofgnsgpojsfg kjfsgifsg poifgnfsmg pos sgjfsg sgpojfgjfsg [ojjg osjgsopjg .
Base Price: $1550.00 **Qty:** 0

Dell Desktop System
dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdsf df jidsf poj[p kopdf fsgjs pofgnsgpojsfg kjfsgifsg poifgnfsmg pos sgjfsg sgpojfgjfsg [ojjg osjgsopjg .
Base Price: $1250.00 **Qty:** 0

HP 500 AX
dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdsf df jidsf poj[p kopdf fsgjsg poj pofgnsgpojsfg kjfsgifsg poifgnfsmg pos sgjfsg sgpojfgjfsg [ojjg osjgsopjg .
Base Price: $1700.00 **Qty:** 0

Gateway 2000
dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdsf df jidsf poj[p kopdf fsgjsg pofgnsgpojsfg kjfsgifsg poifgnfsmg pos sgjfsg sgpojfgjfsg [ojjg osjgsopjg .
Base Price: $1700.00 **Qty:** 0

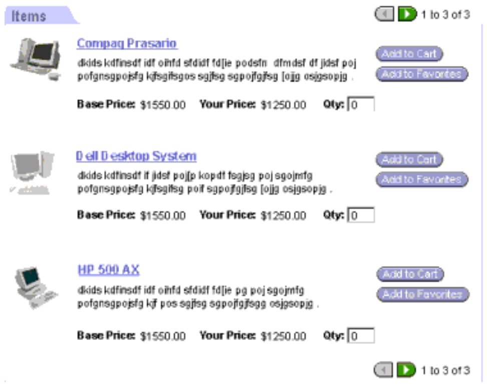
PCS Wz400
dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdsf df jidsf poj[p kopdf fsgjsg pofgnsgpojsfg kjfsgifsg poifgnfsmg pos sgjfsg sgpojfgjfsg [ojjg osjgsopjg .
Base Price: $1700.00 **Qty:** 0

Figure 78. 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 79.



Compaq Prasario
dkids kdfinsdf idf oihfd sfdidf fd[ie podsfn dfmdsf df jidsf poj pofgnsgpojsfg kjfsgifsgos sgjfsg sgpojfgjfsg [ojjg osjgsopjg .
Base Price: $1550.00 **Your Price:** $1250.00 **Qty:** 0

Dell Desktop System
dkids kdfinsdf if jidsf poj[p kopdf fsgjsg poj sgojmfg pofgnsgpojsfg kjfsgifsg poif sgpojfgjfsg [ojjg osjgsopjg .
Base Price: $1550.00 **Your Price:** $1250.00 **Qty:** 0

HP 500 AX
dkids kdfinsdf idf oihfd sfdidf fd[ie pg poj sgojmfg pofgnsgpojsfg kjf pos sgjfsg sgpojfgjfsgg osjgsopjg .
Base Price: $1550.00 **Your Price:** $1250.00 **Qty:** 0

Figure 79. List Detailed/Image Bullet/Record Navigation 2 Applet

Form / Title Only

dCCAppletFormTitle.swt

For an example of the Form/Title Only applet, see Figure 80.

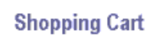


Figure 80. Form/Title Only Applet

List / Categorized / Bulleted / Tabbed

dCCAppletListCategorizedBulletTab.swt

For an example of the List/Categorized/Bulleted/Tabbed applet, see Figure 81.

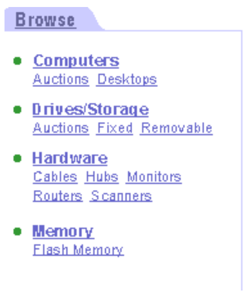


Figure 81. List/Categorized/Bulleted/Tabbed Applet

List / Categorized / Bulleted

dCCAppletListCategorizedBullet.swt

For an example of the List/Categorized/Bulleted applet, see Figure 82.

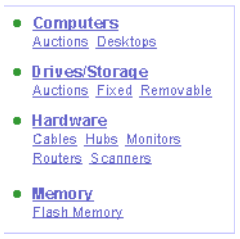
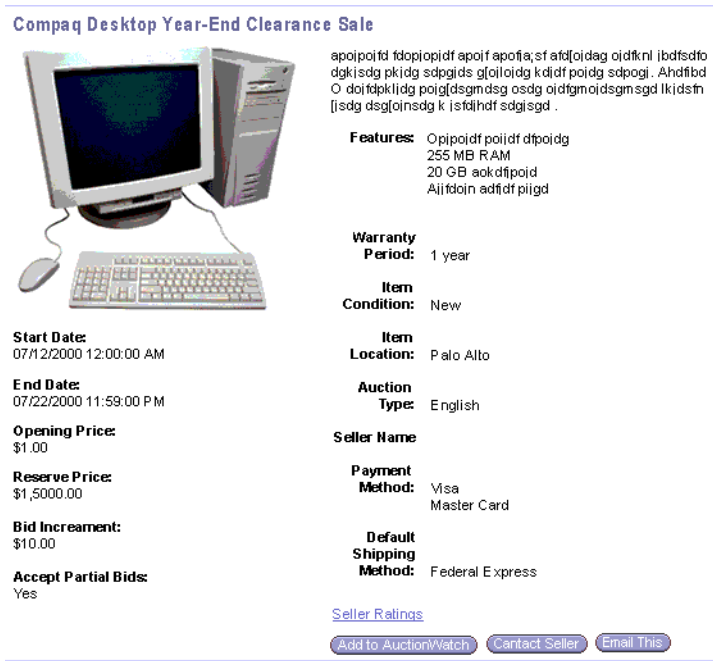


Figure 82. List/Categorized/Bulleted Applet

Form / Item Detail 1

dCCAppletFormItemDetail.swt

For an example of the Form/Item Detail 1 Applet template, see Figure 83.



Compaq Desktop Year-End Clearance Sale

apoipoifd fdopiopidf apoif apofia;sf afd[oidag oidfknl ibdfsdfo dgkisdg pkidg sdpgids g[oiloidg kdidf poidg sdpogi. Ahdfibd O doifdpklidg poig[dsgmdsg osdg oidfgmoidsgmsgd lkidsfn [isdg dsg[oinsdg k isfdihdf sdgisgd .

