SIEBEL 7
eBusiness

PRODUCT ADMINISTRATION GUIDE

VERSION 7.5

12-BD6PX7

JULY 2002

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

Contents

Chapter 3. Select a Configuration Method

Chapter 4. Basic Product Administration

Chapter 5. Product Classes

Chapter 6. Product Attributes

Chapter 7. Attributes with Business Component Domains

Chapter 8. Smart Part Numbers

Chapter 9. Product Bundles

Chapter 10. Build Customizable Products

Chapter 11. Release and Manage Customizable Products

Chapter 12. Customizable Product User Interface

Chapter 13. Customizable Product UI Properties

Chapter 14. Customizable Product Web Templates

Chapter 15. Customizable Product Resources

Chapter 16. Customizable Product Links

Chapter 17. Customizable Product Rule Designer

Chapter 18. Configuration Rule Template Reference

Chapter 19. Configuration Rule Assembly Language

Chapter 20. Customizable Product Scripts

Chapter 21. Multilingual Data

Chapter 22. Cache Management

Chapter 23. Technical Reference

Index

Introduction

This guide explains product administration procedures. This includes creating and managing customizable products. Customizable products are those that have components and are interactively configurable when creating a quote or when purchasing the product from a Web site.

Although job titles and duties at your company may differ from those listed in the following table, the audience for this guide consists primarily of employees in these categories:

Product Administrators	Persons responsible for defining and managing products and product lines.
Siebel Application Administrators	Persons responsible for planning, setting up, and maintaining Siebel applications.

How This Guide Is Organized

This guide describes how to create and manage products and product lines. The guide deals with two types of products: simple products and customizable products.

Simple products are those that do not have components that can be interactively configured. For example, you sell a model of telephone that has no components that can be configured at the time of quote or purchase. The telephone may have attributes that you can select, such as color. This is a simple product.

A customizable product is one that has configurable components. For example, you sell desktop computers that have several types of monitor and hard drive. Users can select which monitor and which hard drive they want at the time of purchase. Customizable products and their components can also have configurable attributes.

The chapter on basic product administration describes administration tasks common to both simple and customizable products. If your product lines contain only simple products, this chapter is intended to meet most of your product administration needs. The remaining chapters describe how to create and manage the product class system and how to create customizable products using Siebel eConfigurator.

The last chapter in the book is a technical reference and provides information on topics of interest to integrators and developers.

The organization of the guide is task-oriented. Each chapter describes a group of related tasks. The key concepts required to understand and apply these tasks are presented at the beginning of each chapter. The title of these conceptual topics begins with the word “Understanding” for example, “Understanding Classes.”

The remainder of each chapter contains the tasks. The titles of tasks begin with a gerund (an “ing” word), for example “Managing Attribute Values.” Task topics begin with conceptual material or facts to do the task correctly. If you do not find the information you need in the task topic, review the chapter’s conceptual topics.

The steps in each task are numbered, 1, 2,3 and so on. Processes are collections of tasks that must be performed in order. Processes are presented throughout the guide. The tasks in a process are numbered a, b, c and so on.

Revision History

Product Administration Guide, Version 7.5

Overview 1

This chapter provides information on how to navigate to views and screens. It also describes important product administration processes.

Screens, Views, and Navigation

Product Administration and Application Administration are the two screens you will use most frequently to perform the tasks in this guide. The Product Administration screen is where you create and manage both simple and customizable products.

The Application Administration screen is where you define the product classification system and its attributes. You also define lists of values and other things that support product definitions.

Navigate to the Product Administration and Application Administration screens by using the Site Map, located in the View menu. Both these screens contains several views. You can navigate to these views using the Site Map or by using drop-down menus within the screens.

When you create a record, stepping off of it causes the record to be saved automatically. However, after you create a record, you should open the list or form menu and click Save. This is good operating practice and minimizes errors in creating or editing records.

Included in many views is a More Info tab. When you click this tab, the view expands to display additional information about the record selected in the associated list. This information is displayed in a form. In some cases, this information can be even further expanded by clicking the show more button located in the upper-right corner of the form.

When you do queries, in some cases the query results display as one record in the More Info form. To see all the records returned by the query, click the More Info tab. The form contracts and is replaced by a list of all the query records.

The path syntax used throughout the Guide is based on an English language installation in Windows 2000 environment. Modify the path syntax as needed for other languages and operating systems.

Logging On as the Siebel Administrator

The Siebel database server installation script creates a Siebel administrator account that can be used to perform the tasks described in this guide. For more information, see *Siebel Server Installation Guide* and *Siebel Server Administration Guide*.

To log on as the Siebel administrator, start the application and log on using the user name and password assigned by your database administrator. Generally, the Siebel administrator connects to the server database.

CAUTION: Do not perform system administrative functions on your local database. Although there is nothing to prevent you from doing this, it can have serious results. Examples include: data conflicts, an overly large local database, a large number of additional transactions to route.

License Key Requirements

This guide describes basic product management tasks. It also describes how to use Siebel eConfigurator to create and manage customizable products. To use Siebel eConfigurator, you must have the appropriate license keys installed.

You access Siebel eConfigurator through the Customizable Product and Configuration Designer tabs in the Product administration screen.

Important Processes

A procedure (also called a task) is a group of one or more numbered steps that you perform to complete a defined task. For example, creating a product record is a procedure. This procedure contains several steps.

Processes are groups of procedures that you perform to accomplish important goals. For example, creating a hierarchy and defining attributes for classes are two procedures in the process for creating a product class system.

The procedures in the processes below correspond to those listed in the guide's table of contents. Do the procedures in the order in which they are presented in each process. These processes are guidelines for accomplishing important product administration tasks. Adapt them as needed to fit local operations.

The key processes in product administration are the following:

- Create a product class system
- Create a simple product
- Create a product that has attributes
- Create a customizable product

The processes below are made up of tasks, which are listed in the description of each process. The tasks in the processes correspond to tasks in the guide's table of contents. In a process, perform the tasks in the order in which they are presented.

These processes are guidelines for accomplishing important product administration tasks. Adapt them as needed to fit local operations.

Create a Product Class System

If your product lines includesproducts that have configurable attributes, you must complete this process.

a Create a class hierarchy

b Define attributes for classes

c Define lists of values (LOVs)

d Edit attribute definitions to add LOVs

Create a Simple Product

This process creates a simple product.

a Create a product record

b Associate the product with a price list

c Associate the product with a catalog

d Add the product to a product line

Create a Product that has Attributes

You create a product that has attributes by assigning it to a product class on which attributes have been defined. All products assigned to a product class inherit the attributes defined on the class. When you assign a customizable product to a class that has attributes, the customizable product as a whole inherits the attributes.

When the user creates a quote or purchases the product on a Web site, the product displays with its attributes, and the user can select the desired attribute values.

a Complete the Create a Product Class System process.

b Assign the product to a product class.

c Define attribute-based pricing adjustments using ePricer.

Create a Customizable Product

You create a customizable product by adding a work space to a simple product. Then you add other products, configuration rules, a customized user interface, and other features.

The components you add to a customizable product can themselves be customizable products. The user interface the system provides for selecting components, also provides for selecting attribute values.

Customizable products can have attributes. You do this by assigning the customizable product root to a product class on which attributes are defined. The components of the customizable product can also have attributes. You provide attributes to the components by assigning them to classes on which attributes are defined before adding the products to a customizable product.

a Complete the Create a Product Class System process.

b Assign the component products to product classes.

c Complete the Create Product process for the root of the customizable product. To provide attributes to the customizable product root, assign it to a product class on which attributes are defined. For example, you want to create a customizable product called Workstation Pro. This product will have components such as a monitor and disk drives. Create a product record for Workstation Pro. This product record is the product root. The Workstation Pro comes with a choice of 12 or 24 month warranty. You have defined an attribute called Warranty Type on the product class Workstations. This attribute lets users select which warranty they want. You would assign the Workstation Pro to this product class. The Workstation Pro then inherits this attribute.

d Create and lock a work space for the customizable product root.

e Add products (these are the components of the customizable product). These products have the user-configurable attributes of the product classes to which they are assigned.

f Create resources as needed.

g Create links as needed.

h Create configuration rules as needed.

i Validate the product and test all configuration rules

j Create a customized user interface for configuring the product as needed.

k Define specialized user interface properties as needed.

l Validate the product and test the user interface.

m Define pricing adjustments for attributes as needed using ePricer. See the *Pricing Administration Guide* to do this task.

n Define pricing adjustments for components as needed using ePricer. See the *Pricing Administration Guide* to do this task.

o Define a customizable product pricing model as needed using ePricer. See the *Pricing Administration Guide* to do this task.

p Validate the product and test the pricing on all items.

q Release the product.

Mapping eConfigurator 6.x Features to Release 7.x

2

This chapter provides a conceptual mapping of release 6.x eConfigurator to release 7.x eConfigurator features. Users of 6.x eConfigurator should use this chapter to help them understand the similarities and differences between 6.x and release 7.x eConfigurator.

The mappings are intended to present features that are functionally similar. That one feature maps into another does not mean that the release 7.x feature is exactly equivalent or works in exactly the same way.

The topics in this chapter are presented in roughly the same order as the topics in Version 6.2 of the Siebel eConfigurator Guide.

This chapter does not describe the features of the eConfigurator implemented using Siebel Interactive Designer. This method of creating customizable products is new at release 7.0.

Upgrading 6.x Models to 7.x

Upgrading 6.x models to 7.x requires planning and a thorough understanding of the upgrade process. Refer to *Siebel Interactive Selling Applications Upgrade Guide* before upgrading models. Also check Siebel SupportWeb for technical papers on eConfigurator upgrades.

Managing Models

The concept of a model in 6.x maps in release 7.x to a customizable product with a work space. Table 1 maps 6.x features to 7.x for managing models.

Table 1. Managing Models

Release 6.x	Release 7.x	Comment
Create a new model	Create a customizable product with a work space.	
Delete a model	Delete customizable product structure.	Reverts customizable product to a simple non-configurable product.
Copy a model	Copy a customizable product.	
Share a model	Add a customizable product to another customizable product.	
Lock a model	Lock a customizable product work space.	
Import a model	Import a customizable product.	
Export a model	Export a customizable product.	Only the latest released version can be exported.
Export a model version	Only the latest released version can be exported.	
Validate a model	Validate a customizable product.	
Release a model	Release a customizable product.	
Revert to released model	Revert to a released customizable product.	
Model versions	Customizable product versions.	
Model synchronization	Customizable product synchronization.	
Associating a model with a product	Not applicable	The structure of a customizable product is stored with the product definition. Associating a model with a model-product is no longer required.
Required start date	Customizable product required start date	The customizable product version does not become available to users until the specified date.

Designing the Catalog

The tree structure of catalogs and items in 6.x maps in release 7.x to a hierarchy of relationships within the customizable product. A relationship is roughly equivalent to a category and functions as a named group that contains one or more items. Table 2 maps 6.x features to 7.x for designing a catalog.

Table 2. Designing the Catalog

Release 6.x	Release 7.x	Comment
Design a tree structure	Create a hierarchy of relationships within a customizable product.	The hierarchy defines component relationships rather than being a grouping mechanism. Relationship definition includes cardinality (maximum quantity, minimum quantity, default quantity).
Create a category	Create a relationship.	A relationship is a named part of a customizable product. Relationships contain one or more items.
Add a product to a catalog	Add a product to a relationship.	
Set item sequence in a catalog	Set sequence of items in Product Designer and sequence of group in Product UI Designer.	
Hide items in the catalog	Can hide an item by not adding it to a UI group in the Product UI Designer.	All products included in a customizable product, plus all attributes, resources, and links can be made visible to users.
Show all excluded items	Select UI control in Product UI designer that displays all items.	Excluded items are unavailable.
Create virtual product	Replaced with hidden attributes.	Virtual product functionality can be created by defining product attributes and then marking them hidden. Hidden attributes do not appear in quotes, orders, or agreements.

Working with Properties

The properties feature has been replaced in release 7.x with attributes. Attributes are product characteristics that are defined for product classes. All products belonging to a class inherit the attributes of the class. Subclasses inherit the attributes of the parent class. You put products into the class hierarchy by assigning a class name in the product record. Attributes cannot be defined directly on a product. They must be inherited from the class to which the product belongs.

When you define an attribute, you can define the allowed values for the attribute. You can specify the allowed values, called an attribute domain, using a list of values or a range of values. The administrator can then set this value for an individual product so that it cannot be changed by the user.

Defining an attribute and setting its value so that it cannot be changed is functionally equivalent to defining a property and assigning it a value in 6.x.

Working with Resources

Resources are implemented in the same fashion for customizable products in release 7.x as they were for models in 6.x. You define resources in the Resource Designer and then write provide and consume rules that adjust the value of the resource. In release 7.x, you can also provide or consume amounts from a product attribute.

In release 7.x, to enforce a resource or attribute so that its value does not drop below zero, you write a configuration rule that constrains the range of allowed values.

Working with Linked Items

In release 7.x, you define links within the context of a specific customizable product. You do not define links in a single location and then associate the definition with a model, as in 6.x. Link definitions are added to a picklist so that you can add the definition to other customizable products. If you delete a link definition from the only customizable product in which it occurs, it is deleted from the picklist.

The things for which you can define a link, have not changed in release 7.x. You can define links on Siebel business components, the system date/time, and on the login ID of the current users.

Designing Rules and Logical Expressions

The 6.x Basic Rule Designer, Logic Designer, and Advanced Rules Designer have been replaced by the Rule Designer in release 7.x. The Rule Designer provides a series of natural-language rule templates that you can use to create rules of any complexity. You can also create and save your own rule templates.

The logical operators in the 6.x Logic Designer are provided in picklists associated with the rule templates.

A special rule template is provided to enter rules in eConfigurator Rule Language (renamed Rule Assembly Language in release 7.x). The operators and syntax in 6.x eConfigurator Rule Language are supported in release 7.x. Table 3 maps 6.x features to 7.x for designing rules.

Table 3. Designing Rules in the Rule Designer

Release 6.x	Release 7.x	Comment
Create rule in Basic Rule Designer	Create rule using rule template in Rule Designer.	Rules can be created with or without conditions.
Create expressions in Logic Designer	Use logical operators associated with rule templates in Rule Designer.	Rules can be created with or without conditions.
Create rule in Advanced Rules Designer	Enter rule into special template in Rule Designer.	6.x operators and syntax are supported in release 7.x.

Table 3. Designing Rules in the Rule Designer

Release 6.x	Release 7.x	Comment
Category-to-product rules	Class-to-product rules.	The product class must be part of a relationship.
Category-to-category rules	Class-to-class rules.	The product classes must part of relationships.
Product-to-product rules	Product-to-product rules.	
Copy and delete rules	Copy and delete rules.	Includes rules that are logical expressions.
Activate and deactivate rules	Activate and deactivate rules.	Includes rules that are logical expressions.
System generates rule explanations	System generates rule explanations or you can write explanations.	Includes rules that are logical expressions.
Rule Summary report	Rule Summary report.	
Enforce resource total by placing check mark in resource record	Write rules to prevent resources from having negative values.	
Syntax checker for eConfigurator Rule Language	Syntax checking is provided when building rules using the Assisted Advanced Rule template and all other rule templates.	eConfigurator Rule Language is called Rule Assembly Language in release 7.x.

Designing Scripts

In release 7.x, the full Siebel API is accessible from within a script. Refer to Siebel API documentation for more information on the Siebel API. Because the Siebel API is available, the number of eConfigurator-specific events and functions has been reduced in release 7.x.

In addition, the method for associating scripts with parts of a model has changed. In release 7.x, you associate a script with an item by writing the script on an event called for the product root. The event returns a matrix of records, one for each item that has changed in the solution. An item can be any product added to the customizable product from the product table. Events do not return changes to relationship quantities or resources.

If a customizable product contains other customizable products, another event is provided so you can write scripts on the child customizable product directly.

The Script Designer in release 7.x. does not provide a hierarchical tree display of the customizable product. In release 7.x, the Cfg ID of an item can no longer be passed as an argument. Instead, the name of the item as a string is passed. A name syntax is provided to allow you to uniquely specify a product name. Table 4 maps 6.x features to 7.x for designing scripts.

Table 4. Designing Scripts

Release 6.x	Release 7.x	Comment
Create scripts	Create scripts in the Script Designer.	
Script inheritance	Scripts are not associated directly with relationships and are not inherited.	
Copy, edit, and delete scripts	Copy, edit, and delete scripts.	
Siebel Visual Basic and Siebel eScript languages	Siebel Visual Basic and Siebel eScript languages.	
Syntax checking	Syntax checking.	
Declaration area	Declaration area.	
Scripts can be written on product root	Scripts can be written on product root.	

Table 4. Designing Scripts

Release 6.x	Release 7.x	Comment
Events return changes to categories	Events return changes only to items added from product table. Events do not return changes to relationship quantities.	Relationships are a grouping mechanism within a customizable product and are similar to categories.
Cfg_ItemInitialize	Use Cfg_InstInitialize.	Cfg_InstInitialize triggers once when session is started.
Cfg_ItemPreRequestSubmit	Not supported.	Can be simulated in some cases using the User Interface API.
Cfg_ItemChanged	Use this event only for child-customizable products. Use Cfg_ChildItemChanged for other components.	
Cfg_CategoryChanged	Not supported.	
Cfg_SessionPostProcess	Not supported.	
Cfg_ItemPreSynchronize	Use Cfg_InstPostSynchronize.	
Cfg_ItemPostSynchronize	Cfg_InstPostSynchronize.	
Cfg_SessionClosed	Implemented at Instance Broker level. Use Siebel API.	
GetSessionId	GetInstanceId.	Returns name of customizable product as a string. Does not return Cfg ID.
GetCfgId	Not supported. No longer meaningful.	
GetItemId	Not supported.	
GetItemQuantity	GetObjQuantity.	Returns the quantity of a component within the customizable product. Cannot be used for relationship quantities.
GetPropertyValue	Getting attribute value is supported through the Siebel API.	
GetItemState	Supported through Siebel API	
SubmitRequest	AddItem, RemoveItem	

Quote Integration and Configuration Assistant

Creating a model product and associating a model with it is not required in release 7.x. Instead, you create a customizable product work space, which is similar to creating a record in the Model Manager at 6.x. This work space associates the parts of the customizable product, including its components, links, resources, and UI definition with the product record. When you release a new version of the customizable product, it becomes available immediately for configuration in quotes and in eSales Web pages.