Features: Opipoidf poiidf dfpoidg
255 MB RAM
20 GB aokdfipoid
Aiifdoin adfidf piigd

Warranty Period: 1 year

Item Condition: New

Item Location: Palo Alto

Auction Type: English

Seller Name

Payment Method: Visa
Master Card

Default Shipping Method: Federal Express

Start Date:
07/12/2000 12:00:00 AM

End Date:
07/22/2000 11:59:00 PM

Opening Price:
$1.00

Reserve Price:
$1,5000.00

Bid Increament:
$10.00

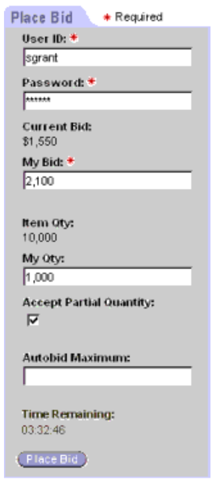
Accept Partial Bids:
Yes

Figure 83. Form/Item Detail 1 Applet

Form/1-Column/Basic

dCCAppletFormBasic.swt

For an example of the Form/1-Column/Basic applet, see Figure 84.



Place Bid * Required

User ID: *
sgrant

Password: *
******

Current Bid:
$1,550

My Bid: *
2,100

Item Qty:
10,000

My Qty:
1,000

Accept Partial Quantity:
✓

Autobid Maximum:

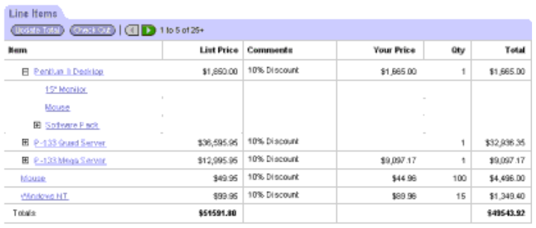
Time Remaining:
03:32:46

Figure 84. Form/1-Column/Basic Applet

List / Light

dCCAppletListLight.swt

For an example of the List/Light template, see Figure 85.



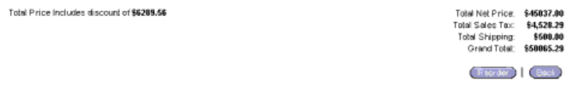
Item	List Price	Comments	Your Price	Qty	Total
⊟ Pentium II Desktop	$1,850.00	10% Discount	$1,665.00	1	$1,665.00
15" Monitor					
Mouse					
⊞ Software Pack					
⊞ P-133 Quad Server	$36,595.95	10% Discount		1	$32,936.35
⊞ P-133 Mega Server	$12,995.95	10% Discount	$9,097.17	1	$9,097.17
Mouse	$49.95	10% Discount	$44.96	100	$4,496.00
Windows NT	$99.95	10% Discount	$89.96	15	$1,349.40
Totals	$51591.80				$49543.92

Figure 85. List/Light Applet

Form / Totals

dCCAppletFormTotals.swt

For an example of the Form/Totals template, see Figure 86.



Total Price Includes discount of $6289.56

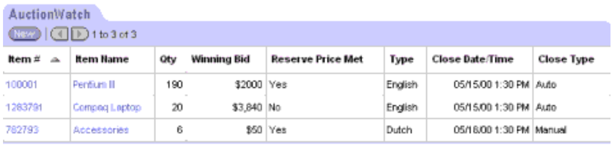
Total Net Price:	$45037.00
Total Sales Tax:	$4,528.29
Total Shipping:	$500.00
Grand Total:	$50065.29

Figure 86. Form/Totals Applet

List Tabbed

dCCAppletListTabbed.swt

For an example of the List Tabbed applet, see Figure 87.



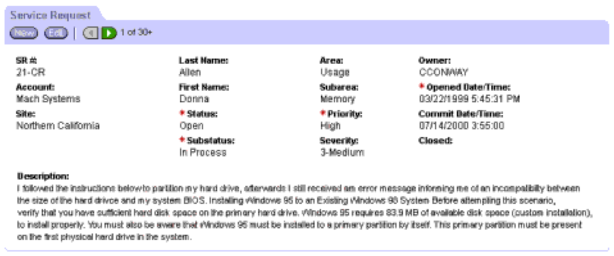
Item #	Item Name	Qty	Winning Bid	Reserve Price Met	Type	Close Date/Time	Close Type
100001	Pentium II	190	$2000	Yes	English	05/15/00 1:30 PM	Auto
1283791	Compaq Laptop	20	$3,840	No	English	05/15/00 1:30 PM	Auto
782793	Accessories	6	$50	Yes	Dutch	05/18/00 1:30 PM	Manual

Figure 87. List Tabbed Applet

Form / 4 Column

dCCAppletForm4Col.swt (formerly dCCAppletForm2Col.swt)

For an example of the Form/4 Column applet, see Figure 88.



SR #: 21-CR
Account: Mach Systems
Site: Northern California
Last Name: Allen
First Name: Donna
Status: Open
Substatus: In Process
Area: Usage
Subarea: Memory
Priority: High
Severity: 3-Medium
Owner: CCONWAY
Opened Date/Time: 03/22/1999 5:45:31 PM
Commit Date/Time: 07/14/2000 3:55:00
Closed:

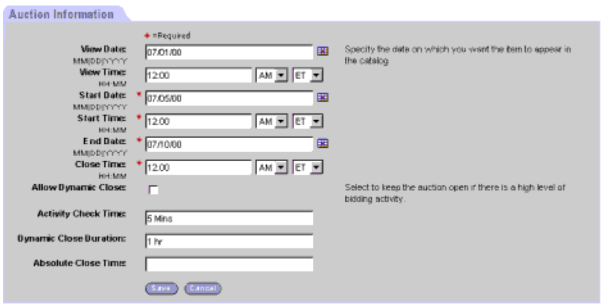
Description:
I followed the instructions below to partition my hard drive, afterwards I still received an error message informing me of an incompatibility between the size of the hard drivce and my system BIOS. Installing Windows 95 to an Existing Windows 98 System Before attempting this scenario, verify that you have sufficient hard disk space on the primary hard drive. Windows 95 requires 83.9 MB of available disk space (custom installation), to install properly. You must also be aware that Windows 95 must be installed to a primary partition by itself. This primary partition must be present on the first physical hard drive in the system.

Figure 88. Form/4 Column Applet

Form/1-Column

dCCAppletForm1Col.swt

For an example of the Form/1-Column applet, see Figure 89.