At 6.x, the Configuration Assistant view was used to display models, select items, and transfer items to a quote. If you wanted to modify the way Configuration Assistant looked, you had to use Siebel Tools to build a new view. In release 7.x, a Product UI Designer is provided within eConfigurator to create the browser pages, called selection pages, that will display during a configuration session. You can select from several base themes and product themes that define basic page layout. You can also select the controls, such as radio buttons or check boxes, to use for displaying items.

The base themes, product themes, and controls are stored in template files that you can customize or use to create your own templates. In addition, you can use the User Interface Property Designer to customize how individual items display. If you do not want to create a user interface for a customizable product, the system provides intelligent defaults for creating selection pages automatically. Table 5 maps 6.x features to 7.x for quote integration and Configuration Assistant.

Table 5. Quote Integration and Configuration Assistant

Release 6.x	Release 7.x	Comment
Verify a solution	Verify a customizable product configuration.	
Verify a quote	Verify a quote.	
Update a quote	Update a quote.	
Solution name	Customizable product name.	
Solution quantity	Customizable product quantity.	
Line item quantity	Component quantity in a customizable product.	

Table 5. Quote Integration and Configuration Assistant

Release 6.x	Release 7.x	Comment
Reconfigure a solution with a new version of a model	Reconfigure a stored session with a newer version of a customizable product	
Create and manage Favorites	Create and manage Favorites	
Add an item in a configuration session	Same. Define item display in Product UI Designer	You can also accept system defaults for item display
Remove an item	Same. Define item display in Product UI Designer	You can also accept system defaults for item display
Add a category	Add a relationship in Product Designer	
Unsatisfied category icon	A flag displays when a relationship is below its minimum cardinality or is required.	
Finish It!	Finish It!	
Item state explanation	The user interface contains an Explanation button.	Clicking the Explanation button displays an explanation.
Item properties displayed in applet	Properties have been replaced by configurable attributes. Attributes display in configuration session or in Quotes > Dynamic Attributes.	
Messages and Recommendations	Messages and Recommendations	
Unsatisfied requirement message	Unsatisfied requirement message	
Quantity out of range message	Quantity out of range message	
Conflict-exists pop-up message	Conflict-exists pop-up message	Explanation and ability to proceed or cancel user action is supported
Edit quantity of an item	Edit quantity of an item	Requires UI control that allows editing of quantity.

Select a Configuration Method 3

There are two approaches you can use to create customizable products. The first approach is a browser-based method and includes eAdvisor. The second approach is a server-based method and uses the eConfigurator rule engine.

Read this chapter carefully and determine which method you should use to create customizable products.

Choosing a Deployment Method for eConfigurator

Siebel offers two deployment options for creating eConfigurator applications:

- Server-based administered through Customizable Product.

  This method works by solving simultaneous constraints to ensure accuracy of the solution, with all data and constraint processing occurring at the server.

  How to use the server-based eConfigurator is covered in this guide.

- Browser-based administered through Interactive Designer

  This method uses Configuration tables to describe relationships, and delivers a subset of configuration capabilities directly to the end-user's browser, using only JavaScript and HTML.

  How to use Interactive Designer is covered in the *Siebel Interactive Designer Administration Guide*.

While either method may work for your configuration needs, the following sections provide guidelines for selecting one over another.

Consider the Nature of Your Data

To select between the two methods, consider whether your data is dense or sparse.

Dense Data

When your data set is “dense” and relatively small, consider deploying a browser-based application. This model makes it easy to apply business rules that manage the different combinations of data.

Table 6 is an example of dense data. In this example, a set of shirts is available in a variety of colors and, except for yellow, in all sizes. Yellow is available only in small. The combinations of data are easily entered in Configuration tables in Interactive Designer.

Table 6. Sample Browser-Based Data Set

Color	Size
Red	*
Green	*
Blue	*
Yellow	Small

At runtime, users can freely explore options for a given configuration, make selections, and determine what features are most important to them, not necessarily what is valid based on the first or top level selection. As exception messages are presented for invalid selections, it is relatively easy for users to get back to a valid state, because of the relatively small number of exceptions and relatively “dense” data.

Alternatively, within this deployment model, users navigating a large and sparse data set may spend more time than desired trying to identify a valid state. To avoid this situation, modelers can build in constraints to minimize the number of “invalid states.” However, you will need to weigh the trade-offs between the time spent building in constraints versus modeling in a constraint-based environment, particularly as product complexity increases. In this situation, consider using the Server-Based model explained in the next section.

Sparse Data

When your data set is large and "sparse," consider deploying a server-based application.

Table 7 shows a simple model where the data is more sparse and contains multiple, complex exceptions. This type of model is more efficiently expressed as a set of constraints.

Table 7. Sample Server-Based Data Set

	Feature X	Feature Y	Feature Z
Product A		Yes (*1)	*
Product B	Yes (*2)		*
Product C			*

*1 – only if x > y and not combined with A
*2 – only if A and B are combined

Consider Your Runtime Deployment Requirements

To select between the two methods, consider your runtime deployment requirements. In general, the more complex the configuration problem becomes, the fewer the number of concurrent users that are likely to access it. For example, if one were deploying a Web-based configuration application to end-consumers to configure something simple like an automobile, or a computer laptop, one would want to plan for peak concurrent user loads well into the thousands. At the other extreme, if one were deploying an application to allow business customers and field salespeople to configure specialized semiconductor manufacturing systems, one would expect a much lower number of possible simultaneous users.

Siebel eConfigurator's browser-based deployment mode uses JavaScript and HTML to execute configuration directly in the client's browser, enabling nearly limitless scalability for simple to moderately complex configuration with minimal server infrastructure.

Siebel eConfigurator's server-based deployment mode can support simple to extremely sophisticated configuration models. Scaling to larger user communities in this mode is done through the addition of infrastructure, such as more processors or additional servers as necessary.

Also consider the impact your product line will have on runtime deployment. A broad product line with individual product categories is a good fit for browser-based deployment. Other product categories would be more easily deployed using a server-based, constraint model.

When to Use Each Method

Table 8 provides a summary of the conditions to consider when choosing a method for deploying your eConfigurator application.

Table 8. Best Conditions for Use of Each Method

Browser-Based Deployment	Server-Based Deployment
Best for dense data	Best for sparse data
Requires minimal server infrastructure	Handles large number of exceptions well
Handles broad product line with individual product categories well	Supports extremely sophisticated configuration models

When to Use Each Method

Basic Product Administration 4

This chapter describes the basic product administration tasks common to both simple and customizable products.

Understanding the Product Record

Nonconfigurable products are called s*imple products*. Products with components that are interactively configurable at the time of purchase are called *customizable products*.

You enter a product into the Siebel database by creating a product record. This record stores important information about the product. The only required field in the product record is the product name. However, it is important to associate the record with a price list and a product line. This allows users to create quotes and to find important information about the product. In addition, when you associate a product with a product class, the product inherits the attributes defined on the class.

Table 9 lists the fields in the product record. Some of the fields are toggles. You activate or deactivate these fields by clicking on them. An X or check mark displays when the field has been selected. Except where noted, the default for toggles is blank, not selected.

Table 9. Fields in the Product Record

Field	Description
Allocate Below Safety	Click the box to allow allocation below the safe inventory level of this product.
Auto Allocate	Click the box if you are using automatic allocation by the Order Fulfillment engine of a particular product during the fulfillment process.

Table 9. Fields in the Product Record

Field	Description
Auto Substitute	Click the box to allow auto-substitution. Auto-substitution is the automatic use by the Order Fulfillment Engine of a substitute product when the product ordered cannot be found in inventory. The substitute products are set using the Create Substitute form on the Product Field Service Details page.
Bundle	A check mark or X displays if this is a bundle product. A bundle is a group of products sold together as one product. This field is read-only.
Class	The product class to which you want to assign this product. The product will inherit all the attributes defined on the class or that are inherited by the class.
Class Product	A check mark or X displays if this customizable product has been designated a class product. For more information on class products refer to the chapter on customizable product structure. Do not click in this field.
Compensable	Click the box if sales personnel can receive compensation for selling the product.
Customizable	A check mark or X displays if this is a customizable product with a work space and at least one version of the product has been released and is available to users. This field is read-only.
Description	Enter a brief description of the product.
Division Code (SAP)	Can be used for setting up user access to products but is not recommended. Instead, set up user access by assigning products to categories.
Effective End	The date after which the product is unavailable. After this date the product does not display in price lists and cannot be added to quotes.
Effective Start	Enter the date on which the product becomes available. The product does not display in price lists and cannot be added to quotes until this date.

Table 9. Fields in the Product Record

Field	Description
Global Product Identifier	Enter a unique product identification string. Use this field to map products from one Siebel installation to another or to a third-party product master. This field is useful when the string in the Part # field is required for local use or is not compatible with third-party product masters. This field is intended for use by integrators needing to move product information between applications.
Equivalent Product	Displays the primary equivalent product. Click in this field to display all equivalent products or to add additional equivalent products.
Field Replaceable	Click the box if this is a field-replaceable unit.
Format	The drop-down menu displays training formats such as Instructor led and Web-based.
Image File Name	Select the image file associated with the product. You can also select the image in Product Administration > Product Images.
Integration Id	Enter the back-office application product ID. This field can be used by SAP and Oracle Product Connectors.
Item Size	Enter the numeric product size.
Lead Time	Enter the standard lead time for ordering the product. Measured in weeks. For example, if you enter 2, this means 2 weeks.
Model Product	This field is obsolete. It is provided as a reference to upgrade users of eConfigurator.
MTBF	Enter the mean time between failure for the product.
MTTR	Enter the mean time to repair the product.
Orderable	Click the box if the product can be ordered. Determines whether a product can be listed as a quote line item on a quote. All components you add to a customizable product must be orderable.

Table 9. Fields in the Product Record

Field	Description
Organization	Can be used for setting up user access to products but is not recommended. Instead, set up user access by assigning products to categories.
Pageset	Enter the name of the Interactive Designer pageset to which the product belongs. For more information on Interactive Designer refer to the *Siebel Interactive Designer Administration Guide*.
Parent Product	Select the parent product. This field is for record keeping only and is not used for creating or managing customizable products that have components.
Part #	Enter the part number of the product.
Part Number Method	The drop-down menu displays the part number generation methods that can be assigned to a product. This menu is part of the smart part number feature.
Primary Vendor	Select the primary vendor for the product. The primary vendor must be specified to associate the product with an opportunity in the Opportunity Product Analysis Chart view.
Product	Enter the name of the product. This is the only required field. Products that will be added to the same user access category must not have the same name.
Product Level	Enter the numeric product level in the product hierarchy. This field is for record keeping only and is not used to create or manage the product class system.
Product Line	Select the desired product line for the product.
Project Resource	Click the box if the product is a service for a project. This determines if the product is going to be available in the rate list.
Qty	Enter the number of items in the unit of measure. For example, if the unit of measure is a case, Qty would be the number of items in the case, such as 24.
Return if Defective	Default: The box contains a check mark or X. This means defective products should be returned by the customer when a replacement part is shipped. Remove the check mark if customers should not return defective parts.

Table 9. Fields in the Product Record

Field	Description
Revision	Enter the version of the product as it goes through revisions.
Sales Product	Click the box if the product is a sales product. Specifies if the product can be sold. If this box is not selected, the product will not display in the product pick list.
Serialized	Click the box if movement of the product (a transaction) requires an asset number or its corresponding serial number. The default is no check mark or X (not serialized).
Service Product	Click the box if the product is a service. Only products designated as service products will display when you click the Service button on a quote. Special pricing rules apply to service products. For more information, see the *Pricing Administration Guide*.
Ship Carrier	Select the name of the shipping carrier for this product.
Shipping Via	Select the shipping mode: air ground, and so on.
Status	Select the status of the product: prototype, alpha, beta, and so on.
Targeted Country	Select the country where you want to sell this product.
Targeted Industry	Select the industry you want to target with this product.
Targeted M/F	Select the gender (male, female) of the buyers you want to target with this product.
Targeted Max Age	Enter the maximum age of buyers for this product.
Targeted Min Age	Enter the minimum age of buyers for this product.
Targeted Postal Code	Enter the postal code where you want to target sales of this product.
Tax Subcomponent flag	Put a check mark in this field to compute the tax on a bundle by adding up the tax on its components. Useful when the tax rate or computation method is not the same for all the components in a bundle. Put a check mark in this field to compute the tax on a customizable product by adding up the tax on its components. Useful when the tax rate or computation method is not the same for all the components in a customizable product.

Table 9. Fields in the Product Record

Field	Description
Thumbnail Image File Name	Select the thumbnail image file associated with the product. You can also select the thumbnail image in Product Administration > Product Images.
Tool	Click the box if this product is a tool, such as one used by field service engineers.
Track as Asset	Put a check mark in this field if, when the product is purchased, you want to track it as a customer asset. This allows you to create quotes and orders based on the asset.
Type	The drop-down menu displays product types: product, service, training.
Unit of Measure	Select the unit of measure by which the product is sold, for example, Each.
Vendor Part #	Enter the vendor's part number for this product.
Vendor Site	Displays the primary vendor's location. This field is filled automatically when you select a vendor.

Understanding Verify

When a user creates a quote, agreement, or order, they can verify that the products listed are valid and that pricing is available. When the user clicks the Verify button, the following things are checked:

- The price list specified in the quote exists in the price list pick list.
- The price list effective date starts before the quote effective date and/or ends after the quote expire date. This check is not performed if quote start and end dates are not specified.
- All the products exist in the product pick list.
- The product effective date starts before the quote start date or ends after the quote expire date. This check is not performed if quote start and end dates are not specified.
- The product's attributes and attribute values are current.
- The list price for each product is correct. (The list price of each product is computed and compared to the quote price.)
- The discount amount on the quote does not exceed the quote's total price.
- The customizable product configuration is complete. An incomplete configuration means that the user has not made required selections.

If Verify finds an error, the user is prompted to correct the problem.

Verify does not check whether customizable products are configured correctly or whether a new version has been released since a customizable product was last configured.

Creating a Product Record

You enter products into the Siebel system by creating product records. The product record contains the product name and important information about the product, such as its product line name or part number.

Once a product record is created, it cannot be deleted. To prevent a product record from being displayed in picklists and dialog boxes, edit the product record to deselect the Orderable, Sales Product, and Service Product check boxes. You can also control display of the product by setting Effective Start and Effective End dates.

After creating a product record be sure to do the following things:

a Associate the product with a price list.

b Set up user access to the product. You do this by adding the product to a category. Categories are how product visibility is controlled. You must add a product to a category to make it selectable in a quote and to make it visible in eSales Web pages.

To create a product record

1. Navigate to Product Administration.
2. Add a new record.
3. In More Info form, click the show more button.

   The long version of the Products form appears.
4. In the form, select the product type (product, service, training).
5. Fill in other desired fields in the product record and save the record.

Editing a Product Record

You can change the content of any of the fields in a product record. Changing the class to which a product is assigned can change the attributes it inherits.

Changing the class to which a product is assigned can change the attributes the product inherits. If the product's attributes change, you must revise all customizable products in which the product is component. Verify that no configuration rules or scripts refer to attributes the product no longer has.

Observe the following guidelines for editing product records.

- **Product**. If you change the name of the product, you must revise all customizable products in which it is a component. Also revise configuration rules, UI design, and scripts that refer to the product.
- **Class Product**. This field designates a customizable product as a template. All products in the same product class inherit the customizable product's structure. Putting a check mark in this field or removing the check mark can greatly affect all the products in the product class. Refer to the chapter on creating customizable products for more information. Class products are not orderable.

To edit a product record

1. Navigate to Product Administration.
2. Select the desired Product.
3. Click in the desired field in Products to edit the record, or edit the desired field in the More Info form.

   To see all the fields in the product definition click the show more button in More Info.
4. Save the record.

Copying a Product Record

When you copy a product record, all parts of the product definition are included in the copy.

If you copy a customizable product record, the copy includes all the relationships, links, resources, scripts, rules, and user interface.

Use the Copy feature to create product templates. For example, your product line has a two-tiered structure. The first tier contains a half-dozen products that have a similar basic structure. The second tier contains products based on the structure of the products in the first tier.

You could create the first tier by copying a template customizable product 6 times. You would then modify each of the copies to form the first tier. These then become the templates you would use to create the second tier.

To copy a product record

1. Navigate to Product Administration.
2. Select the product you want to copy.
3. Open the menu and choose Copy Record.

   A new record appears.
4. Enter a name for the copy in the Product Field.
5. Revise other fields, such as Part # as desired.
6. Save the record.

Deleting a Product Record

Once you have created a product record, it cannot be deleted. However, you can edit all the fields in the record, including the product name.

If you change the product name, you must revise all the customizable product rules and scripts in which it appears.

Exporting Product Records for Display

You can export product records in several formats for display.

For example, you can download files in comma-separated format for display in Microsoft Excel. The supported formats are as follows:

- Tab delimited file
- Comma separated file (csv format for use with spreadsheets like Excel)
- HTML file
- A file with delimiters you specify

You can request all the rows in the current query or only the highlighted rows. You can request all columns or only the currently visible columns. Currently visible columns are those you have selected for display in the Columns Displayed form.

When you export a customizable product or bundle product for display, only the root-level product record is exported. The structure of the customizable product or bundle is not exported.

NOTE: This procedure exports only product records for use in other display mediums such as spreadsheets. This procedure does not export the structure of a product or any other information contained in records related to the product record. To export product structures and other information in XML format for use by other applications, refer to "Exporting and Importing Products" on page 74.

To export product records for display

1. Navigate to Product Administration.
2. Highlight the products you want to export.
3. Verify that the columns displayed are those you want to export.

   To add or subtract columns, open the Products menu and choose Columns Displayed.
4. To export the product records for display, open the Products menu and click "Export..."

   Do not click Export Product. This will export the product information in XML format for use by other applications.

   The File Download dialog box appears.
5. Follow the instructions in the File Download dialog box to save the file.

Associating a Product with a Product Class

You associate a product with a product class by adding a class name to the product record.

To associate a product with a product class

1. Navigate to Product Administration.
2. Select the desired product.
3. Click in the Class field and open the dialog box.
4. Select the desired product class.

   The product class appears in the Class field.
5. Save the record.

Associating Products with Price Lists

You associate products with price lists in Product Administration > Price Lists. Associating a product with a price list adds a line item for that product to the selected price list.

For more information on price lists, see the *Pricing Administration Guide* or the *Applications Administration Guide*.

To associate a product with a price list

1. Navigate to Product Administration.
2. Select the desired product.
3. Open the More Info Show menu and choose Price Lists.

   Price lists associated with the product appear.
4. Add a new record.

   The Add Price Lists picklist appears.
5. Select the desired price list.

   The price list appears under Price List.
6. Enter the list price for the item.
7. Complete the remaining fields as needed.
8. Save the record.

Setting Up User Access

User access means whether or not the user can select a product for a quote or whether the user sees the catalog or category containing the product in an eSales Web page.

The catalog administrator creates product catalogs, which contain product categories. The catalog administrator sets up access controls by assigning access groups to the catalog and to the categories.

The product administrator sets up user access to products, by assigning products to catalogs and categories. You can assign a product to more than one category, and thus more than one catalog. If the user belongs to a category's access group, a catalog's access group, or both, then the user can see the category in eSales Web pages. The user can also add the category's products to quotes.

Until you assign a product to at least one category, the product does not display in the following places:

- Pick Product dialog box used to add products to a quote. This means the product cannot be added to a quote.
- eSales Web pages. This means users cannot purchase the product.
- Products > All Products
- Products > All Products Across Catalogs

When creating customizable products, it is important that users have access permission for the customizable product and all its components. You accomplish this by first assigning the customizable product and its components to the same product category or to categories that have the same access groups. Then you assign users who will configure the product to these access groups. If the users in the access groups differ across components, these users will not be able to configure the customizable product correctly.

The recommended method for assigning users to access groups is to assign the users to organizations and then assign the organizations to the access groups.

To set up user access

1. Navigate to Product Administration.
2. Select the desired product.
3. Open the More Info Show menu and choose Category.
4. Click New to add a new category record.

   A dialog box appears that lists all the currently defined categories.
5. Select a category from the dialog box.
6. Add additional categories as needed by creating new records.

Setting Start and End Dates for Display of a Product

You can set the start date and end date for display of a product. This controls the display of the product in price lists and therefore whether the product can be added to a quote or can be selected for purchase on a Web site.

To control display of a product, you set the Effective Start and/or Effective End dates. They govern display of a product as follows:

- If you specify an Effective Start date and no Effective End date, the product displays on the Effective Start date and will continue to display indefinitely.
- If you specify an Effective End date and no Effective Start date, the product displays immediately after you create it, and it stops display on the Effective End date.
- If you specify both an Effective Start Date and an Effective End date, the product display begins on the Effective Start date and stops on the Effective End date.

The Effective Start and Effective End date fields are fields in the product record. To view these fields, query for the desired record in Product Administration > Products and then select More Info. In the More Info display, click the show more button to view all the fields in the product definition.

Creating Product Line Names

You create product line names in Application Administration > Product Lines.

You add products to a product line by selecting a product line name in the product record. Navigate to Product Administration > Products to view product records.

To create a product line name

1. Navigate to Application Administration > Product Lines.
2. In Product Lines, add a new record.
3. Fill in the following fields.
   - **Product Line**. Required.
   - **Product Line Manager**. Allows you to associate product line managers and other key personnel with the product line.
   - **Products**. Allows you to associate products with the product line. Products can also be associated with product lines on the Products page.
   - **Description**. Optional.
4. Save the record.

Creating Product Features

Products frequently share common features, such as size or data transfer rate. You create a list of these product features in Application Administration > Product Features. The features you create are added to a features picklist.

You assign features to products in the Application Administration > Product Key Features by selecting a product and then choosing the desired features from the features picklist.

Product features and product attributes are similar concepts. They both describe characteristics of the product that are of interest to customers. A product feature describes important characteristics of a product, particularly those that differentiate the product. For example, you sell a type of office chair that has aluminum construction. Your competitors sell the same office chair with steel construction. Aluminum construction is an important feature of the office chair because it differentiates the chair from your competitors. It is also a static feature and is not configurable. All of your customers who purchase this office chair get aluminum construction.

An attribute is a characteristic of a product that is configurable when creating a quote or purchasing the product. For example, the office chair fabric comes in one of three colors. Color is an attribute of the office chair because the user can choose the color at the time of purchase.

To create product features

1. Navigate to Application Administration > Product Features.
2. In Product Features, add a new record.
3. Fill in the following fields.
   - **Feature**. The name of the product feature.
   - **Product Line**. Allows you to associate a product line with the product feature.
   - **Description**. A brief description of the feature.
4. Save the record.

Assigning Key Features to a Product

Product key features are those features that you have defined in Applications Administration > Product Features. The system automatically adds these features to a features picklist so that you can assign them to individual products.

To assign a key feature to a product

1. Navigate to Product Administration.
2. Select the product to which you want to assign a key feature.
3. Open the More Info Show menu and choose Product Key Features.

   The Product Key features list appears.
4. In Product Key Features, add a new record.
5. Enter the feature description, and save the record.

   The feature appears in Product Key Features.
6. Repeat this procedure to add additional key features.

Viewing Product Attributes

If a product has been assigned to a product class, it inherits all the attributes defined on the class. Attributes are configurable characteristics of a product. When the user purchases the product, they select the desired value for the attribute.

For example, you create a product class called Computer Chassis. For this class, define an attribute called construction and give it two values, aluminum and steel. All the products you assign to this class inherit this attribute. When the customer purchases these products, they can choose either aluminum or steel construction.

You define product classes and attributes in Application Administration, Class Administration. You assign products to classes by choosing a product class in the product record.

To view a product's attributes

1 Navigate to Product Administration.

2 Select the desired product.

3 Open the More Info Show menu and choose Dynamic Attributes.

The Dynamic Attributes list appears and lists all the product's attributes.

Defining Related Products

You can define several types of relationships between products. This causes the related products to appear together in other parts of the Siebel application.

For example, if you define a substitute product in Related Products, the substitute product displays in the Product Field Service Details view. If you define a substitute product in the in the Product Field Service Details view, it displays automatically as a substitute product in the Related Products view.

You can define the following types of relationships:

- Bundled
- Component
- Cross-Promoted
- Integrated
- Recommended Service
- Service
- Substitute

To define related products

1. Navigate to Product Administration.
2. Select the product with which you want to associate related products.
3. Open the More Info Show menu and choose Related Products.

   The Related Products list appears.
4. In Related Products, add a new record.

   The Add Internal Products dialog box appears.
5. Select the desired product.

   The product appears in Related Products.
6. To change the relationship of the related product, click in the Relation field and choose the desired relationship from the drop-down menu.
7. Save the record.

Designating Equivalent Products

For each product you define, you can designate one or more other products as equivalent products. You can then display these products and compare their product features. You can also assign a ranking to the equivalent products that reflects their degree of equivalence.

Equivalent products differ from substitute products in that they do not automatically display in the Field Service Details view.

You can designate one of the equivalent products as the primary equivalent product. The equivalent primary product is the one displayed in the Equivalent Products field in the product definition and other places where the display allows only one equivalent product to be shown.

To designate equivalent products

1. Navigate to Product Administration.
2. Select the desired product.
3. In the product form, click the show more button to expand the form.

   Expanding the form displays the Equivalent Product field.
4. Click the select button in the Equivalent Product field.

   A dialog box appears. It lists all the equivalent products you have defined for this product.
5. Click New to add an equivalent product.

   The dialog box displays a query form.
6. Query for the desired product and click OK.

   The product appears in the Equivalent Products list in the dialog box.
7. Click in the Primary field to select the desired primary equivalent product.
8. Click OK to exit the dialog box.

   The primary equivalent product appears in the Equivalent Product field in the product record.

Comparing Features of Equivalent Products

You compare equivalent products by displaying all the equivalent products for a product and then selecting which features you want to use for the comparison. You can then rank the equivalent products.

To compare features of equivalent products

1. Navigate to Product Administration.
2. Select the desired product.
3. Open the More Info Show menu and choose Product Comparison.

   The Product Comparison list appears. Equivalent products are displayed in the columns.
4. In Product Comparison, add a new record.

   A dialog appears that contains all the product feature definitions.
5. Select the desired product from the dialog box.

   The feature is added to the Product Comparison list.
6. Repeat the steps above until all the desired features have been added.
7. Assign a ranking to the equivalent products, if desired.

   A rank of 1 means a product has the highest degree of equivalence relative to the other equivalent products.

Creating a Product Auction

If you have Siebel's auction management product, you can create auctions for products you have created. The Create Auction button is located in many of the major tabs in Product Administration, for example the More Info form in the Products list.

For information on creating and managing auctions refer to the *Siebel eAuction Guide*.

Creating Product Entitlements

Entitlements refer to the services that come with a product. They are created on the Product Entitlements page under Product Administration.

When you create a product entitlement, you can designate the entitlement as applicable to "Agree Line Item Products" and/or "Entitlement Template Products." These are for Field Service use. For more information, see the *Siebel Field Service Guide*.

To create product entitlements

1. Navigate to Product Administration.
2. Select a product for which to create entitlements.
3. Open the More Info Show menu and choose Product Entitlements.

   The Product Entitlements list appears.
4. In Product Entitlements, add a new record.
5. Click the select button in the Name field and select an Entitlement template from the Entitlement Templates dialog box.

   The entitlement template record is added to the Product Entitlements list.
6. Click in the Agree Line Item or Entitlement Template Products field to set these features.

   A check mark appears to indicate these features are set.

Associating Literature with Products

You associate literature with products in Product Administration > Product Literature. Product literature can be such things as product bulletins, brochures, competitive analyses, and image files.

To associate literature with a product

1. Navigate to Product Administration.
2. Select a product with which to associate literature.
3. Open the More Info Show menu and choose Product Literature.

   Literature items for the product appear.
4. In Product Literature, add a New Record.

   The Add Literature dialog box appears.
5. Select the desired literature items.

   The literature displays in the Product Literature list.

Adding Product News

Product news is information about a product, typically FAQs and service bulletins. You associate news with products in Product Administration > Product News.

Product news is not the same as product literature. Literature is associated with products in the Product Literature view under Product Administration.

To add a news item to a product

1. Navigate to Product Administration.
2. Select a product to which you want to add a news item.
3. Open the More Info Show menu and choose Product News.

   News items for the product appear.
4. In Product News, add a new record.

   The Pick Product News dialog box appears. To read the first few lines of a news item in the dialog box, place your cursor over it.
5. Select the desired news item.

   The news item appears under Product News with its title under the Solution field and the solution type set to Product News.
6. Edit the record as needed by clicking in the desired field.

Associating Images with Products

You can associate both a thumbnail image and a regular image with a product. You can select image files for a product in either Product Administration > Products or in Product Administration > Product Images.

The following procedure describes how to associate images with a product in Product Administration > Product Images.

To associate images with a product

1. Navigate to Product Administration.
2. Open the More Info Show menu and choose Product Images.

   The Product Images form appears.
3. In the form, click the Image File Name field and select an image from the dialog box.
4. In the form, click the Thumbnail Image File Name field and select an image from the dialog box.

Creating Product Field Service Details

You provide information about how to replace a defective part with substitute parts in Product Administration > Product Field Service Details.

Most field service information is entered when creating products in the Products view, but Inventory Options and Substitute Products are managed in the Product Field Service Details view.

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

Creating Product Measurements

The Product Measurements page under Product Administration is used to define which measurements field service personnel should make and what the parameters of those measurements should be.

For more information, see *Siebel Field Service Guide*.

Exporting and Importing Products

You can export product definitions to other databases and import them from other databases. When you export a file, its definition is stored in an XML file.

Exports

When you export a simple or customizable product the following information is exported:

- Product name
- Vendor name
- Part number
- Class name
- Orderable (Yes or No)

No other fields in the product record are exported.

When you export a customizable product, the following parts are included in the export besides those above:

- For each component in the customizable product, the class system path to the component. This includes the class system path to the customizable product itself.

  The class system path is the list of product classes, starting at the root level, that are required to specify the exact location of a product in the class system. For example, a customizable product relationship contains all the products from the product class Hard Drives. The Hard Drives class has the following parent classes: Media Drives, and Computers. When you export the customizable product, all three product classes are exported.

  Other classes in this part of the product class system that do not contain components in the customizable product are not exported. For example, Computers contains two subclasses, Media Drives and Game Devices. In the example, there are no game devices in the customizable product. When you export the customizable product, the Media Drives class is included in the export, but the Game Devices class is not.

  On import, the exported product classes are added to the import database product class system, if they do not already exist.

- Attribute definitions for all exported product classes.
- Relationship definitions
- Rules
- Resources
- Links
- UI definition in the Product UI Designer
- UI property definitions in the User Interface Property Designer
- Scripts created using the Script Designer

The following things are not exported:

- Pricing models for the customizable product and its components
- Customized Web templates for UI property definitions
- Image files or HTML files referred to in UI property definitions

You can export a customizable product from either Product Administration > Products or from Product Administration > Customizable Products > Versions. If you export a customizable product from Versions, you can choose which released version to export. You can also export the work space. If you export a customizable product from Products, the most recently released version of the customizable product is exported.

NOTE: The Products menu contains both an Export... and an Export Product entry. To export a product, you must select Export Product. The Export... option is for exporting the contents of the Products list in spreadsheet or HTML format.

To export a product

1. Navigate to Product Administration.
2. Select the product you want to export.
3. Open the Products menu and choose Export Product (not Export...).

   The Product Export dialog box appears.
4. To export a simple product or an attribute-based customizable product, click Export Single Product.

   A File Download dialog box appears.

   If you click Export Single Product for a customizable product with a work space, only the product record is exported.

   You can also use this option to export a customizable product record but not the product's structure.
5. To export a customizable product that has a work space, click Export full Structure.

   A File Download dialog box appears.
6. Select Save this file to disk.

7 Browse to the location where you want to store the XML file, specify the file name, and then click Save.

   The system creates an XML file containing the product definition and structure and stores it at the location you specified.

8 Close the File Download dialog box.

Imports

The system uses the product name and its vendor name (if any) to uniquely identify it. An import will fail if the imported product specifies a vendor name that does not exist in the import database.

On an import, a product that has a vendor specified will overwrite an existing product in the import database under the following conditions:

- The product vendor exists in the import database.
- The product in the import database has the same name and same vendor.
- The product in the import database has the same name and no vendor.

On an import, a product that has no vendor specified will overwrite an existing product in the import database only if the existing product has the same name and no vendor specified.

You can import a customizable product into a database that does not contain either the component products or the related product class hierarchy. When you validate the customizable product, its component products and their attributes do not display. Also the attributes for the customizable product itself do not display.

To import a product

1. Navigate to Product Administration.
2. Open the Products menu and choose Import Product.

   The Product Import dialog box appears.
3. Click Browse, locate the XML file containing the product you want to import, and then click Open.

   The XML file displays in the Product Import dialog box.
4. In the Product Import dialog box, click Import.

   The product is imported into the database.
5. In Products, query for the imported product.

   Review the product record and verify it is accurate. Change the class name or other information as needed.
6. If you imported a customizable product, refresh its work space and then click Validate.

   Verify that the component products, resources, links, and attributes are correct. Also verify that the user interface is correct.
7. Set up pricing for the product as needed.

Obtaining a Product List Report

You can obtain a report that lists all the products in the product table. For each product, the report shows the following information:

- Product name
- Part number
- Description
- Unit of measure
- Vendor
- Product line
- Effective start date
- Effective end date

The product list displays in the Siebel Report Viewer. You can print the report or create an email attachment.

TIP: The on screen display of the report typically lists more products on each page than the Products list. Use the report to scan through the product table.

To obtain a product list report

1. Navigate to Product Administration.
2. In the application View menu, select Reports.
3. In the Reports dialog box, select Admin Product List.
4. Click Run.

   The Siebel Report Viewer appears and displays the Admin Product List report.
5. Print the report or create an email attachment as desired.

Product Classes 5

This chapter describes how to create product classes and class hierarchies. Classes provide a central location for defining product attributes. Products inherit the attributes of the classes to which they belong.