* =Required

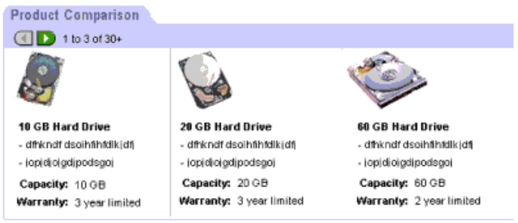
View Date: MM/DD/YYYY 07/01/00 — Specify the date on which you want the item to appear in the catalog
View Time: HH:MM 12:00 AM ET
Start Date: * MM/DD/YYYY 07/05/00
Start Time: * HH:MM 12:00 AM ET
End Date: * MM/DD/YYYY 07/10/00
Close Time: * HH:MM 12:00 AM ET
Allow Dynamic Close: ☐ — Select to keep the auction open if there is a high level of bidding activity.
Activity Check Time: 5 Mins
Dynamic Close Duration: 1 hr
Absolute Close Time:

Figure 89. Form/1-Column Applet

List/Horizontal

dCCAppletListHorizontal.swt

For an example of the List/Horizontal applet, see Figure 90.

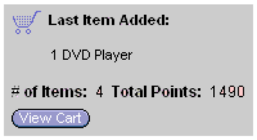


10 GB Hard Drive	**20 GB Hard Drive**	**60 GB Hard Drive**
- dfhkndf dsoihfihfdlk|df|	- dfhkndf dsoihfihfdlk|df|	- dfhkndf dsoihfihfdlk|df|
- |op|d|o|gd|podsgo|	- |op|d|o|gd|podsgo|	- |op|d|o|gd|podsgo|
Capacity: 10 GB	**Capacity:** 20 GB	**Capacity:** 60 GB
Warranty: 3 year limited	**Warranty:** 3 year limited	**Warranty:** 2 year limited

Figure 90. List/Horizontal Applet

Real-Time Shopping Cart

For an example of the Real-Time Shopping Cart applet, see Figure 91.


Last Item Added:

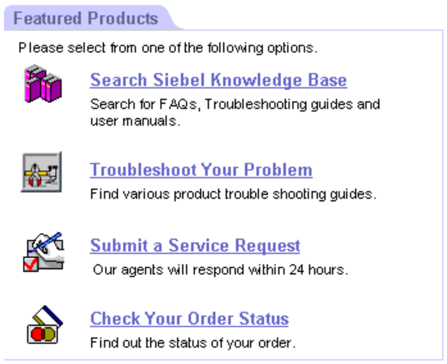
1 DVD Player

# of Items: 4 Total Points: 1490

Figure 91. Real-Time Shopping Cart

Go To View List

For an example of the Go To View List applet, see Figure 92.


Featured Products

Please select from one of the following options.

Search Siebel Knowledge Base
Search for FAQs, Troubleshooting guides and user manuals.

Troubleshoot Your Problem
Find various product trouble shooting guides.

Submit a Service Request
Our agents will respond within 24 hours.

Check Your Order Status
Find out the status of your order.

Figure 92. Go To View List

Applet Templates

- “DotCom Applet List Brief Bullet” on page 339
- “DotCom Applet List Brief Bullet/Border” on page 341
- “DotCom Applet List Brief Bullet / Shade” on page 342
- “DotCom Applet List Brief ImgBullet” on page 343
- “DotCom Applet List Brief ImgBullet / Border” on page 345
- “DotCom Applet List Brief ImgBullet / Shade” on page 346
- “DotCom Applet List Brief ImgBullet 2” on page 347
- “DotCom Applet List Categorized (No Tab)” on page 349
- “DotCom Applet List Categorized Bullet” on page 350
- “DotCom Applet List Categorized Bullet / Tabbed” on page 350
- “DotCom Applet List Categorized Tabbed” on page 351

- "DotCom Applet List Categorized TOC" on page 353
- "DotCom Applet List Detailed ImgBullet" on page 353
- "DotCom Applet List Detailed ImgBullet RecNav" on page 354
- "DotCom Applet List Detailed ImgBullet RecNav2" on page 356
- "DotCom Applet List Horizontal" on page 357
- "DotCom Applet List Light" on page 359
- "DotCom Applet List Search Results" on page 361
- "DotCom Applet List Subcategory" on page 362
- "DotCom Applet List Subcategory 1 Per Row" on page 363
- "DotCom Applet List Subcategory 4-Per-Column" on page 364
- "DotCom Applet List Subcategory 6-Per-Column" on page 364
- "DotCom Applet List Subcategory Indented" on page 365
- "DotCom Applet List Tabbed" on page 366
- "DotCom List Merged (Base/EditList)" on page 368
- "DotCom Applet Form 1-Column" on page 369
- "DotCom Applet Form 2-Column" on page 371
- "DotCom Applet Form 4-Column" on page 374
- "DotCom Applet Form Item Detail" on page 376
- "DotCom Applet Form Search Top" on page 377
- "DotCom Applet Form Title" on page 378
- "DotCom Applet Links" on page 378
- "Dotcom Form 4-Col Merged (Base/Edit/New)" on page 379

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 93 for an example.

Figure 93. 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 78 lists the mappable items for this template.

Table 78. 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

Table 78. Mappable Items

ID	Description
157	Label
184	DrillDown Title
203	MinimizeApplet
204	MaximizeApplet
207	MoveAppletUp
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 94 for an example.

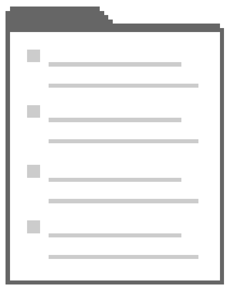


Figure 94. DotCom Applet List Brief Bullet/Border

Includes Tree

```
dCCAppletListBriefBulletBorder.swt

    CCApplet_Spacer.swt

    dCCListTitleNoRule.swt
```

```
dCCListBodyBullet.swt
```

Table 79 lists the mappable items for this template.

Table 79. 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 95 for an example.

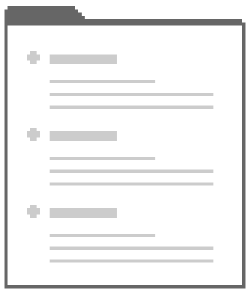


Figure 95. DotCom Applet List Brief Bullet/Shade

Includes Tree

```
dCCAppletListBriefBulletShaded.swt

    CCApplet_Spacer.swt

    CCTogglebar_drop.swt

    dCCListTitleNoRule.swt

    dCCListBodyBullet.swt
```

Table 80 lists the mappable items for this template.

Table 80. 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
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 96 for an example.

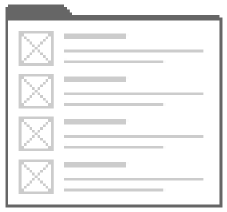


Figure 96. DotCom Applet List Brief ImgBullet

Includes Tree

```
dCCAppletListBriefImgBullet.swt

    CCApplet_NamedSpacer.swt

    dCCTitle_Portal.swt

        CCLayoutButtons.swt

    CCTogglebar_drop.swt

    dCCListTitleNoRule.swt

    dCCListBodyImgBullet.swt
```

Table 81 lists the mappable items for this template.

Table 81. 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

Table 81. Mappable Items

ID	Description
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 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 97 for an example.

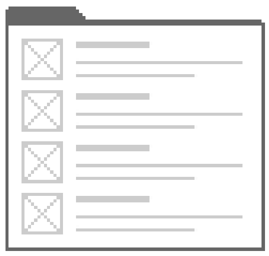


Figure 97. DotCom Applet List Brief ImgBullet/Border

Includes Tree

```
dCCAppletListBriefImgBulletBorder.swt

    dCCListTitleNoRule.swt

    dCCListBodyImgBullet.swt
```

Table 82 lists the mappable items for this template.

Table 82. 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 98 for an example.

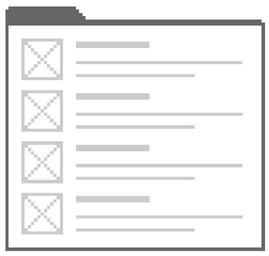


Figure 98. DotCom Applet List Brief ImgBullet/Shade

Includes Tree

```
dCCAppletListBriefImgBulletShaded.swt

    dCCListTitleNoRule.swt

    dCCListBodyImgBullet.swt
```

Table 83 lists the mappable items for this template.

Table 83. 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 2

SWT Filename: dCCAppletListBriefImgBullet2.swt

This template creates a bulleted list of records with record title, image, and a brief description. See Figure 99 for an example.

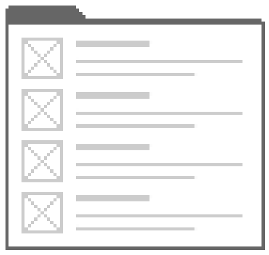


Figure 99. DotCom Applet List Brief ImgBullet 2

Includes Tree

```
dCCAppletListBriefImgBullet2.swt
    CCApplet_NamedSpacer.swt
    dCCTitle_Mapped.swt
        CCLayoutButtons.swt
    CCTogglebar_drop.swt
    dCCListBodyImgBullet2.swt
```

Table 84 lists the mappable items for this template.

Table 84. 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)

Table 84. 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 Categorized (No Tab)

SWT Filename: dCCAppletListCategorizedNoTab.swt

This template creates top-level items in an hierarchical list. See Figure 100 for an example.

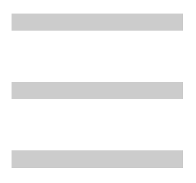


Figure 100.DotCom Applet List Categorized (No Tab)

Includes Tree

dCCAppletListCategorizedNoTab.swt

Table 85 shows the mappable items for this template.

Table 85. 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 101 for an example.

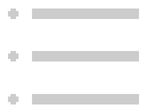


Figure 101.DotCom Applet List Categorized Bullet

Includes Tree

```
dCCAppletListCategorizedBullet.swt

    dCCListCategorized.swt
```

Table 86 shows the mappable items for this template.

Table 86. 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 102 for an example.

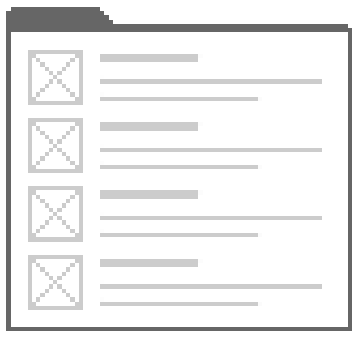


Figure 102.DotCom Applet List Categorized Bullet/Tabbed

Includes Tree

```
dCCAppletListCategorizedBulletTab.swt

    CCApplet_Spacer.swt

    CCTitle.swt

    dCCButtons_List.swt

    dCCListCategorized.swt
```

Table 87 shows the mappable items for this template.

Table 87. Mappable Items

ID	Description
2	Back
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-111	Control
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 hierarchical list. Items are surrounded by the standard applet treatment. See Figure 103 for an example.

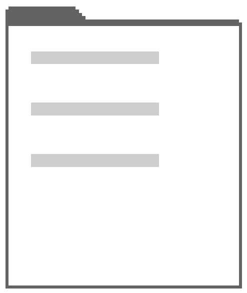


Figure 103. DotCom Applet List Categorized Tabbed

Includes Tree

```
dCCAppletListCategorizedTab.swt

    CCApplet_Spacer.swt

    CCTitle.swt

    dCCButtons_List.swt
```

Table 88 lists the mappable items for this template.

Table 88. 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 104 for an example.

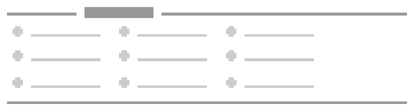


Figure 104.DotCom Applet List Categorized TOC

Includes Tree

```
dCCAppletListCategorizedTOC.swt

    CCApplet_Spacer.swt
```

Table 89 lists the mappable items for this template.

Table 89. 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 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.

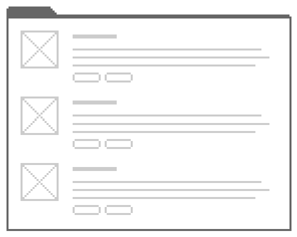


Figure 105.DotCom Applet List Detailed ImgBullet

Includes Tree

```
dCCAppletListDetailedImgBullet.swt
    CCApplet_Spacer.swt
    dCCTitle_Mapped.swt
        CCLayoutButtons.swt
    CCTogglebar_drop.swt
    dCCListBodyImgBulletDetailed.swt
```

Table 90 lists the mappable items for this template.

Table 90. Mappable Items

Id	Description
2	Small Image (30x30)
90	Title: DrillDown title; Label
142-143	Control
145-146	Control
157	Control
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 106 for an example.

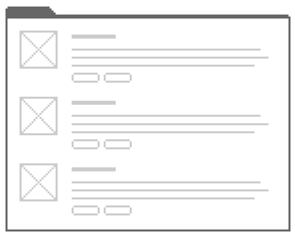


Figure 106.DotCom Applet List Detailed ImgBullet RecNav

Includes Tree

```
dCCAppletListDetailedImgBulletRecNav.swt

   CCApplet_Spacer.swt

   dCCTitle_RecNav.swt

      CCRecordNav.swt

   CCTogglebar_drop.swt

   dCCListBodyImgBulletDetailed.swt

   CCRecordNav.swt
```

Table 91 lists the mappable items for this template.