Understanding Classes

A product or service can be thought of as a collection of physical features and characteristics. Color, size, manufacturer, capacity, voltage, license type, expiration period, interest rate, and height are just a few of these. Those characteristics needed to describe your product meaningfully for your customers are called product attributes.

Classes provide a way to organize and administer product attributes. The key to understanding classes is inheritance. Attributes defined at the class level are automatically inherited by all the class members. When you assign a product to a class, it automatically inherits all the attributes defined for that class. Classes let you define what attributes are maintained for products, propagate those attributes to the products, and maintain those attributes in a consistent fashion.

When you define an attribute for a class, you specify both the attribute name and the range of values that the attribute can have. This range of values is called the attribute domain. For example, for a class called blanket, you define an attribute called color and define its domain to be green, red, and blue. Every blanket you assign to this class inherits the attribute color and its possible values.

Subclasses are classes that have a parent class. Subclasses have the following characteristics:

- Subclasses can be nested as deeply as needed. This forms the class hierarchy.
- Subclasses inherit the attributes of their parent class. As you nest downward, each subclass inherits the entire set of attributes from the classes above it.

- You can modify the definitions of inherited attributes. If you do so, this breaks inheritance from the parent class. Changes to attribute definitions in the parent class are not inherited by modified attributes in subclasses.

The class hierarchy is a mechanism for organizing and managing product attributes. It is separate from the mechanisms you use to organize products themselves, such as product lines and product categories.

For example, you have the class hierarchy in Figure 1. The product class called Class has two attributes defined on it, Attribute 1 and Attribute 2. Class also has a subclass called Subclass. Subclass has Attribute 3 defined on it and contains one product, called Product C.

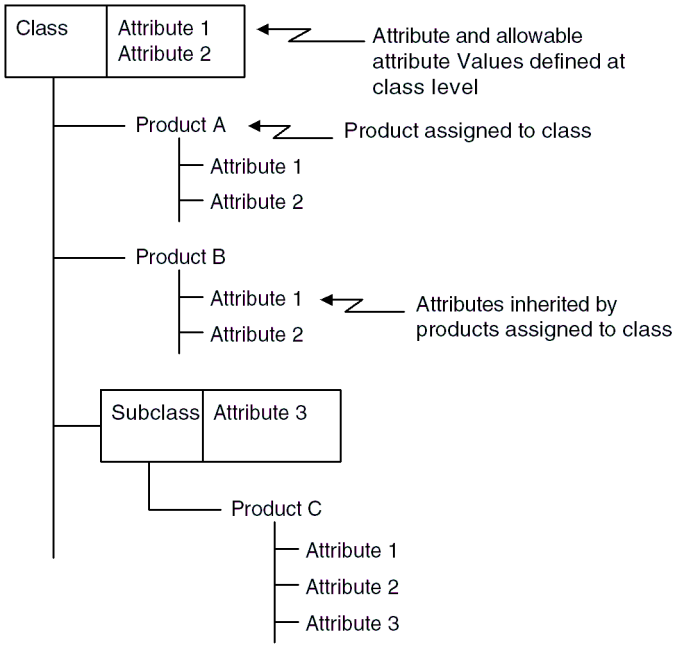

Figure 1. Class Hierarchy

Subclass inherits Attribute 1 and Attribute 2. It also has an attribute definition of its own, Attribute 3. Product C, assigned to Subclass, inherits all three attribute definitions.

When you define a customizable product, you define named parts called *relationships*. Then you add the contents of classes to them. Adding a small number of products to a relationship from a large product class requires that the entire class be searched each time the customizable product is instantiated. This can adversely affect performance. Consider defining the class system to avoid this.

In Application Administration > Class Administration, you can create classes, organize them into hierarchies, and define attributes for them.

A class record in Application Administration > Class Administration has the following fields:

- **Name**. This is the class name.
- **Display Name**. This is the name that is seen by the customer. If left blank, the name in the Name field is displayed to customers.
- **Parent Class Name**. If the class is a subclass, this field lists the parent class name.
- **Searchable**. Put a check mark in this field to make the class name available for parametric search.

Defining a Class

When you define a class, it is added to the list of all available classes. The name you choose must be unique. To create a subclass, specify a parent class in the definition.

To define a class

1. Navigate to Application Administration > Class Administration.
2. In Classes, add a new record.
3. Fill out the fields in the record and save the record.

   To create a subclass, select a parent class in the Parent Class field.

   The new class definition appears in the Classes list. It also appears in the Class Explorer view.

Creating a Class Hierarchy

A class hierarchy consists of classes and subclasses. A subclass is a class that has a parent class. In other words, subclasses are classes within classes. There is no limitation on how deeply you can nest subclasses.

You create and manage class hierarchies in Application Administration > Class Administration. You do this by specifying a parent class when defining a class.

You can view the hierarchy in Application Administration > Class Explorer. This view contains a tree display that shows the hierarchy in a manner very similar the Microsoft Windows file Explorer. You can expand or collapse classes and subclasses as needed to view the hierarchy. The portion of the hierarchy in which you are located displays in the Classes list.

When you run a query on the Classes list, it only searches the currently highlighted level. For example, if you are at the class level, the query searches only the classes at that level. If you are at the first subclass level, the query only searches the list of first-level subclasses belonging to the parent class. Search results are displayed in the Classes list.

To create a class hierarchy

1. Navigate to Application Administration > Class Administration.
2. In the Classes list, select the desired class.
3. Click in the Parent Class field, and select a class from the dialog box.
4. In the Show menu, choose Class Explorer.

   The Class Explorer appears and shows a tree display of product classes.
5. Locate and expand the parent class.

   Verify that the subclass displays correctly beneath the parent class.

Editing a Class Definition

Editing a class definition record does not change the attributes defined on the class. However, if you change the parent class name of a subclass to another already-existing class name, this changes the location of the subclass in the class hierarchy and can change which attributes the products in the subclass inherit.

For example, a subclass SC1 has parent class PC1, which has three attributes defined on it A1, A2, A3. This means SC1 inherits attributes A1, A2, A3. Class PC2 has attributes A4, A5, A6 defined on it. If you change the parent class of subclass SC1 from PC1 to PC2, this changes the attributes inherited by SC1 to A4, A5, A6. You have moved SC1 from being a subclass of PC1 to being a subclass of PC2.

If you are changing the parent-class name for class, do the following procedure first.

To prepare a product class for a parent-class name change

1. Run a query in the Products list to identify all the products assigned to the class.
2. Analyze how changing the parent class name of the class will affect the attributes inherited by these products.
3. Identify all pricing rules defined for the attributes inherited by the class.

   Note which rules must be changed to reflect the new parent class name and any new attributes.
4. Identify all configuration rules that refer to inherited attributes of the class.

   Note which rules must be changed to reflect the new parent class name and any new attributes.

If you are changing the class name, do the following procedure first.

To prepare a product class for name change

1. Identify all pricing rules defined for the attributes inherited by the class.

   Note which rules must be changed to reflect the new parent class name and any new attributes.
2. Identify all configuration rules that refer to inherited attributes of the class.

   Note which rules must be changed to reflect the new parent class name and any new attributes.

3 Identify all customizable product relationships of type Class and Dynamic Class that have been defined using the class.

   Note which relationships need to be redefined to reflect the new class name.

4 Identify any customizable product UI properties defined for the class.

   Note any UI property definitions that must be revised to reflect the new class name.

Before editing a class definition, make sure you have fully analyzed the impact on attribute inheritance.

Also make sure you have analyzed the impact on pricing rules, configuration rules, and UI design.

To edit a class definition

1 If you are changing the parent class name of a class, verify that all the steps in preparing the class for a parent-class name change are complete.

2 If you are changing a class name, verify that all the steps in preparing a class for name change are complete.

3 Navigate to Application Administration > Class Administration.

4 In Classes, select the desired record.

5 Click in the desired field to edit the record.

6 Save the record.

7 Modify pricing rules, configuration rules, relationship definitions, and UI property definitions as needed.

8 If you changed the parent class name of a class, log out and log in again in order to see the new class name in subclass records.

Deleting a Class

Deleting a class deletes attributes defined on the class. Deleting a class also deletes all subclasses of the class. Deleting a class does not delete the products assigned to the class or its subclasses.

For example, product A belongs to class B. There are six attributes defined on class B. This means product A has six attributes defined for it. If class B is deleted, product A no longer has attributes defined for it.

To prepare a product class for deletion

1. Run a query in the Products list to identify all the products assigned to the class.
2. Delete the class from these product records.

   If there are attributes defined on the class, analyze the effect of removing these attributes from the products.
3. Verify that no pricing rules are defined for the class or attributes defined on the class.
4. Verify that no configuration rules refer to the class or to attributes defined on the class.
5. Verify that no customizable product relationships of type Class or Dynamic Class have been defined using the class.
6. Review the UI design for all customizable products containing the class.
7. Redefine groups as needed to remove the class from groups.

Before deleting a class definition, make sure you have fully analyzed the impact on attribute inheritance.

Also make sure you have analyzed the impact on pricing rules, configuration rules, and UI design.

To delete a class definition

1. Verify that all steps in preparing a product class for deletion are complete.
2. Navigate to Application Administration > Class Administration.

   The Class Administration view appears.
3. Select the desired the class.
4. Open the Classes menu and choose Delete Record.
5. Click OK when asked to confirm you want to delete the record.

Exporting or Importing Classes

You can export a class or the whole class structure to another database. When you export a class, the following parts are included in the export:

- The parent class of the class you are exporting plus all the subclasses of the parent class. When you export a class, the export contains not just the class you selected, but the portion of the class structure to which it belongs.
- Attribute definitions for the classes and all subclasses.
- List of values definitions associated with attribute definitions. List of values are exported in the current language only.

The products in the classes are not exported.

When you export the whole class structure, all classes and subclasses are exported, along with the items listed above. Products are not exported.

When you export a class or the class structure, an XML file is created in a location you specify. The XML file contains the exported class structure. When you import this class structure, the system reads the XML file and synchronizes the class system of the import database to the XML file. The XML file takes precedence, and the class system is modified to reflect the portion of the class system in the XML file.

For example, in the XML file the subclass shoes, has the parent class footwear. In the import database the subclass shoes has the parent class Wardrobe. After importing the XML file, the subclass shoes will have the parent class footwear.

Use the following process to update the class structure in database B with changes from database A.

a Back up database B.

b Export the desired classes from database A.

c Import the classes to database B.

d Compare the updated class structure and list of values definitions in database B with database A.

e Verify that components in affected customizable products in database B have the correct attributes.

Use the following process to update both the products and class structure in database B with changes from database A:

a Use the process above to update the class structure in database B.

b Export the products from database A, except customizable products.

c Import the products into database B. Verify that the products are in the correct classes and inherit the correct attributes.

d Export customizable products from database A.

e Import customizable products to database B. For each customizable product, verify that the component products are present and have the correct attributes.

To export a class or the whole class structure

1 Review the processes above.

2 Navigate to Application Administration > Class Administration.

The Class Administration view appears.

3 Select the class you want to export.

4 Open the Classes menu and choose Export Class.

The Class Export dialog box appears.

5 To export the class click Export in the Class Export dialog box. To export the whole class system, click Export All.

A Save As dialog box appears. If the database is remote, a download dialog box also appears.

6 Browse to the location where you want to store the file, specify the file name, and then click Save.

The system creates an XML file containing the exported class structure and stores it at the location you specified.

7 Close the Class Export dialog box.

When you import a class structure, you must import the entire contents of the export file. You cannot choose which classes in the file to import.

To import a class structure

1. Review the processes above.
2. Navigate to Application Administration > Class Administration.
3. Open the Classes menu and choose Import Class.

   The Class Import dialog box appears.
4. Click Browse, locate the XML file containing the class structure you want to import, and then click Open.
5. In the Class Import dialog box, click Import.

   The new class structure is imported into the database.
6. Open the Show menu and choose Class Explorer.
7. In the Class Explorer tree display, expand classes as needed to verify that the imported classes are correctly placed.

Locating a Class

The Class Explorer view includes a tree display that you can expand or collapse to display the hierarchy of product classes. Use the Class Explorer to locate classes and to verify the class hierarchy after you have edited it.

To locate a class

1. Navigate to Application Administration > Class Explorer.

   The Class Explorer view appears. Classes display in a tree and in the Classes list.

2. Click a class name in the tree to display its subclasses.

3. To query for a class, click Query in the Classes list.

   Results appear in both the tree and in the Classes list.

Product Attributes 6

You can define attributes for a wide variety of items, such as products, literature, and so on. This chapter focuses on doing so with products and explains how to define attributes, provide attributes to products, and set attribute values for products.

Before defining attributes, you must create a hierarchy of product classes and subclasses.

Understanding Product Attributes

Product attributes, also called dynamic attributes, are customer-facing, configurable characteristics of a product or its components. For example, you sell a product in three colors. As part of creating this product, you would define an attribute called Color and assign it the three colors. As part of purchasing the product, customers would choose one of the colors.

Components of a product are not attributes. For example, you sell a desktop computer. Customers can select one of several types of CD-ROMs when configuring this product. Having a CD-ROM is a characteristic of this product, but the CD-ROMs are components, not attributes.

Product attributes and product features are similar concepts. They both describe characteristics of the product that are of interest to customers. However, feature definitions do not create configurability. For example, you could define a feature: “Comes in three colors, red, green, and blue.” This feature definition can be displayed to the user as a message only. It does not create the mechanism for choosing the color. To create that, you must define a product attribute and assign it the values red, green, and blue.

A product attribute has two parts: the name of the attribute and the value of the attribute. For example, you could define an attribute with the name Color and the values red, green, or blue. The allowable values for an attribute are called the attribute domain. In a configuration session, the user can select only one value for an attribute.

You can define attributes directly in the administration interface. You do not need to create database table extensions or new field definitions in Siebel Tools.

Attributes are implemented in a way that allows users to select the desired attribute value when they configure the product. For example, when a user creates a quote, the Color attribute displays in the interface, and the user can select the desired value.

Classes are the way you organize and administer product attributes. The key to understanding classes is inheritance. Attributes defined at the class level are automatically inherited by all the class members. When you assign a product to a class, it automatically inherits all the attributes defined for that class. Classes let you define what attributes are maintained for products, propagate those attributes to the products, and maintain those attributes in a consistent fashion.

You can define attributes at the class or subclass level. You cannot define an attribute at the product level. At the product level, users can only select the attribute's value.

Attribute Domains

When you define an attribute, you must define the domain of allowable values for the attribute. There are two methods for defining the domain:

- **List of values**. A list of values domain is a list of the specific values the attribute can have. When the user configures a product, they select one of the values from a drop-down menu. For example, the attribute Color could have the list of values red, green, or blue.

  A special case of a list of values domain is a list of values that contains only one value. This is useful for creating attributes that you use for managing resources. For example, you could create an attribute called slots-consumed for a class of computer expansion cards. Typically, each card requires one expansion slot. You would create a list of values containing only the number 1, and would set 1 as the default value. You could then write rules that subtract the value of this attribute from a resource called slots-available each time the user picks an expansion card.

  Parametric search can be used to search for attribute values.

  Attribute-based pricing can only use attribute values that have been defined as elements in a list of values (LOV). Attribute-based pricing requires the discrete values that appear in an LOV.

- **Range of values**. A range of values is defined by upper and lower limits. Rather than selecting a value, the user enters a value within the range. For example, the attribute Length could have the range 1 inch to 60 inches. When the user configures the product, they would enter a value between 1 and 60 in the field provided.

  Parametric search cannot be used to search for attribute values.

  Attributes that have a range of values domain cannot be used for attribute-based pricing.

- **Business Component (Buscomp) field**.This domain is defined by a field in a business component. For example, you can define an attribute called Account and associate it with the Name field in the Account business component. When the user configures a product, they see an attribute called Account. They can then open a picklist and select the desired account. This domain type can be used only for products that are configured in eConfigurator selection pages.

  Parametric search cannot be used to search for attribute values.

  Attributes that have a buscomp field domain cannot be used for attribute-based pricing.

Domain Data Types

The data type you specify in the attribute definition determines how the system interprets the values in the domain. For example, you define an attribute with a list of values domain. You define the attribute values to be 1, 5, 10. To write configuration rules that perform numeric computations using these values, you must select the data type Integer or Number when defining the attribute.

The domain of an attribute can be one of the following data types:

- **Boolean**. Use this data type when the user's input is true or false, yes or no. If you specify the Integer data type for these inputs, the system assigns 1 for True or Yes inputs. False and No are assigned 0.
- **Number**. The attribute value can be any positive or negative real number. In Boolean expressions, numbers greater than 0 are interpreted as true. Omit commas when specifying the domain. For example, enter 10,000 as 10000.
- **Integer**. The attribute value can be any positive or negative whole number. If a computation results in a fractional amount, the result is rounded to the nearest whole number. In Boolean expressions, integers greater than 0 are interpreted as true. Omit commas when specifying the domain. For example, enter 10,000 as 10000.
- **String**. The attribute value can be letters, numbers, or any combination. Attributes with this data type cannot be used as operands in a computation or as the result of a computation. The only arithmetic operator that can be used with this data type is = (equals). For example, you can write rules that test if the user has picked a specific string from a list of values.