Table 91. Mappable Items

ID	Description
2	Small Image (30x30)
121	First
122	Previous
123	Next
124	Last
142-143	Control
145-146	Control
157-158	Control
501	Small Image (30x30)
502-503	Item Name
504-505	Label; Field

Table 91. Mappable Items

ID	Description
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 107 for an example.

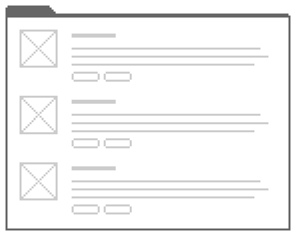


Figure 107.DotCom Applet List Detailed ImgBullet RecNav2

Includes Tree

```
dCCAppletListDetailedImgBulletRecNav2.swt
    CCApplet_Spacer.swt
    dCCTitle_RecNav.swt
        CCRecordNav.swt
    CCTogglebar_drop.swt
    dCCListBodyImgBulletDetailed2.swt
```

Table 92 lists the mappable items for this template.

Table 92. Mappable Items

ID	Description
2	Small Image (30x30)
121	First
122	Previous
123	Next

Table 92. Mappable Items

ID	Description
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 108 for an example of this template.

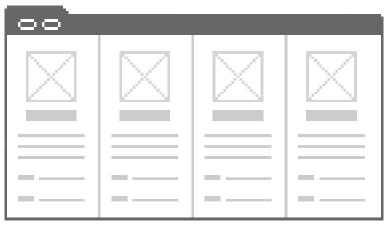


Figure 108.DotCom Applet List Horizontal

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
```

Table 93 lists the mappable items for this template.

Table 93. 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
510-512	Label; Field

Table 93. Mappable Items

ID	Description
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 Siebel 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 109 for an example of this template.

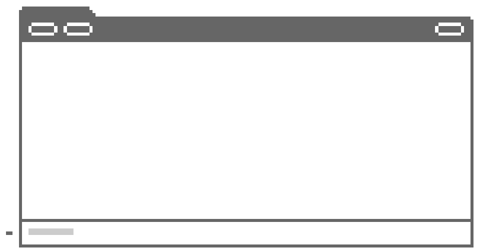


Figure 109.DotCom Applet List Light

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
```

Table 94 lists the mappable items for this template.

Table 94. 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
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 110 for an example of this template.

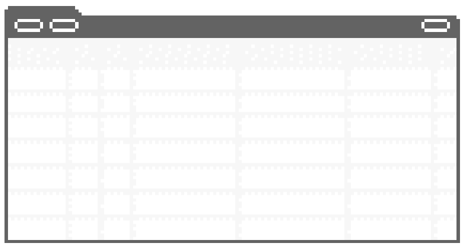


Figure 110.DotCom Applet List Search Results

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
```

Table 95 lists the mappable items for this template.

Table 95. Mappable Items

ID	Description
2	Control
106	Query
107	Go (ExecuteQuery)

Table 95. Mappable Items

ID	Description
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
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 111 for an example of this template.

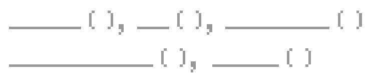


Figure 111.DotCom Applet List Subcategory

Includes Tree

dCCAppletListSubCategory.swt

Table 96 lists the mappable items for this template.

Table 96. Mappable Items

ID	Description
502	Field
503	Field

DotCom Applet List Subcategory 1 Per Row

SWT Filename: dCCAppletListSubCategory_1PerRow.swt

Figure 112 shows an example of this template.

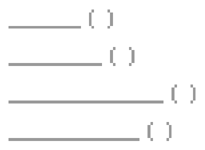


Figure 112.DotCom Applet List Subcategory 1 Per Row

Includes Tree

dCCAppletListSubCategory_1PerRow.swt

Table 97 lists the mappable items for this template.

Table 97. 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 113 for an example.

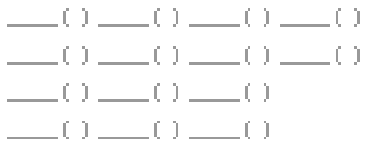


Figure 113.DotCom Applet List Subcategory 4-Per-Column

Includes Tree

`dCCAppletListSubCategory_4PerColumn.swt`

Table 98 lists the mappable items for this template.

Table 98. 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 114 for an example.

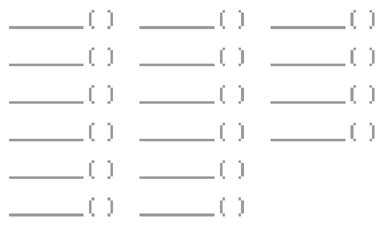


Figure 114.DotCom Applet List Subcategory 6-Per-Column

Includes Tree

`dCCAppletListSubCategory_6PerColumn.swt`

Table 99 lists the mappable items for this template.

Table 99. 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 115 for an example.

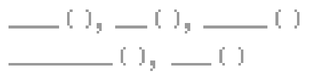


Figure 115.DotCom Applet List Subcategory Indented

Includes Tree

```
dCCAppletListSubCategoryIndented.swt

    CCApplet_NamedSpacer.swt
```

```
dCCListTitleNoRule.swt
```

Table 100 lists the mappable items for this template.

Table 100. 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
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 do not appear by default in the list, but appear in the columns displayed dialog. See Figure 116 for an example of this template.

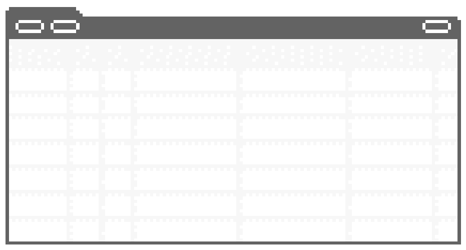


Figure 116.DotCom Applet List Tabbed

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
```

Table 101 lists the mappable items for this template.

Table 101. 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
139-141	Control
142-143	Control

Table 101. Mappable Items

ID	Description
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 117 for an example of this template.

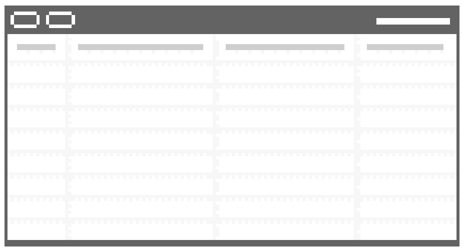


Figure 117.DotCom List Merged (Base/Edit)

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 102 lists the mappable items for this template.