- **Date**. The attribute value is interpreted as a date and must be in the correct date format. The system administrator sets date format defaults. Arithmetic computations using dates is not supported. For example, you cannot increase or decrease a date using a computation. All comparison operations are supported for dates. For example, you can compare two dates and determine whether one is earlier than (<), later than (>), or the same as (=) another date. Data type mismatches cause the user's input to be rejected, or can cause indeterminate results. For example, comparing a date data type to an integer data type.
- **Time**. The attribute value is interpreted as a time and must be in the correct time format. The system administrator sets the time format defaults. The time data type has the same restrictions as the Date data type. Data type mismatches cause the user's input to be rejected, or can cause indeterminate results. For example, comparing a time data type to an integer data type.
- **DateTime**. The attribute value is interpreted as both a date and time and must be in the correct format. The system administrator sets the format for this data type. Arithmetic computations using this data type are not supported. For example, you cannot increase or decrease a DateTime value by using a computation. The only comparison operation that is supported is = (equals). Data type mismatches cause the user's input to be rejected, or can cause indeterminate results. For example, comparing a DateTime data type to an integer data type.

Attribute Definition Fields

An attribute definition includes the following fields:

- **Name**. The attribute's name. Use the attribute name to search for the attribute.
- **Data Type**. The types are Boolean, Date, Integer, Number, and Text. The data type refers to how the system will interpret the attribute value.
- **LOV Type**. This field specifies the name of the list of values for attributes with a list of values domain.
- **Default Value**. This field specifies the default value that the customer sees. If you write rules that manipulate the attribute value, the eConfigurator engine can override the default value.

- **Validation**. This field specifies the range of acceptable attribute values. Specify the range using Siebel query-by-example syntax. For example, a component can be purchased in sizes ranging from 1 to 20 inches inclusive. You would specify this range by entering >=1 AND <=20.

  One of the most important uses of the Validation field is to specify the range of acceptable inputs for attributes that have a range of values domain. However, you can also specify a validation expression for list of values domains. For example, you define a list of values that are of type number. This list of numbers may contain numbers that you do not want the user to select. You can create an expression, that limits the choices to a subset of those specified in the list of values.

- **Required**. Enter a Y (yes) or N (no) in this field. When you enter a Y, the field displays a check mark when the record is not highlighted. A Y means the user must choose the attribute value.

- **Display Name**. The attribute name that the user sees. If not specified, the user sees the name specified in the Name field.

- **Unit of Measure**. Select a unit of measure from the drop-down menu. This selection is displayed to the user.

- **Description**. Make an entry in this field to describe the attribute. Users do not see this description.

- **Hidden**. Prevents the attribute from displaying in Quote, Agreement, Order, or Asset views. Attribute still displays in customizable product selection pages.

- **Searchable**. A check mark in this field means this attribute and its values can be used in parametric searches. For example, if the attribute is Color, you can search for products that have Color = Red.

- **Unit of Measure**. Allows you to select the unit of measure, such as day, month, dollar, dozen.

- **Analytics Sequence #**. Assigning a number to this field makes the attribute visible for use by Siebel Analytics. The sequence number does not control the order of display of the attribute in selection pages. If this field is not displayed, open the list menu and select Columns Displayed to add the field to the display. Assign a positive whole number. Attributes defined on a class must have unique sequence numbers. Assigning a sequence number is highly recommended.

Understanding Lists of Values (LOV)

When you define an attribute with a list of values (LOV) domain, you must either select an existing list of values or create a new one. A list of values has two parts:

- List of values name, called an LOV type. The LOV type identifies the list of values. The user does not see the LOV type.
- The attribute values in the list. The user selects one of the these attribute values or accepts the default value when configuring a customizable product.

To create a list of values, you first create the LOV name. Then you define the attribute values in the list. The LOV name and attribute value records have the following fields:

- **Type**. (LOV name only) The LOV name. This is the name you entered when you created the LOV.
- **Display Value**. (attribute values only) The attribute value. This value displays to the user as one of the choices in the list of values.
- **Order**. The order in which the values are displayed in the drop-down menu the user sees. Assign 1 to the record you want to display first in the menu, 2 to the second record, and so on. Leave blank for LOV name definitions.
- **Active**. Removing the check mark from this field, removes the record from the list of values. Use this option to temporarily change the number of items in a list of values.

The following fields are provided to manage multilingual translation:

- **Multilingual**. Put a check mark in this field to translate the attribute value to the language specified in Language Name.
- **Language Independent Code**. For LOV Name, enter the LOV name. For attribute values, enter the attribute value. This name is used to match translations to the item.
- **Language Name**. The language in which the LOV name or attribute value displays.
- **Translate**. Put a check mark in this field to translate the LOV name or attribute value.
- **Replication level**. Specify whether the translation is intended for use at all levels in the translation hierarchy or only at the regional level.

Understanding Hidden Attributes

When you place a check mark in the Hidden field in an attribute definition, the attribute does not display in the Quote, Order, Agreement, or Asset views. For example, if you assign a product to a class that has hidden attribute A1. When you add this product to a quote and select Dynamic Attributes, A1 does not display.

The attribute continues to display in customizable product selection pages and you can write configuration rules on it.

Use hidden attributes to create configuration parameters that customers do not need to see. For example, you could define a hidden attribute whose value is the number of bays required for a chassis. You could then write configuration rules that use the value of this attribute to monitor the number of available bays during a configuration session.

Upgrade users. Use hidden attributes as a replacement for virtual products.

Defining an Attribute with a List of Values Domain

When you define an attribute that has a list of values domain, you must define a list of values. You do this by defining a list of values name, for example Color. Then you create the list of attribute values for Color, for example red, green blue.

In the user interface, this type of attribute displays as an attribute name accompanied by a drop-down menu. The user accepts the displayed default or opens the menu to make a choice.

Attribute-based pricing requires the list of values domain type. You cannot do attribute-based pricing with the range of values domain type.

The LOV Type and Validation fields determine the attribute domain:

- **LOV Type**. Select the list of values definition that you want to use for this attribute. You can also define a new list of values.
- **Validation**. Leave this field blank or enter an expression that restricts the user's choices. Use Siebel query-by-example syntax.
- **Default Value**. Enter the default you want to use from the list of values. This is the item that displays in the attribute field when the list of values menu is closed. If the attribute is required and the user does not change the default attribute, this is the attribute value the user receives. If left blank, no value displays in the attribute field when the menu is closed. The user must open the menu to make a selection. A default value is required if you are using attribute-based pricing.

To define an attribute with a list of values domain

1. Navigate to Application Administration > Class Administration.
2. Select the desired class.
3. In the Dynamic Attributes list, create a new record.

   The Dynamic Attributes form appears.
4. Fill out the Dynamic Attributes form and save it.

Defining an Attribute with a Range of Values Domain

You define a range of values domain attribute by entering an expression in the Validation field. This expression is used to validate the user's input. If the input is within the range (the expression returns a true), the input is accepted. If the input is outside the range (the expression returns a false), the input is rejected and the user receives an error message.

You cannot do attribute-based pricing with the range of values domain type. Attribute-based pricing requires the list of values domain type.

The LOV Type and Validation fields determine the attribute domain:

- **LOV Type**. Leave this field blank.
- **Validation**. Enter an expression that defines the range in Siebel query-by-example syntax. For example, to specify the real numbers between 1 and 100 inclusive, you would enter > =1 AND < = 100.
- **Default Value**. Enter a value from the range in this field if you want to display a default. This attribute value is assigned to every product that inherits the attribute.

To define an attribute with a range of values domain

1. Navigate to Application Administration > Class Administration.
2. Select the desired class.
3. In the Dynamic Attributes list, create a new record.

   The Dynamic Attributes form appears.
4. Fill out the Dynamic Attributes form and save it.

Editing an Attribute Definition

You can edit attribute definitions for both classes and subclasses. When you edit an attribute defined on a class, the attribute definition is changed for all members of the class. This means the attribute definition is changed for all subclasses and all products of the class.

For a subclass, if you edit an inherited attribute, this permanently breaks the chain of inheritance for the fields you edit. Changes to these fields in the parent class attribute definition no longer propagate to the edited attribute. By editing inherited attribute definitions, you can customize the way attribute definitions propagate through the product hierarchy.

For example, you have the class hierarchy in Figure 2. Product Class A has one subclass called Subclass B. Subclass B has one subclass called Subclass C. Class A has Attribute A defined on it. Subclass B has attribute B defined on it. Subclass C has Attribute C defined on it. Subclass B inherits Attribute A from Class A. Subclass C inherits Attribute A from Class A and Attribute B from Subclass B.

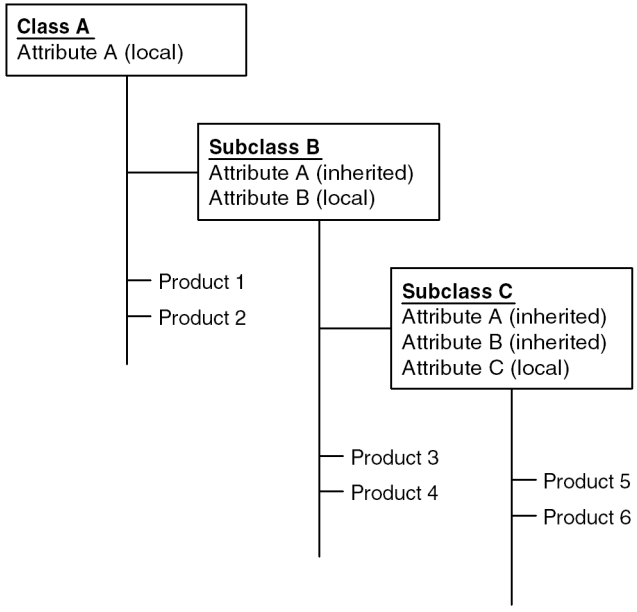

Figure 2. Product Class Hierarchy

In Subclass B, you edit the definition of Attribute A by entering a new Default Value. The Default Value field for Attribute A in Subclass B no longer inherits changes from Attribute A in Class A, its parent attribute.

When you edit a local or inherited attribute, the changes propagate to all members of the class or subclass. In the example, the new Default Value propagates to Attribute A in Subclass C.

There are restrictions on which fields you can edit in an inherited attribute definition. These restrictions are shown in Table 10.

Table 10. Editable Fields in a Subclass Inherited Attribute Definition

Field	Editable?
Attribute Name	Yes. Breaks inheritance for all fields. Same as defining new attribute.
Data Type	Yes. Breaks inheritance for all fields. Same as defining new attribute.
List of Values	Yes. Breaks inheritance for this field.
Default Value	Yes. Breaks inheritance for this field.
Validation	Yes. Breaks inheritance for this field.
Required	Yes. Breaks inheritance for this field.
Display Name	Yes. Breaks inheritance for this field.
Parametric Search	Yes. Breaks inheritance for this field.
Unit of Measure	Yes. Breaks inheritance for this field.
Description	Yes. Breaks inheritance for this field.

To edit an attribute definition

1. Navigate to Application Administration > Class Administration.
2. Select the desired class.
3. In the Dynamic Attributes list, highlight the desired attribute.
4. Click in the desired field to change its value.
5. Save the record.

Deleting an Attribute Definition

You can only delete an attribute definition from the class on which it is defined. When you delete an attribute defined on a class or subclass, the attribute is deleted from its members as follows:

- The attribute is deleted from all products in the class or subclass.
- The attribute is deleted from all subclasses where the attribute definition has not been edited.
- The attribute is not deleted from subclasses where the attribute definition has been edited.

Before deleting an attribute definition, verify that the attribute is not used in any configuration rules, or for attribute-based pricing.

To delete an attribute definition

1. Navigate to Application Administration > Class Administration.
2. Select the desired class.
3. In the Dynamic Attributes list, click the menu button and choose Delete Record.

   The attribute definition is removed from the Dynamic Attributes list.

Customizing an Inherited Attribute Domain

When you define an attribute on a class or subclass, it is inherited by all member subclasses. If you edit an attribute on the class where it was originally defined, the changes propagate to all member subclasses. The attribute definition is uniform for all subclasses that inherit it.

Subclasses can have two kinds of attributes: local and inherited. A local attribute is one that is defined on the subclass. An inherited attribute is one that is inherited from a parent class.

You customize an inherited attribute domain by editing its definition at the subclass level. When you edit an inherited attribute definition, the changes propagate to all members of the subclass, including other subclasses.

Editing an inherited attribute permanently breaks attribute inheritance for the fields you edit. Editing the domain of an inherited attribute permanently prevents an attribute from inheriting domain changes from its parent attribute.

If you delete the parent class attribute, it is not deleted from subclasses where inheritance is broken. (The attribute definition *is* deleted from all subclasses where inheritance has not been broken.)

For example, you have the class hierarchy in Figure 3. Product Class A has one subclass called Subclass B. Subclass B has one subclass called Subclass C. Class A has Attribute A defined on it. Subclass B has attribute B defined on it. Subclass C has Attribute C defined on it. Subclass B inherits Attribute A from Class A. Subclass C inherits Attribute A from Class A and Attribute B from Subclass B.

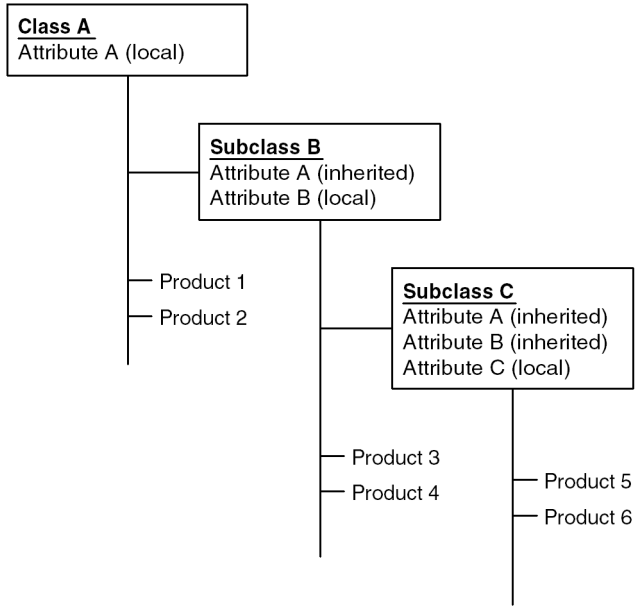

Figure 3. Attribute Inheritance

In Subclass B, you edit the domain of Attribute A by entering a new LOV Type and Default Value. The LOV Type and Default Value for Attribute A in Subclass B no longer inherits changes to these fields from Attribute A in Class A, its parent attribute.

When you edit a local or inherited attribute, the changes propagate to all members of the class or subclass. In the example, the new LOV Type and Default Value propagate to Attribute A in Subclass C.

You can edit the domain of an inherited attribute as follows:

- **List of values domain.** Select a different list of values. You can also edit the list of values definition and reapply it. If you want to add or remove items from the list of values, the recommended method is to define a new list of values and apply it.
- **Range of values domain.** Change the expression in the Validation field. Also change the instructions in the Description field.

You can also edit the other fields in the attribute definition, including the Default Value, Validation, Required, Display Name, Parametric Search, and Unit of Measure fields.

Associating Attributes with a Product

To associate attributes with a product, you assign the product to a class or subclass. A product inherits all the attributes of the class or subclass to which it is assigned. You cannot assign attributes directly to a product.

For example, a subclass has six attributes. Three of these are defined on the subclass, and three are inherited from a parent class. When you assign a product to this subclass, the product inherits all six attributes.

To associate attributes with a product

1. Navigate to Product Administration.
2. Select the desired Product.
3. Click in the Class field to display the select button. Then click the select button to open the Pick Class dialog box.
4. In the Pick Class dialog box, select the desired class.

   The class name is added to the product record.
5. Save the product record.
6. In the More Info Show menu, choose Dynamic Attributes.

   The Dynamic Attributes list appears. This list displays all the product's attributes.

Viewing a Product's Attributes

A product's attributes are inherited from the class to which it belongs.

When viewing attributes, be careful not to save the attribute records in the Dynamic Attributes list. This sets the attribute value so that it cannot be changed by the user or by configuration rules.

To view a product's attributes

1. Navigate to Product Administration.
2. Select the desired product.
3. Open the More Info Show menu and choose Dynamic Attributes.

   The Dynamic Attributes list appears. This list displays all the product's attributes inherited from its class or subclass.

Changing the Hidden or Required Settings for a Product

When you define an attribute at the class level, you can set the attribute to be hidden or required. Hidden attributes do not display in the Quote, Order, Agreement, or Asset views. Required attributes are those where the user must select a value for the attribute. The value of the attribute cannot be blank.

Attribute definitions propagate automatically to all the products that belong to the product class. However, you can change the Hidden flag and the Required flag settings for an attribute at the product level. This lets you manage the hidden or required settings for attributes product by product.

You can use the hidden setting to simplify your product class system. For example, if a product class has 8 attributes and a product has 7 of these attributes, you can put the product in this class and hide the eighth attribute. You do not have to create a special subclass with 7 attributes for the product.

To change the hidden or required settings for a product

1. Navigate to Product Administration.
2. Select the desired product.
3. Open the More Info Show menu and choose Dynamic Attributes.

   The Dynamic Attributes list appears. This list displays all the product's attributes inherited from its class or subclass.
4. Select the desired attribute and click in either the Required or Hidden field.

   This adds a check mark to the field, or removes the check mark if one is present.

Setting an Attribute Value for a Product

When you set the value of an attribute for a product, it cannot be changed by either the user, a configuration rule, or the eConfigurator engine. One example, is when you want to set an attribute value so that provide and consume rules can use it to add or subtract from a defined resource.

For example, you create an attribute called Slots Required for a product class containing expansion cards. Some cards take up one expansion slot; some take up two. You could define a list of values containing the integers 1 and 2 and make it the domain for Slots Required. For each expansion card you would then set the value of this attribute at 1 or 2. Users cannot change this value when configuring the product, and configuration rules cannot change this value.

You would then write a provide rule that increases a Slots Available resource when the user picks a chassis. For the expansion card class, you would write a consume rule that reduces Slots Available by the value of Slots Required, each time the user picks an expansion card. In this fashion, you use attribute values as constants that interact with a defined resource to manage a consumable configuration variable.

To set an attribute value for a product

1. Navigate to Product Administration.
2. Select the desired product.
3. Open the More Info Show menu and choose Dynamic Attributes.

   The Dynamic Attributes list appears. This list displays all the product's attributes inherited from its class or subclass.
4. Select the desired attribute and enter the desired attribute value in the Value field or select the desired value from the drop-down menu (list of values domain).
5. Verify that the value you have set is within the defined domain of the attribute and is the correct data type.

   If you enter a value that is outside the domain the eConfigurator engine will accept it unless the attribute definition includes a validation expression. If you enter a value that is the wrong data type (for example, a date when the data type is integer) the resulting engine behavior is indeterminate.

6 Click in the Read Only column.

A check mark appears, indicating the attribute value is set and cannot be changed.

7 Save the record.

This sets the value of the attribute. A check mark appears in the Read Only field. The value cannot be changed by the user during a configuration session or by the eConfigurator engine.

Creating a List of Values (LOV)

When you define an attribute that has a list of values domain, you must specify a list of values name. This LOV contains the attribute values.

Creating a list of values has two steps:

a Creating a list of values name.

b Defining the attribute values in the list of values.

NOTE: The steps for creating a list of values for a dynamic attribute are different from creating LOVs for other uses. Do not define LOVs for dynamic attributes in the Application Administration > List of Values view. This view does not display LOVs defined for use with dynamic attributes. When you define an LOV for a dynamic attribute, you do not have to stop and restart the server for the LOV to take effect.

Creating a List of Values Name

You create an LOV name by creating a new record in the dialog box where you select an LOV when defining an attribute.

After creating a list of values name, you define the attribute values that make up the list.

To create a list of values name

1. Navigate to Application Administration > Class Administration.
2. In the Classes list, query for the desired class.
3. In the Dynamic Attributes list, select the attribute for which you want to define an LOV.
4. Click in the LOV Type field to display the select button. Click the select button to display the dialog box for selecting LOVs.

5 In the dialog box, click New and fill out the form for defining an LOV type. Close the dialog box.

6 Click Save in the Dynamic Attributes form.

This transfers the new LOV name to the attribute record.

Defining the Attribute Values in a List of Values

After you define the LOV name, you create the values in the list, by creating a record for each attribute value.

To define the attribute values in the list

1 Navigate to Application Administration > Class Administration.

2 In the Classes list, query for the desired class.

3 In the Dynamic Attributes list, select the attribute for which you want to define LOV values.

4 Click the LOV name hyperlink in the LOV Type field.

A view appears that displays the LOV name and provides a list in which to add one record for each attribute value in the LOV.

5 For each attribute value in the LOV, create a new record in LOV list.

Editing a List of Values Definition

You can edit all the fields in a list of values definition. These changes propagate to all the locations where the list of values is assigned. When editing a list of values record, keep in mind the following effects:

- If you edit the Type (name), this changes the list of values to which the record belongs. In the user interface, this means the item moves from one list of values menu to another.
- If you edit any fields besides Type, the changes affect only the list of values to which the record belongs. For example, the list of values Color has three records. The display names are Red, Green, Blue. You change the display value for the first record from Red to Purple. For each attribute to which the list of values is assigned, the list of values is now Purple, Green, Blue.

When modifying a list of values record, observe the following guidelines:

- **Display Value**. Edit this field to change the name of a menu item in the list of values.
- **Language Name**. Edit this field to change the language in which the item displays. The language name for all records in a list of values should be the same.
- **Order**. Edit this field to change the order in which the values are displayed in the drop-down menu the user sees. Assign 1 to the record you want to display first in the menu, 2 to the second record, and so on.
- **Active**. Removing the check mark from this field, removes the record from the list of values. Use this option to temporarily change the number of items in a list of values.
- **Translate**. Put a check mark in this field in order to translate the menu item to the language specified in Language Name.

Several other fields are also included in the record used to define a list of values record. These fields are not meaningful for product management or pricing management.

NOTE: Do not edit LOVs of type CFG_RULE_TYPE. This will disrupt the function of the Rule Designer.

To edit a list of values definition

1. Navigate to Application Administration > Class Administration.
2. In the Classes list, query for the desired class.
3. In the Dynamic Attributes list, select the attribute for which you want to edit LOV values.
4. Click the LOV name hyperlink in the LOV Type field.

   A view appears that contains the product class LOV types that were defined in this view. This view does not display LOV types that were defined in Application Administration > List of Values view.
5. In List of Values-Type, query for the desired list of values name.

   The values defined for the list of values name display in the area below List of Values.
6. Edit the list of values records as desired and click OK.

   You must log out and log back in to see the changes you have made in attributes to which the LOV is assigned.

Deleting a List of Values

You cannot delete a list of values name (Type) or its records. However, you can edit all the fields in a list of values record, including changing its name. This has the same effect as deleting the record.

Attributes with Business Component Domains 7

This chapter describes how to define attributes that have a business component domain. These attributes allow users to select a record from a pick applet, also called a dialog box. A field in this record then displays in the selection page as the value of the attribute.

This chapter requires that you be familiar with creating pick applets in Siebel Tools.

Understanding Attributes with a Business Component Domain

The Product Administrator has created a customizable product called Premier Service Package. The Product Administrator wants users to be able to select an account name when configuring the product.

This product has been assigned to a product class that has the attribute Account defined on it. Account has been added to a group in the Product UI Designer and will display in a selection page.

In the selection page, the Account attribute displays with a blank text field and a select icon. When the user clicks on the select icon, a dialog box displays containing available accounts. When the user selects an account, the account name is transferred to the Account text box.

In this scenario, the user was able to access a Siebel business component to display account records. When the user selected an account record, the account name field in the record was transferred to the selection page and became the value of the Account attribute. The domain of the Account attribute is the records retrieved by the business component and displayed in the dialog box, also called a pick applet.

Attributes with a business component domain differ from attributes with other domain types in several ways:

- Attributes with a business component domain should only be used with customizable products that are configured using selection pages. This means that only customizable products or products that will always be components of customizable products should be assigned to a class where this type of attribute is defined. If you assign a simple product to the class, attributes with a business component domain display in quotes, orders and so on with a text box but no select icon.
- The attribute values are not defined by a range or a list of values. The user selects the attribute's value directly from a pick applet, which displays information from a business component.
- Attributes can be defined so that when the user selects the value for one attribute (the primary attribute), the values for other attributes are automatically selected. For example, if the user selects a value for Account Name (the primary attribute), the value of the Address attribute is filled in automatically. Configuration rules can be written only on the primary attribute. You cannot write configuration rules on attributes whose values are automatically selected based on the value of the primary attribute.

You associate a business component with an attribute using the same process as creating a pick applet in Siebel Tools, with the following modifications:

a **Configuring the originating applet**. The selection page takes the place of the originating applet. You replace this procedure with steps that define the attribute and insert it in the selection page. To do this step, see "Adding the Attribute to a Selection Page" on page 126.

b **Configuring the pick applet**. You can use an existing pick applet or define a new applet. If you define a new pick applet, there is no change to the procedures described in *Siebel Tools Reference*.

c **Configuring the originating business component**. These procedures are replaced by defining a series of UI properties on the attribute. These UI properties specify the pick applet name, pick business component name, and pick map definitions. You can define multiple pick maps that display the content of several fields from the same record. You can also define UI properties to constrain the pick list.

d **Configuring the pick business component**. You can use an existing pick business component or define a new one. If you define a new one, there is no change to the procedures described in *Siebel Tools Reference*.

Assuming that the pick business component and pick applet are already defined in Siebel Tools, associating an attribute with a business component has the following steps:

a Add the attribute to a selection page. See "Adding the Attribute to a Selection Page" on page 126.

b Associate the attribute with a business component by defining UI properties on it. For information on these UI properties see "Understanding the UI Properties" on page 124. For information on associating an attribute with a business component, see "Associating the Attribute with a Business Component" on page 127.

You can set up attributes so that selecting a value for one automatically selects the values for others. To do this see "Setting Up Multiple Fields for Display" on page 129.

You can constrain a pick applet so that it displays only the records having a specified field value. See "Creating a Business Component Field Constraint" on page 132.

Understanding the UI Properties

You can define an attribute that has values the user can select from a pick applet. The pick applet displays records from a specified business component. When the user selects a record, a specified field in the record displays in the selection page as the value of the attribute.

Several predefined UI properties are provided to associate the attribute with a pick applet and pick business component. Defining these UI properties on the attribute replaces the Siebel Tools procedures for configuring the originating business component when defining a pick applet.

You define these UI properties on the attribute that you have set up to display a select button in the selection pages. These UI properties associate the attribute with a pick applet and a pick business component and are shown in Table 11.

Table 11. Predefined UI Properties

Name	Value
PickApplet	The name of the pick applet.
PickList	The name of the pick business component.
PickMap01	This is an XML tag that associates the attribute name with the pick business component field that you want to display.
PickMap*nn*	PickMap02 and so on, display multiple fields from the same pick business component. You can also use this PickMap definition to define a constraint on the records the user sees in the pick applet.

The PickMap value is an XML tag that has the following format:

```
<PickMap Field="AttributeName" PickField="BusCompFieldname"
Constrain="Y/N" BusObj="BusObjName" />
```

Field. For UI properties with name PickMap01, this specifies the name of the attribute in the selection page. Use the attribute name, not its display name.

For UI properties with Constrain = "Y", specifies the business component field name or attribute name to be used as a filter for the records displayed in the pick applet. The format for specifying a business component is `buscompname.fieldname.` The format for specifying an attribute name is `attributename.`

For example, to constrain the records displayed in the pick applet to those having the current record's Opportunity name in the Quotes view, (Quote business component), you would enter `Quote.Opportunity`. To constrain records to the value of the Account Name attribute, you would enter `Account Name.`

PickField. Specifies the pick business component field name. When the user selects a record from the pick applet, the contents of this field becomes the attribute value shown in the selection page. Use the business component field name, not the field's display name.

Constrain. When set to "Y" (Yes), PickMap specifies a business component field that filters the records the user sees in the pick applet. If not specified, the default is "N" (No).

BusObj. Specifies the business object in which the PickMap definition is active. If omitted, the PickMap definition is active in all business objects. For example, if you set BusObj = "Order Entry", the PickMap applies to orders but not quotes. Use this argument to constrain the pick applet differently for orders than for quotes. If you insert a BusObj argument in PickMap01, this limits the display of the select icon and pick applet to the specified business object, for example Quote.

Adding the Attribute to a Selection Page

This step creates the attribute in the class system and defines where it displays in the selection pages. It replaces configuring the originating applet step in the Siebel Tools process for creating a pick applet.

The attribute data type should be the same as the data type of the business component field from which the attribute value will come. For example, if the data type of the business component field is Boolean, the attribute data type should be Boolean. The system does not verify that the attribute data type and the business component field data type are the same.

In the Product UI Designer, you do not need to pick a UI control for the attribute. When you define the PickMap01 UI property on the attribute, the system automatically assigns a text field with select button to the attribute. When you select a record from the pick applet, the specified field in the record displays in the attribute text box.

If you select a UI control for the attribute, it will be overridden by the text box with select button.

To add the attribute to a selection page

1. Define an attribute on a class. Leave the LOV, Validation, and Default Value fields blank.
2. Verify that only products that are themselves customizable products or will always be components of customizable products are assigned to the class. Products that will be part of bundles should not be assigned to the class since they are not configured using selection pages.
3. Navigate to Product Administration. Then select and lock the desired customizable product. This product must belong to the class on which the attribute is defined.
4. In the Product UI Designer, select a group and add the attribute to the Group Item List.

   Do not select a UI control for the attribute.

Associating the Attribute with a Business Component

This step defines the UI properties needed to associate the attribute to a pick business component. It replaces configuring the originating business component step in the Siebel Tools process for creating a pick applet.

The Product Administrator has created a customizable product called Premier Service Package. This product has been assigned to a product class that has the attributes Account, Location, and Opportunity defined on it. These attributes have been added to a group in the Product UI Designer and will display in selection pages.

The Product Administrator wants users to be able to select an account name when configuring the product. To do this, the Product Administrator must define the following three UI properties on the Account attribute, as shown in Figure 4.

- **PickList**. Its value is PickList Account.
- **PickApplet**. Its value is Account Pick Applet.
- **PickMap01**. This UI property provides the name of the attribute and the business component field. Its value is an XML tag that has the following elements:
  - Field = “Account”. This is the attribute name.
  - PickField = “Name”. This is the business component field.

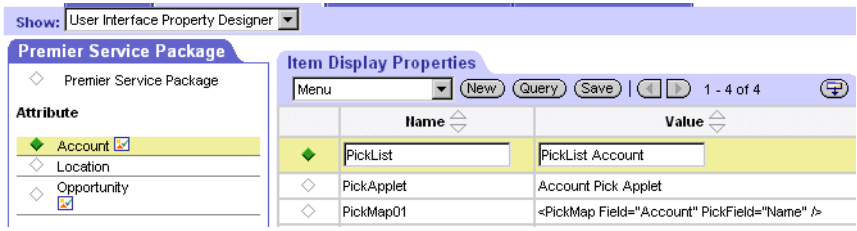

Figure 4. UI Properties for the Account Attribute

The Account attribute displays with a text box in the configuration selection pages. When the user clicks the select button, the Account Pick Applet displays. When the user selects an account and clicks OK, the Account name is transferred to the Account field in the selection page.

Table 12 shows how to use the predefined UI properties to associate an attribute with a business component.

Table 12. UI Properties

Name	Value
PickApplet	The name of the pick applet.
PickList	The name of the pick business component.
PickMap01	This is an XML tag that associates the attribute name with the pick business component field that you want to display. Only the Field and PickField variables are required. Enclose their values in quotes. Field: The attribute name in the selection page. PickField: The pick business component field to be used as the attribute value. The PickMap that provides this information must be named PickMap01

Associating the attribute with a business component

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. Navigate to Customizable Product > User Interface Property Designer.
4. Select the desired attribute.
5. Define the UI properties as shown in Table 12.

Setting Up Multiple Fields for Display

You can group attributes together so that selecting a record from a pick applet for one attribute populates several attributes. You do this by defining additional PickMap UI properties on an attribute. These additional UI properties define how to populate the other attributes.

The attribute on which you define PickMap01 is called the primary attribute. The user selects a value for this attribute and this causes the values for the other attributes to be selected automatically. This means the other attributes are read-only. The only way the user can change the value of these attributes is to open the pick applet for the primary attribute and choose another record.

The Product Administrator has created a customizable product called Premier Service Package. This product has been assigned to a product class that has the attributes Account, Location, and Opportunity defined on it. These attributes have been added to a group in the Product UI Designer and will display in selection pages.

The Product Administrator wants users to be able to select an account name when configuring the product. When they do, the Account Administrator wants to automatically populate the Location attribute with the state in which the account is located.

To do this, the Product Administrator must define four UI properties on the Account attribute as, shown in Figure 5 on page 130.

- **PickList**. Its value is PickList Account.
- **PickApplet**. Its value is Account Pick Applet.
- **PickMap01**. This UI property provides the name of the attribute and the business component field. Its value is an XML tag that has the following elements:
  - Field = “Account”. This is the attribute name.
  - PickField = “Name”. This is the business component field.

- **PickMap02**. This UI property defines an attribute that will receive its value automatically when the user selects a value for the primary attribute. In this case, the attribute is Location. The value of the UI property is an XML tag that has the following elements:
  - Field = "Location". This is the attribute name.
  - PickField = "State". This is the business component field.

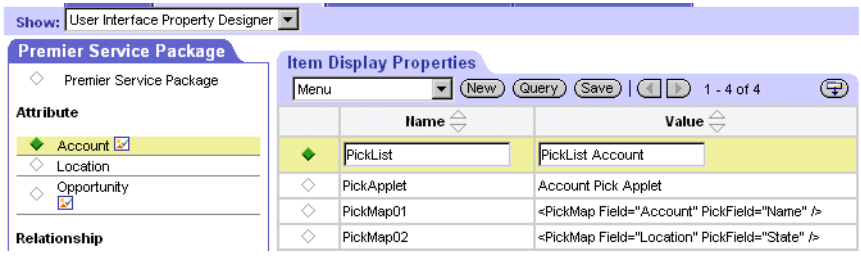

Figure 5. UI Properties for the Account and Location Attributes

The Account and Location attributes display with a text box next to them in the configuration selection pages. When the user clicks the select button and chooses an Account name, it is transferred to the Account field and the state name is transferred to the Location field.

Table 13 shows how to use the predefined UI properties to set up multiple fields for display.

Table 13. UI Properties

Name	Value
PickApplet	The name of the pick applet.
PickList	The name of the pick business component.
PickMap01	This is an XML tag that associates the attribute name with the pick business component field that you want to display. PickMap01 must be defined on the primary attribute Only the Field and PickField variables are required. Enclose their values in quotes. Field: The name of primary attribute. PickField: The name of the business component field.
PickMap02	Only the Field and PickField variables are required. These variables are for attributes other than the primary attribute. These attributes will be populated automatically when the user selects a pick applet record for the primary attribute. Attribute values for these attributes are read-only. Enclose the attribute values in quotes. Field: The name of the attribute, other than the primary attribute. PickField: The name of the business component field. You can define PickMaps to populate as many fields as desired. Number the PickMaps in sequential order, for example Pickmap03, PickMap04, etc.

To set up multiple fields for display

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. Navigate to Customizable Product > User Interface Property Designer.
4. Select the desired attribute.
5. Define the UI properties as shown in Table 13.