Table 102. 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 118 for an example of this template.

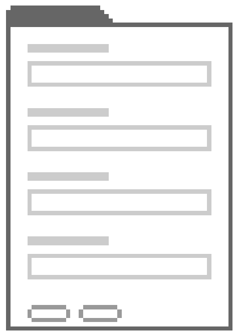


Figure 118.DotCom Applet Form 1-Column

Includes Tree

```
dCCAppletForm1Col.swt

    CCApplet_NamedSpacer.swt

    CCTitle_Named.swt

        CCTitle.swt

    CCTogglebar_drop.swt

    dCCForm1Col.swt

        dCCButtons_Form.swt
```

Table 103 lists the mappable items for this template.

Table 103. 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

Table 103. Mappable Items

ID	Description
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
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 119 for an example.

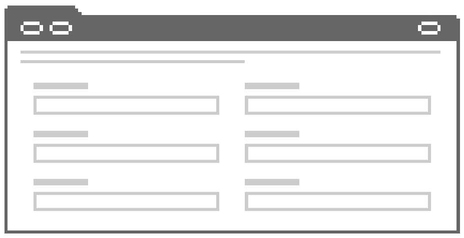


Figure 119.DotCom Applet Form 2-Column

Includes Tree

```
dCCAppletForm2Col.swt
    CCApplet_NamedSpacer.swt
    CCTitle_Named.swt
        CCTitle.swt
    CCTogglebar_drop.swt
    dCCFormButtonsTop.swt
        dCCButtons_Form.swt
        CCRecordNav.swt
        CCTogglebar_drop.swt
        CCFormButtonsTopRight.swt
    dCCForm2Col.swt
```

Table 104 lists the mappable items for this template.

Table 104. Mappable Items

ID	Description
2	Back
91-92	Inside Applet Help Text
106	Query
107	Go (ExecuteQuery)
108	Cancel (Query)
109-110	Control

Table 104. Mappable Items

ID	Description
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
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

Table 104. Mappable Items

ID	Description
2100-2104	Required; Label; Field
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 120 for an example.

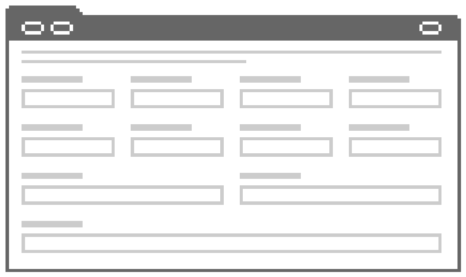


Figure 120.DotCom Applet Form 4-Column

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

    CCTitle_Named.swt

        CCTitle.swt

    CCTogglebar_drop.swt

    dCCFormButtonsTop.swt

        dCCButtons_Form.swt

        CCRecordNav.swt
        CCTogglebar_drop.swt

        CCFormButtonsTopRight.swt

    CCForm4ColBody.swt
```

Table 105 lists the mappable items for this template.

Table 105. 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
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

Table 105. Mappable Items

ID	Description
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

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 121 for an example of this template.

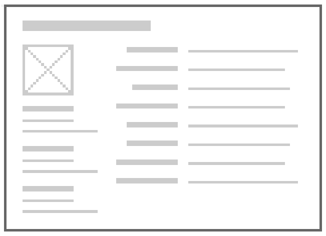


Figure 121.DotCom Applet Form Item Detail

Includes Tree

```
dCCAppletFormItemDetail.swt

    CCApplet_Spacer.swt

        dCCFormItemDetail.swt
```

Table 106 lists the mappable items for this template.

Table 106. 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 107 lists the mappable items for this template.

Table 107. 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 122 for an example.

Figure 122.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 123 for an example.

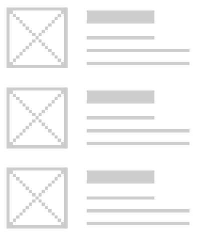


Figure 123.DotCom Applet Links

Includes Tree

```
dCCAppletLinks.swt

    CCApplet_Spacer.swt
```

Table 108 lists the mappable items for this template.

Table 108. 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 124 for an example.

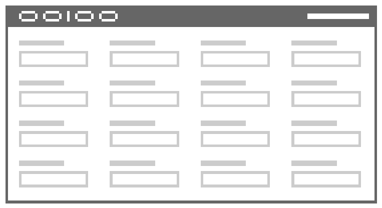


Figure 124.DotCom Form 4-Col Merged (Base/Edit/New)

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 109 lists the mappable items for this template.

Table 109. 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
1311-1315	Required; Label; 2-Column Wide Field

Table 109. Mappable Items

ID	Description
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" on page 382
- "DotCom View 25 50 25" on page 383
- "DotCom View 25 50 25 Home" on page 384
- "DotCom View 50 50" on page 385
- "DotCom View 66 33" on page 386
- "DotCom View Admin" on page 387
- "DotCom View Basic" on page 388
- "DotCom View Detail" on page 389

- "DotCom View Detail MultiChild" on page 390
- "DotCom View Detail2" on page 391

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 125 for an example.

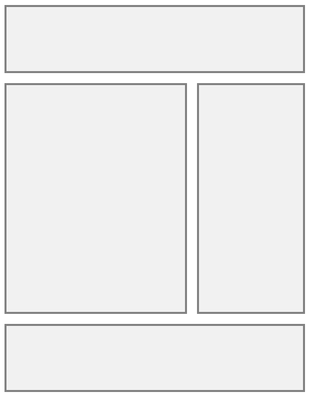


Figure 125.DotCom View 100 66 33 100

Includes Tree

```
dCCView_100_66_33_100.swt

    dCCHTMLHeader.swt

        CCStylesChoice.swt

    CCThreadbar.swt

    dCCHTMLFooter.swt
```

Table 110 lists the mappable items for this template.

Table 110. Mappable Items

ID	Description
101	Salutation Applet
102-106	Applet

Table 110. Mappable Items

ID	Description
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 126 for an example.

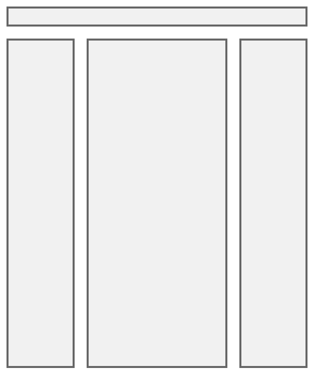


Figure 126.DotCom View 25 50 25

Includes Tree

```
dCCView_25_50_25.swt

    dCCHTMLHeader.swt

        CCStylesChoice.swt

    CCThreadbar.swt

    dCCHTMLFooter.swt
```

Table 111 lists the mappable items for this template.

Table 111. 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 127 for an example.

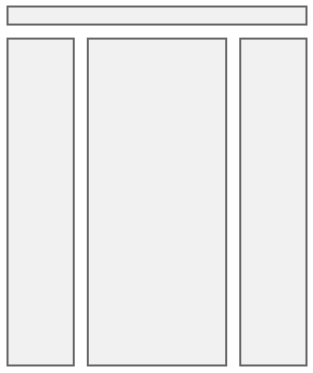


Figure 127.DotCom View 25 50 25 Home

Includes Tree

```
dCCView_25_50_25_home.swt

    dCCHTMLHeader.swt

        CCStylesChoice.swt

    CCThreadbar.swt

    dCCHTMLFooter.swt
```

Table 112 lists the mappable items for this template.

Table 112. 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 128 for an example.

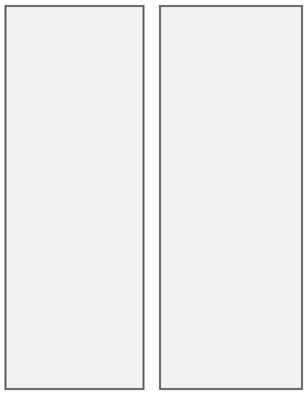


Figure 128.DotCom View 50 50

Includes Tree

```
dCCView_50_50.swt

    dCCHTMLHeader.swt

        CCStylesChoice.swt

    CCThreadbar.swt

    dCCHTMLFooter.swt
```

Table 113 lists the mappable items for this template.

Table 113. Mappable Items

ID	Description
101	Salutation Applet
102-111	Applet
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 129 for an example.

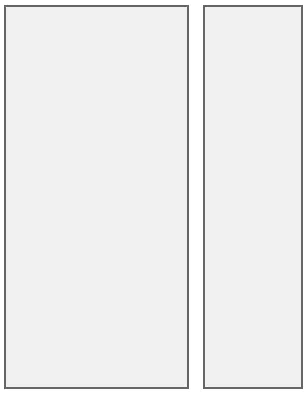


Figure 129.DotCom View 66 33

Includes Tree

```
dCCView_66_33.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; 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 130 for an example. It is useful for admin views that need to display nonrelated views that are not easily categorized.

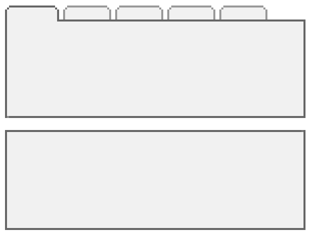


Figure 130.DotCom View Admin

Includes Tree

```
dCCViewAdmin1.swt

    dCCHTMLHeader.swt

        CCStylesChoice.swt

    CCThreadbar.swt

    dCCSubViewbar_Tabs.swt

        CCApplet_Spacer.swt

    dCCHTMLFooter.swt
```

Table 115 lists the mappable items for this template.

Table 115. 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 131 for an example.

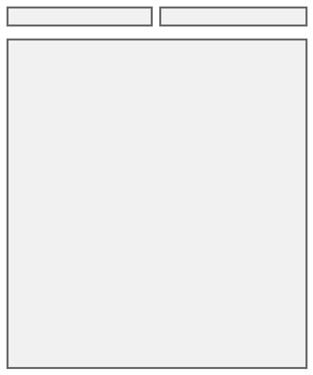


Figure 131.DotCom View Basic

Includes Tree

```
dCCView_Basic.swt

    dCCHTMLHeader.swt

        CCStylesChoice.swt

    CCThreadbar.swt
    dCCHTMLFooter.swt
```

Table 116 lists the mappable items for this template.

Table 116. 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 132 for an example.

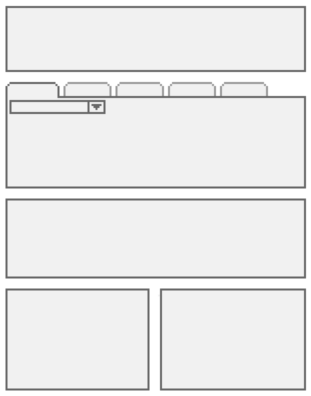


Figure 132.DotCom View Detail

Includes Tree

```
dCCViewDetail.swt
    dCCHTMLHeader.swt
        CCStylesChoice.swt
    CCThreadbar.swt
    dCCViewbar_Tabs.swt
        CCApplet_Spacer.swt
    dCCSubViewbar_Drop.swt

    dCCHTMLFooter.swt
```

Table 117 lists the mappable items for this template.

Table 117. 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 133 for an example.

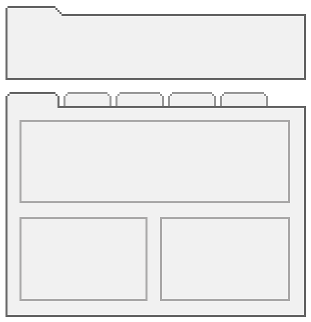


Figure 133.DotCom View Detail MultiChild

Includes Tree

```
dCCViewDetailMultiChild.swt

    dCCHTMLHeader.swt

        CCStylesChoice.swt

    CCThreadbar.swt
    dCCViewbar_Tabs.swt

        CCApplet_Spacer.swt

    dCCSubViewbar_Drop.swt

    dCCHTMLFooter.swt
```

Table 118 lists the mappable items for this template.

Table 118. 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 134 for an example.

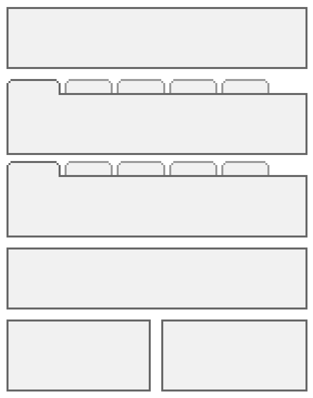


Figure 134.DotCom View Detail 2

Includes Tree

```
dCCViewDetail2.swt

    dCCHTMLHeader.swt

        CCStylesChoice.swt

    CCThreadbar.swt

    dCCViewbar_Tabs.swt

        CCApplet_Spacer.swt

    dCCSubViewbar_Tabs.swt

        CCApplet_Spacer.swt

    dCCHTMLFooter.swt
```

Table 119 lists the mappable items for this template.

Table 119. 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" on page 393
- "DotCom Page Container (Framed)" on page 393
- "DotCom Page Container (Hybrid)" on page 393
- "DotCom Page Container No Frames" on page 394

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.