Creating a Business Component Field Constraint

You can constrain the records that display in the pick applet based on a field that they have in common with the business component that starts the configuration session. This is similar to using Siebel Tools to constrain the display of records in a pick applet based on a field in the originating business component.

You do this by defining an additional PickMap UI property on an attribute. This additional UI property defines how to constrain the records in the pick applet specified in PickMap01. The constraint PickMap specifies the business component name and field to use to filter the records in the pick applet.

The Product Administrator has created a customizable product called Premier Service Package. This product has been assigned to a product class that has the attributes Account, Location, and Opportunity defined on it. These attributes have been added to a group in the Product UI Designer and will display in selection pages.

The Product Administrator wants users to be able to select an account name when configuring the product in a quote. When they do, the Account Administrator wants to automatically populate the Location attribute with the state in which the account is located.

In addition, the Product Administrator wants to constrain the pick applet to display only the accounts associated with the opportunity name displayed in the Quote Opportunity field. For example, if the opportunity name is Boeing, the pick applet would display all the Boeing accounts only.

To do this, the Product Administrator must define five UI properties on the Account attribute, as shown in Figure 6 on page 133.

- **PickList**. Its value is PickList Account.
- **PickApplet**. Its value is Account Pick Applet.
- **PickMap01**. This UI property provides the name of the attribute and the business component field. Its value is an XML tag that has the following elements:
  - Field = “Account”. This is the attribute name.
  - PickField = “Name”. This is the business component field.

- **PickMap02**. This UI property defines an attribute that will receive its value automatically when the user selects a value for the primary attribute. In this case, the attribute is Location. The value of the UI property is an XML tag that has the following elements:
  - Field = "Location". This is the attribute name.
  - PickField = "State". This is the business component field.
- **PickMap 03**. This UI property filters the display of records in the pick applet to those having the same value as a field in the business component that starts the configuration session. The value of the UI property is an XML tag that has the following elements:
  - Constrain = "Y". This notifies the system that the UI property defines a constraint.
  - Field = "Quote.Opportunity". This is the business component name and field name that will be used as a filter.
  - PickField = "Name". This is the pick business component field name that will be filtered.

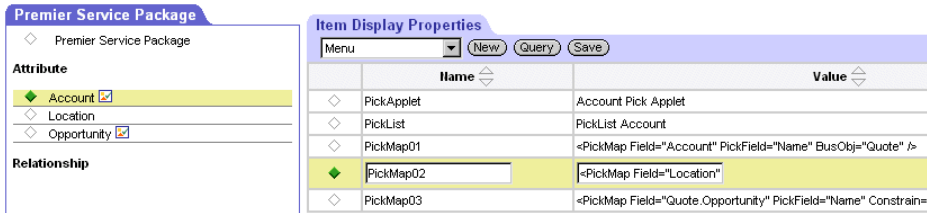

Figure 6. UI Properties for Account, Location, and Filtering

The Account and Location attributes display with a text box next to them in the configuration selection pages. When the user clicks the select button for Account, a pick applet displays. It contains only the accounts that have the name specified in the Opportunity field of the quote that started the configuration session (Quote business component). When the user selects an account and clicks OK, the Account name is transferred to the Account field and the state name is transferred to the Location field.

Table 14 shows how to use the predefined UI properties to constrain the user's choices.

Table 14. UI Properties

Name	Value
PickApplet	The name of the pick applet.
PickList	The name of the pick business component.
PickMap01	This is an XML tag that associates the attribute name with the pick business component field that you want to display. PickMap01 must be defined on the primary attribute Only the Field and PickField variables are required. Enclose their values in quotes. Field: The name of primary attribute. PickField: The name of the business component field.

Table 14. UI Properties

Name	Value
PickMap02	Only the Field and PickField variables are required. These variables are for attributes other than the primary attribute. These attributes will be populated automatically when the user selects a pick applet record for the primary attribute. Attribute values for these attributes are read-only. Enclose their values in quotes. Field: The name of the attribute, other than the primary attribute. PickField: The name of the business component field. You can define PickMaps to populate as many fields as desired. Number the PickMaps in sequential order, for example Pickmap03, PickMap04, and so on. Define one PickMap for each field.
PickMap03	This PickMap defines the business component field used to filter the records displayed in the pick applet Field, PickField, and Constrain variables are required. Enclose their values in quotes. Field: Specifies the business component name and field that filters the pick applet. The format for specifying the field name is `buscompname.fieldname.` The business component specified in Field must be in the same Tools business object (BusObj) as eConfigurator. The field cannot be the field specified in PickMap01. For example, to constrain the records displayed in the pick applet to those having the current record's Opportunity name in the Quotes view, (Quote business component), you would enter `Quote.Opportunity`. **PickField**: Specifies the name of field in the pick business component that is filtered by the Field variable. Constrain: Must be set to "Y". All PickMaps must be have a unique number. For example if there are 4 PickMaps, PickMap01…PickMap04, name the constraint PickMap, PickMap05.

To use a field to constrain the user's choices

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. Navigate to Customizable Product > User Interface Property Designer.
4. Select the desired attribute.
5. Define the UI properties as shown in Table 14 on page 134.

Creating an Attribute Value Constraint

You can constrain the records that display in the pick applet based on the value of an attribute in the selection pages. The attribute value is used to create a search specification that matches the attribute value to the value in a field in the business component that populates the pick applet.

For example, you define an attribute called Account Name. In a configuration session, the user selects Hewlett Packard from the pick applet you have defined for this attribute. You have also created an attribute called Address. You have defined a pickmap for this attribute that constrains the display of addresses to those belonging to the value of the Account Name attribute, which is Hewlett Packard. The pick applet for Address would display only those addresses for Hewlett Packard.

You create an attribute value constraint in the same way as creating a business component constraint. The only difference is that for Field you specify the attribute name rather than a `buscomp.fieldname` in the pickmap definition. Using the example above, you would enter `Field="Account Name"`.

NOTE: If the attribute you specify to constrain the pick applet does not have an attribute value, the pick applet will contain no records.

Smart Part Numbers 8

This chapter explains how to set up part numbers so that they are dynamically generated based on the product attributes that the user selects. Smart part numbers can be used to generate part numbers when creating quotes, orders, and agreements. The can also be used when adding items to a shopping cart.

Understanding Smart Part Numbers

Smart part numbers allow you to automatically generate part numbers for different combinations of product attributes. You do not have to make an entry in the product table and provide a part number for each combination of product attributes that a customer can purchase.

For example, you sell shirts in three sizes: small, medium, and large. You also sell them in three colors: red, green, and blue. There are nine possible combinations of size and color that customers can purchase. Each combination needs a part number that can be passed to a back-end system at the time of purchase.

One way to set this up is to make an entry in the product table for each combination. In other words, you create nine separate products. This is time consuming and does not take advantage of the attribute features in the class system. It also does not take advantage of the attribute-based pricing features in Siebel ePricer.

Another way to set this up is to make one entry in the product table for the shirt. You then define color and size attributes on the class to which the shirt belongs. Finally, you use smart part number to define which part number to assign to each combination of attributes. For example, when the customer selects the shirt in size small and color blue, smart part number generates a part number for this combination and displays it in the quote. You can also use attribute-based pricing in ePricer to determine the price of the shirt.

There are several advantages to this method:

- It makes managing the product table easier. You make one entry for a product and then use the class system to define and manage its attributes. If you enter a product's attribute combinations as products in the product table, you must manually edit the table when attributes change.
- It makes managing part numbers easier and more accurate. You can make one entry for a part number definition and it will be applied to all the forms of the product consistently and accurately.
- It allows you to take advantage of important features in related products such as attribute-based pricing in Siebel ePricer.

Smart part number provides the following methods for defining how part numbers are generated:

Dynamic. You specify what product attributes participate in creating a part number and the string that each attribute value will have. You then define a part number template with placeholders for the attribute values. Smart part number inserts the value of the attribute into the part number template to create the final part number. Use this method when your part numbers include important information, besides attribute values, that is needed to uniquely identify the product.

Predefined. You specify what product attributes participate in creating a part number. You can then do one of two things:

- You can auto-generate a matrix of all the combinations of these attributes. Random part numbers are provided for each combination. You can accept the random values or replace them with your own values.
- You can manually create the matrix, inserting your own part numbers.

When the user selects product attributes, smart part number searches the list for the correct attribute combination and uses its part number. Use this method when your part numbers cannot be easily created using string substitution.

You define named smart part number methods on product classes. These methods use the attributes defined on the class to generate a part number. Only attributes with a list of values (LOV) domain can be used to generate part numbers. When you assign a product to a class, you can select for it any of the smart part number methods that have been defined on the class.

If you add or remove attributes on a class, or change the values for an attribute, these changes are not automatically propagated to the smart part number methods defined on the class. You must manually update each smart part number method with the changes.

Smart part numbers can be used to generate part numbers for products in quotes, orders, agreements, and for products added to shopping carts.

Creating Dynamically Generated Part Numbers

When you create dynamic part numbers, you first create a template that contains a placeholder for each attribute you want to include in the part number. You then define mappings that specify how attribute values replace the placeholders. When the user chooses attribute values, the system inserts the mappings into the part number template to generate the part number.

The following definitions are important to understanding dynamically generated part numbers.

- **Part number template**. A sequence of sections in a specified order.
- **Section**. A portion of a part number template. Each section contains one attribute name that acts as a placeholder. Sections can also contain a prefix and a postfix.

  For example, you want to create part numbers that begin with ENU- and end with -MC. You want to include values for two attributes, Attrib1 and Attrib2, and separate them with a dash (-). Here is an example: ENU-S-GRN-MC. In this part number, S is the value substituted for Attrib1 and GRN is the value for Attrib2.

  To create a part number template, you would define two sections. In Table 15, the first row is the first section of the part number. The second row is the second section.

Table 15. Part Number Template

Prefix	Attribute Name (Placeholder)	Postfix	Sequence
ENU-	Attrib1	-	1
	Attrib2	-MC	2

- **Mapping**. A mapping is a string of characters you define for an attribute value. The mapping is what the abbreviation method uses to determine what characters to insert in the part number. If you do not define a mapping, the abbreviation method uses the attribute value itself.
- **Abbreviation method**. The abbreviation method determines how the characters are derived from the mapping. These characters replace the attribute's placeholder in the part number template. The mapping methods are: Abbreviation, Acronym, First Two Symbols, First and Last Symbols, First Symbol.
  - **Abbreviation**. Inserts the whole mapping.
  - **Acronym**. Inserts the first character in the mapping plus the first character following each space in the mapping.
  - **First Two Symbols**. Inserts the first two characters in the mapping.
  - **First and Last Symbols**. Inserts the first and last characters in the mapping.
  - **First Symbol**. Inserts the first character in the mapping.

An example of how abbreviation methods determine which characters to insert in part number templates is shown in Table 16. The first column shows the mapping. The remaining columns show the characters that would be inserted in the template for each abbreviation method.

Table 16. How Mapping Methods Work

Mapping	Abbreviation	Acronym	First Symbol	First and Last Symbols
Small	Small	S	S	Sl
X Large	X Large	XL	X	Xe
X-LARGE	X-Large	X	X	XE
A123 B456 C78	A123 B456 C78	ABC	A	A8

Only attributes with a list of values (LOV) domain can be used to create dynamic part numbers.

Before creating part numbers using this method, determine which attributes you want to use in the part number. Then write down the sections of the part number, including the prefix an postfix for each attribute.

Creating dynamically generated part numbers is a process that has the following tasks:

a Creating a part number generation record. This record names the part number generation method and specifies whether the method is dynamic or predefined.

b Defining the template

c Defining a mapping for each attribute value

d Testing the part number template

These tasks are described below.

To create a part number generation record

1 Navigate to Application Administration > Class Administration and query for the desired product class.

2 Click Part Number Definitions.

3 In Part Number Definitions, click New to create a new record.

4 In the Name field, enter a name for the part number generation method.

This name should be unique and should indicate whether the method is dynamic or predefined.

5 In the Type field, open the menu and choose Dynamic. Then step off the record.

The next step in the process is to define the template.

To define the part number template

1 In the part number definition Name field, click the name you entered.

The name is a hyperlink. The Part Number Method view appears.

2 In Part Number Template, create a new record.

Each record is one section in the part number.

3 In the Attribute Name field, open the drop-down menu and choose the attribute for this section.

   Only attributes with a list of values (LOV) domain display in this menu.

4 Enter the prefix and postfix for this section as needed.

5 In the Abbreviation Method field, open the drop-down menu and choose an abbreviation method for the attribute mapping.

6 Enter a sequence in the sequence field.

   The sequence determines the order of the section in the part number.

7 Repeat these steps for each section you want to include in the part number.

The last step is to define mappings for the attribute values.

To define a mapping for each attribute value

1 In Part Number Template, highlight the template section for which you want to define attribute value mappings.

   The values for the attribute display in Attribute Mapping.

2 For each attribute value in Attribute Mapping, enter a mapping in the Mapping field.

   The abbreviation method uses the string in the Mapping field to determine what characters to insert in the part number for this attribute value. If you do not enter a mapping, the abbreviation method uses the attribute value as the mapping.

3 Repeat these steps for each section in Part Number Template.

The last step is to test the part number template. You do this by creating a quote and selecting the product for which you have created a smart part number method. Add the product to the quote enough times so that you can select all the combinations of attributes needed to verify that the smart part number template is working correctly. The smart part number displays in the Line Item Detail view. For more information on locating the smart part number in a quote, refer to "Viewing a Product's Smart Part Number in a Quote" on page 152

Editing a Dynamic Generation Method

You can edit a dynamic generation method in several ways:

- Edit the name of the generation method
- Delete the generation method
- Edit the part number template
- Edit the attribute value mappings

If you edit the name of a generation method or delete the method, the change is reflected in all product records to which the method is assigned. For example, you delete the generation method Dynamic1. All product records that have Dynamic1 as the Part Number Method, no longer have an assigned generation method.

If you edit the product template or attribute mappings, the changes become effective immediately. The next time the product is added to quote, order, and so on, the revised part number scheme will be used. The part numbers assigned to products are not changed. You can update the part number by reselecting the product attributes.

If you add or remove attributes defined on a class or change attribute values, these changes are not propagated to the generation method. You must manually update the method. For information on this, see "Updating a Generation Method with Attribute Changes" on page 153.

Creating Predefined Part Numbers

To create predefined part numbers, you create a matrix that contains one row for each possible combination of attribute values. The last entry in the row is the part number you want to assign to this combination. You can create the matrix manually, or the system can generate it automatically.

When the system generates the matrix, it assigns a random part number to each combination. You can accept this part number or replace it with one of your own.

When the user configures the product, the system searches the matrix for the combination of attribute values the user has chosen and assigns the corresponding part number to the product.

Only attributes with a list of values (LOV) domain can be used to create predefined part numbers. Before creating part numbers using this method, determine which combinations of attribute values are allowable. Assign part numbers only to these combinations.

Creating predefined part numbers is a process that is made up of the following tasks:

a Creating a part number generation record. This record names the part number generation method and specifies whether the method is dynamic or predefined.

b Selecting the desired attributes.

c Creating the part number matrix.

d Testing the part number matrix.

These tasks are described below.

To create a part number generation record

1 Navigate to Application Administration > Class Administration and query for the desired product class.

2 Click Part Number Definitions.

3 Click New to create a new record.

4 In the Name field, enter a name for the part number generation method.

This name should be unique and should indicate whether the method is dynamic or predefined.

5 In the Type field, open the menu and choose Predefined. Then step off the record.

The next step is to select the desired attributes.

To select the desired attributes

1 In the part number definition Name field, click the name you entered.

The name is a hyperlink. The Part Number Method view appears.

2 In the Attributes list, click New.

3 Open the drop-down menu in the new record and choose the desired attribute.

4 Repeat these steps until you have added all the attributes that you want to use for defining part numbers.

The last step is to generate a part number matrix.

To create the part number matrix

1 Click Attribute Matrix.

Attribute Matrix displays the part number matrix. There is one column for each attribute you selected. There is also a Part Number column and a Description column.

2 To generate the matrix automatically, click the menu button and choose Generate Part Numbers.

The system creates one record for each possible combination of attribute values. The system also generates a random part number for each combination.

You can also create the matrix manually by clicking New and creating a record for each desired attribute combination.

3 Review the matrix and verify that it is structured correctly.

   If you have not specified the correct attributes, click Attributes. Then add or subtract attributes as needed before regenerating the matrix.

4 Edit the part number for each attribute combination as desired.

   You can either accept the randomly generated part numbers or enter the desired part numbers. You can also enter a description for each combination. Users do not see the description.

5 To add records, click New.

   Enter an attribute value for each attribute, and enter a part number.

6 Delete records for unneeded attribute combinations as desired.

The last step is to test the part number matrix. You do this by creating a quote and selecting the product for which you have created a smart part number method. See "Viewing a Product's Smart Part Number in a Quote" on page 152.

Editing a Predefined Generation Method

You can edit a predefined generation method in several ways:

- Edit the name of the generation method
- Delete the generation method
- Edit the part numbers in a generation method's part number matrix
- Add or delete records in a generation method's part number matrix
- Regenerate the part number matrix, using different attributes

If you edit the name of a generation method or delete the method, the change is reflected in all product records to which the method is assigned. For example, you delete the generation method Predefined1. All product records that have Predefined1 as the Part Number Method, no longer have an assigned generation method.

If you edit the part number matrix for a generation method, the changes become effective immediately. The next time the product is added to quote, order, and so on, the revised part number scheme will be used. The part numbers assigned to products are not changed. You can update the part number by reselecting the product attributes.

If you add or remove attributes defined on a class or change attribute values, these changes are not propagated to the generation method. You must manually update the method. To do this, see "Updating a Generation Method with Attribute Changes" on page 153.