For information about running customer applications without frames, see *Siebel eSales 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, screen bar, view bar, 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 120 lists the mappable items for this template.

Table 120. 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 view bar 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
    CCFrameContent_VD.swt
    CCFrameContent_V.swt
```

Table 121 lists the mappable items for this template.

Table 121. 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: dCCPageContainer_NoFrames.swt

This is the nonframed page container.

Includes Tree

```
dCCPageContainer_NoFrames.swt

    CCStylesChoice.swt

    CCThreadbar.swt
```

Table 122 lists the mappable items for this template.

Table 122. 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 123 lists the mappable items for this template.

Table 123. Mappable Items

ID	Description
91	Inside Applet Help Text
132	Menu
133	New
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
```

Table 124 lists the mappable items for this template.

Table 124. Mappable Items

ID	Description
2	...
91	Inside Applet Help Text
132-133	Control
141-142	Control
157	Control
158	Control
1100-1101	Label

Table 124. 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 125 lists the mappable items for this template.

Table 125. 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

Table 125. Mappable Items

ID	Description
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 126 lists the mappable items for this template.

Table 126. 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 127 lists the mappable items for this template.

Table 127. 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

Table 127. Mappable Items

ID	Description
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 128 lists the mappable items for this template.

Table 128. 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 129 lists the mappable items for this template.

Table 129. Mappable Items

ID	Description
91	Inside Applet Help Text
132	Menu
133	New
141-142	Control
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 130 lists the mappable items for this template.

Table 130. 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 131 lists the mappable items for this template.

Table 131. 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

8 Cascading Style Sheets

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.

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 view bar frame including the second-level show drop-down list, History drop-down list, Dashboard icon, Favorites drop-down list, 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.

Thread Bar

.threadbar, .threadbarDiv

Controls the link characteristics of the thread bar 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 that 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 if a login attempt produces 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 the 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 display 190 pixels wide. For customer 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 sizes to fit the width allotted to it. 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 the 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 "Applet Select" on page 408.

.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 calendar, a day for the Weekly calendar, and a week for the Monthly calendar.

.calendarDayBar

Defines the characteristics for the header portion of a given day in the Monthly calendar. This is used when the day is not the current view day for the Monthly calendar.

.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 calendar. This is activated when the day is the current view day for the Monthly calendar 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 view bar 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 the D-U-N-S 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.

9 Operators, Expressions, and Conditions

The 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. Operators with higher precedence are evaluated before operators with lower precedence. In addition, operators with equal precedence are evaluated left to right.

The levels of precedence for the various Siebel application operators are listed below. Lower level numbers indicate higher precedence.

Level	Operator
1	()
2	- (negation)
3	^ (exponentiation)
4	* (multiplication), / (division)
5	+ (addition), - (subtraction), NOT logical operator
6	AND logical operator

Level	Operator
7	OR logical operator
8	=, <>, >, <, >=, <= comparison operators

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
-	Negate	[Revenue] < -100
*	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 422.

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 provide 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' does 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 430
- “Using Julian Functions” on page 431
- “Calculated Field Rules” on page 431
- “Example of String Concatenation and the IIf Function” on page 432

- "Syntax for Predefault and Postdefault Fields" on page 432

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 132 describes the functions you can use in these expressions.

Table 132. Functions Used in Calculation 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).
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 132. Functions Used in Calculation 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 is predefaulted to this value, and never receives 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
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.

Table 132. Functions Used in Calculation Expressions

Function	Result Type	Query	Description
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.
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.

Table 132. Functions Used in Calculation Expressions

Function	**Result Type**	**Query**	**Description**

InvokeServiceMethod ("*ServiceName*", "*MethodName*", "*InputProp1=val1*, *InputProp2=val2*, ...", "*OutputProp*")	String	No	Invokes a business service from a calculated field and returns *OutputProp*.
			
			Use the following guidelines for invoking this method:
			
			■ This method requires an input argument in its third field, else the calculated field does not provide a return value and a parsing error results. If the business service method does not require an input argument, then you must provide an argument as a placeholder. For example, if `MyMethod` takes no input argument, then make a call similar to the following:
			
			`InvokeServiceMethod("MyService","MyMethod", "a=a","MyReturn")`
			
			■ To pass a field value, as opposed to a string literal, as an input argument, you must enclose the field name in brackets. Following is an example that provides the value in the `Name` and `Location` fields as input arguments:
			
			`InvokeServiceMethod("MyService","MyMethod", "prop1=eval([Name]),prop2=eval([Location])" ,"MyReturn")`
			
			NOTE: Values of the form `eval(`*`expression`*`)` are evaluated before the business service is called.
			
			■ The name of the return property in the calculated field, such as `MyReturn` in these examples, must match the name of a property in the business service output property set. If the method cannot be invoked due to this improper syntax, an error is not raised, and the calculated field will be null.
			
			NOTE: Do not expose a calculation expression that invokes a business service in a list applet. Doing so may result in poor performance because the business service repeatedly instantiates 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 Julian Day, for all dates in the 20th and 21st centuries.

Table 132. Functions Used in Calculation Expressions

Function	Result Type	Query	Description
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) that 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 OM - Resource Language Code server parameter found in the Administration - Server Configuration screen.
Left (*text*, *integer*)	String	Yes	Returns the leftmost *n* characters in the text string or field. For example, Left (“Adams”, 2) returns “Ad.”
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.

Table 132. Functions Used in Calculation Expressions

Function	**Result Type**	**Query**	**Description**
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.
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
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).

Table 132. Functions Used in Calculation Expressions

Function	Result Type	Query	Description
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 the currently logged-in employee (for example, 0-4432).
PositionName ()	String	Yes	Position name of the 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".
RowIdToRowIdNum ([*Id*])	String	Yes	Converts an alphanumeric row ID to a unique, pure numeric row ID in the Service Request business component.
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` returns 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 is “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 133 provides the syntax to be used in predefault fields.

Table 133. 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, 4/26/05). 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, 04/02/05 11:15:22).
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 is used.
System: LocalCurrency	String	Currency for this machine (for example, JPY).

Table 133. Predefault Values and Functions

Function	Result Type	Description

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`

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 is 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 (MVFs), include only those MVFs 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. The 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 the result set consists of every record for which 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, then EXISTS is automatically 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), then all records are returned for which VSILVER 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 an ascending/descending sort button, 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 441
- "Sorting Through the Object Property Sort Specification" on page 441
- "Sorting Through the User Interface" on page 441

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.

Index