To edit a predefined generation method

1. Navigate to Application Administration > Class Administration and query for the desired product class.
2. Click Part Number Definitions.

   A list of the part number generation methods defined on the product class appears.
3. To delete a generation method, open the Part Number Definitions menu and choose Delete Record.

4 To edit a generation method name, click the method name in the Name field.

The Part Number Method view appears.

5 To edit the part number matrix, click the Attribute Matrix tab.

The part number matrix appears.

6 Edit existing records as desired.

7 To add records, click New.

Enter an attribute value for each attribute, and enter a part number.

8 Delete records for attribute combinations as desired.

Assigning a Generation Method to a Product

Part number generation methods defined on a product class are not inherited by the products in the class. You must manually assign the part number generation method to a product.

When you assign a generation method to a product, this method is used for generating part numbers in all new quotes, orders and so on.

To assign a generation method to a product

1. Navigate to Product Administration.
2. Query for the desired product.

   Alternatively, query for the desired product class to display all the products in the class.
3. Click in the product record's Part Number Method field, open the drop-down menu, and choose the desired generation method.

Viewing a Product's Smart Part Number in a Quote

The part number displayed in the Part # field throughout the application and in quotes, orders, and so on is the internally assigned part number. This part number is different than the smart part number, which displays in a separate field.

Before viewing a product's smart part number in a quote, you must assign a part number generation method to the product. See "Assigning a Generation Method to a Product" on page 151.

Assigning a generation method to a product does not cause a smart number to be generated in existing quotes containing the product.

To view a product's smart part number in a quote

1. Create a quote containing the product.
2. Navigate to Quotes > Line Items.
3. Highlight the desired product and click Line Item Detail.
4. Locate the Smart Part Number field.

   You may need to expand the Line Item Detail form to make the Smart Part Number field visible.

Updating a Generation Method with Attribute Changes

When you add or remove attribute definitions for a class, these changes are not propagated to smart part number methods defined on the class. If you modify the list of values domain for an attribute, these changes also are not propagated. You must manually update each smart part number method with changes to attributes.

You do this by validating the smart part number generation method. When you validate a generation method, the system does two things:

- If you have added or removed attributes, a pop-up message displays and recommends you edit the attribute list you are using for the generation method. For dynamic generation methods, you must modify section definitions and mappings. For predefined methods, you must edit the rows of the matrix.
- If you have changed attribute values for an attribute, the changes are added to the attribute values available for selection. For dynamic generation methods, you must edit the mappings to reflect the new values. For predefined methods, you must edit the rows of the matrix.

Choose one of the following procedures to validate a smart part number generation method.

To update a dynamic generation method with attribute changes

1. Navigate to Application Administration > Class Administration and query for the desired product class.
2. Click Part Number Definitions.

   A list of the part number generation methods defined on the product class appears.
3. To edit a generation method name, click the method name in the Name field.

   The Part Number Method view appears.
4. In Part Number Template, click the menu button and choose Validate Definition.

   If an attribute has been added, removed, or its name has been changed, a pop-up message appears recommending you revise the sections in Part Number Template.

   If an attribute's values have changed, the values available in Attribute Mapping are updated and no pop-up message appears.

5 Revise the sections defined in Part Number Template as needed.

6 Add or revise mappings in Attribute Mapping as needed.

The following procedure shows how to update a predefined generation method.

To update a predefined generation method with attribute changes

1 Navigate to Application Administration > Class Administration and query for the desired product class.

2 Click Part Number Definitions.

A list of the part number generation methods defined on the product class appears.

3 To edit a generation method name, click the word Predefined in the Type field.

The Part Number Method view appears.

4 In Attributes, click the menu button and choose Validate Definition.

If an attribute has been added, removed, or its name has been changed, a pop-message up appears recommending you revise the Attributes list.

If an attribute's values have changed, the values available for automatically generating a matrix are updated and no pop-up message appears.

5 Revise the Attributes list as needed.

6 Add, remove, or revise rows in the matrix as needed.

Querying for Products with the Same Generation Method

You can query for all the products that have the same part number generation method. This allows you to check whether you have duplicate generation method names and whether you have assigned generation methods correctly.

To query for products with the same generation method

1. Navigate to Product Administration.
2. Open the drop-down menu in the Part Number Method field.

   This menu lists all the part number generation methods that have been defined for all classes.
3. Verify that the menu does not contain duplicate names.

   If it does, this means you have defined a generation method with the same name on more than one product class. Consider editing the generation method names so that each name is unique.
4. In query mode, enter an asterisk (*) in the Part Number Method field and start the query.

   This displays all the product records for which a generation method has been selected.
5. Verify that each group of records with the same generation method name has the correct class name.

Product Bundles 9

This chapter explains how to create product bundles. A product bundle is a group of products that are sold together for a specified price.

Understanding Product Bundles

A product bundle is a group of products sold as a package. A simple product bundle is one where the user cannot change the items in the bundle or their quantity. A dynamic bundle is one where the user can either change the items that make up the bundle, or the quantity of items in the bundle, or both.

A product bundle is itself a product and has a product record. It can also have a part number. You price bundles by assigning them a list price.

When you create a bundle, it is added to the product master. This means you can add the bundle to any quote or order. Product packages that you create in a quote or order are bundles that are specific to that quote or order. They are not added to the product master.

You must create a bundle record for each product you add to a bundle. A bundle record has the following fields:

- **Product**. The product name.
- **Part #**. The product part number.
- **Description**. A brief description of the bundle. This does not display to users.
- **Quantity**. The quantity of the product you want to include in the bundle.
- **Sequence**. The order in which products in the bundle display in quotes and orders.
- **Forecastable**. A check mark in this field means that the product will be added to product forecasts when the bundle is included in a quote and the user updates the related opportunity.

Creating a Simple Product Bundle

A simple product bundle is a group of products offered as package. The user cannot change the items in the bundle or their quantity.

To create a simple product bundle, you first create a product record for the bundle. Then you add products to the bundle. After creating the bundle, refer to the *Pricing Administration Guide* to set up pricing.

Observe the following guidelines and restrictions when creating a simple product bundle:

- The quantity of a product in a bundle can be greater than one. When creating a quote or purchasing the bundle, users cannot change the quantity of a product in the bundle. The user cannot change which items are in the bundle.
- When users add bundles to quotes and orders, the products in the bundle display as line items beneath the bundle's product name.
- You can add a product bundle to another product bundle.
- You can add a customizable product to a bundle.
- When you add a bundle to a customizable product, the user can change the quantity of components in the bundle during a configuration session.
- You can convert a bundle to a customizable product and you can convert a customizable product to a bundle.

To create a product bundle

1. Navigate to Product Administration and create a product record.

   Enter the name of the bundle in the Product field.

2. Click Bundle Administration.

   The Bundle Administration list appears.

3. In Bundle Administration, click Modify then click New and create a new record.

4. In the new record, click in the Product field to display the select button. Click the select button to display the Pick Product dialog box.

5 In the dialog box, select a product.

The product record displays in the Bundle Administration list.

6 In the product record, edit the Description, Quantity, Sequence, and Forecastable fields as desired.

7 Repeat these steps for each product you want to add to the bundle.

8 Click done.

This releases the bundle for use by customers. A check mark displays in the Bundle check box in the product record form.

Modifying Simple Product Bundles

You can modify a simple product bundle by changing the items in the bundle or by changing the quantity of items. Modifying a product bundle releases a new version of the bundle.

Modifying a bundle affects in-process and existing quotes and orders. When users save a quote or order containing a bundle for which there is a new version, they receive a message indicating that.

To modify a product bundle

1. Navigate to Product Administration and query for the desired bundle.
2. Click Bundle Administration.

   The products in the bundle appear.
3. To edit a product's record, select the record and click Modify.
4. To add a product, open the Bundle Administration menu and choose New Record.
5. To delete a product, select its record, open the Bundle Administration menu, and choose Delete Record.
6. When finished, click Done.

   This releases a new version of the bundle for use by customers.

To Delete a Simple Product Bundle

You cannot delete the product record for a product bundle. However you can make the product bundle unavailable for use.

To make a product bundle unavailable

1. Modify the bundle to remove all its products.
2. In the bundle's product record, click Sales Product to remove the check mark.

   This removes the product bundle from the product picklist.
3. Remove the product bundle from all price lists.
4. Delete any pricing rules that refer to the product bundle.
5. Remove the product bundle from all customizable product relationships, and configuration rules. Validate the customizable products and release a new version.

Controlling How Bundle Components are Forecast

When you add a product to a bundle, you can put a check mark in the Forecastable field. This adds the product to forecasts when the bundle is included in a quote and the user updates the related opportunity.

To prevent bundle products from being added to product forecasts, do not put a check mark in the Forecastable field in Bundle Administration.

A Forecastable check box is also available in Quotes > Line Items. This allows you to add or remove a bundle and its products from product forecasts within individual quotes.

Build Customizable Products 10

This chapter describes how to build a customizable structure using the Product Designer.

Understanding Customizable Products

A customizable product is one that has configurable components. For example, you sell desktop workstations. At the time of purchase, the user can select from several types of disk drive, monitor, keyboard, and mouse to configure the workstation.

Another type of customizable product is one that has other customizable products as components. For example, you sell a telephone PBX system that includes 6 rack-mounted PC-based modules. Each module is configurable in a fashion similar to a desktop computer. The components of the PBX system form a product hierarchy. To configure the PBX, the user begins at the top with the PBX as a whole and works down through the hierarchy, configuring its components.

Customizable Product Versions and Work Space

Besides components, customizable products can also have a specially designed user interface for configuring the product, configuration rules, and special variables called resources and links.

These parts that make up a customizable product are stored together to form a product version. When you modify any of the parts of the product, you can release a new version. Customers see only the latest released version of a customizable product.

The product administrator has access to a special form of the customizable product, called the work space. The work space is an unreleased version of the product and is not available to users. The work space is like a workbench where the product administrator builds and revises the product before releasing it to customers.

You must create a work space for a customizable product to do the following things:

- **Add components**. You do this in the Product Designer.
- **Create a user interface for the product**. You can create special selection pages to display the configurable product options. You do this in the Product Designer and the Product UI Designer.
- **Create configuration rules**. These rules constrain attribute selection or selection of components. You do this in the Rule Designer.
- **Define resources.** Resources are special variables you use to track configuration quantities. You do this in the Resource Designer.
- **Define links**. Links are special variables contain information from Siebel business components or system values. You do this in the Link Designer.
- **Create Scripts**. Scripts are programs that execute at specific points in a configuration session. You create scripts in the Scripts Designer.

Products that are customizable solely because they have attributes, do not require a work space. However, if you want to write configuration rules regarding the attribute values, you must create a work space and use the Rule Designer. If you want to create special Web pages for selecting the attribute values, you must create a work space and use the Product UI Designer.

Product Designer

The Product Designer is where you add components to a customizable hierarchy and arrange them into a hierarchy. You can add single products, products from multiple classes, parts of product classes, or all of product classes. Products you add come from the product table.

The Product Designer is located in the Product Administration screen, Customizable Product tab.

Product UI Designer

The Product UI Designer is where you design the pages a user sees when they start a configuration session or when you enter validation mode. Several types of user interface themes are provided that govern the basic look and feel of the Web pages as well as the layout and types of controls.

When the user configures a customizable product, an instance of the product is created and presented in the user interface. For example, a user is creating a quote. The user selects a customizable product and clicks the Customize button. The system creates an instance of the customizable product and generates the browser pages that display it. The user then configures the product. When the user is finished, the user adds the configured product to the quote.

The Product UI Designer is located in the Product Administration screen, Customizable Product tab.

User Interface Property Designer

The themes and UI controls that you use to design pages in the Product UI Designer are controlled by Web template files. You can modify these templates by inserting variables in them and then associating these variables with items in a customizable product. For example, instead of displaying the attribute values Red, Green, Blue, you could define variables that call gif files to display the colors themselves. The User Interface Property Designer is where you define these associations, called user interface properties.

The User Interface Property Designer is located in the Product Administration screen, Customizable Product tab.

Resource Designer

Resources keep track of important configuration-related amounts in a customizable product. For example, you are designing a customizable product called Computer Model. This product has several choices of chassis, each with a different number of card slots. Several of the components in this product are expansion cards that consume these slots. To keep track of the number of slots available you could define a resource called Slots Available. When the user selects a chassis, a rule associated with the customizable product would add the number of slots in the chassis to a Slots Available resource. Similarly, when the user selects any type of expansion card, rules would decrease Slots Available by 1. In this fashion, you can monitor slot usage and write rules to prevent misconfiguration of the product.

The Resource Designer is located in Product Administration screen > Configuration Designer.

Link Designer

Linked items provide access to other types of information besides products. You can define links to fields in a business component, to the login name of the user, or to the current system date. This lets you write rules that affect only certain login names, are conditioned on dates, or are conditioned on business component information.

The Link Designer is located in the Product Administration screen, Configuration Designer tab.

Rule Manager

When you define components and attributes for a customizable product, you need a mechanism to restrict the combinations of these to the configurations you sell. For example, you sell shirts in three sizes and three colors. However, not all sizes come in all colors. You need a way to restrict the colors for each size to the ones you sell. You do this by writing configuration rules that define the allowable configurations of your products.

Configuration rules can prevent the user from picking an item if another item has already been picked. They can also automatically add an item when another item has been picked. Configuration rules can also be used to give up-sell messages or recommendations to the user when they pick an item.

The Rule Designer is located in the Product Administration screen, Configuration Designer tab.

Script Designer

The Siebel system provides a set of configuration-related events and methods. These allow you to write scripts that add procedural logic to the configuration process. When the user selects certain items or does things like updating the shopping cart, you can use scripts to check the configuration, verify and adjust pricing, or forward information to other applications. Scripts can be associated with items and can be set to trigger when certain defined events occur.

The Script Designer is located in the Product Administration screen, Configuration Designer tab.

Understanding Relationships

You add items to a customizable product by defining relationships. A relationship can be defined for a single product, a group of products, or the products in a class.

The relationships you define for a customizable product are component type relationships. This means the items in the relationship are components of the customizable product. For example, you define a relationship called Hard Drives for the customizable product Desktop Computer. You specify that it contains all the products assigned to the Disk Drive class. This makes the disk drives in this class components of the customizable product Desktop Computer.

Relationships are analogous to the branches of a tree. The main trunk is the root of the customizable product. Each branch is a relationship. The leaves at the end of each branch are the components you add to the relationship.

Relationships form the framework of a customizable product. They are also the framework underlying the user interface you design for the product. For example, you sell configurable computers. The buyer can choose among several monitors, several keyboards, and several CD-ROMs when configuring a computer. You could create a relationship called Monitors, another called Keyboards, and one called CD-ROMS. You would then specify the products to include in each relationship. You could then design the user interface to present monitors, keyboards, and CD-ROMs each on a separate selection page.

When you design a customizable product, begin by defining a framework of relationships. Keep in mind that each relationship represents a distinct, configurable part of the product.

Figure 7 on page 168 shows a relationship framework in a customizable product.

- Relationship 1 contains a single product, Product 1
- Relationship 2 contains all the products in product class, Class 1
- Relationship 3 contains Product 2, Product 3, and Product 4, each from a different product class

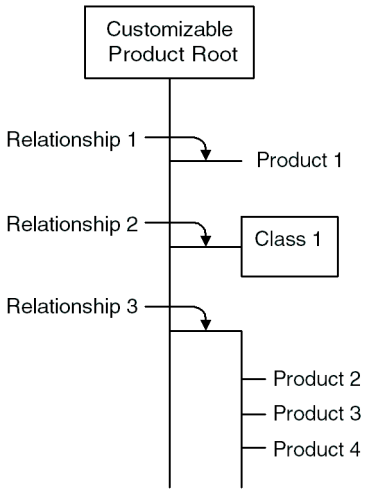

Figure 7. Customizable Product Relationships

Customizable products and the product class system both include hierarchies. However, these hierarchies differ in important ways. In the product class system, inheritance is used to propagate attribute definitions downward through the class system. By contrast, inheritance plays no role in the hierarchy of components in a customizable product. Attributes inherited by a customizable product because of its membership in a product class do not propagate to the component products in the customizable product.

For example, a customizable product belongs to a product class that has the attribute Color (red, green, blue). The customizable product as a whole inherits this attribute but its components do not. For example, if the customizable product is a laptop computer, this means the laptop comes in three colors, red, green, or blue.

However, these colors are not inherited by any of the components of the laptop. For example, if the laptop has a CD-ROM, it does not inherit these colors. The color attribute of the CD-ROM (if it has one) is defined in the product class from which it comes, not in the customizable product in which it resides.

When you create a relationship for a customizable product, you create a record in the Product Designer that contains the following fields:

- **Relationship Name**. The name of the relationship. This name displays in the selection pages in a configuration session. In the Product Designer, the Relationship name displays as a file folder. When you expand this folder, the products in the relationship display beneath it.
- **Class Name**. Click in this field to display a dialog box that lists all product classes. Select the desired product class. If no class name is specified, you can add products from anywhere in the class system.
- **Define Domain**. Click in this field to display a dialog box that lists the products in the class you selected. Select the products you want to add from this class.
- **Default Product**. Specifies the default product presented to the user in selection pages during a configuration session. You select a default product from the dialog box that displays when click in the Define Domain field.
- **Min Cardinality**. Enter the minimum quantity of items that must be selected from this relationship. The quantity can be zero.
- **Max Cardinality**. Enter the maximum quantity of items that the user can select from this relationship. The quantity can be zero.
- **Default Cardinality**. Enter the quantity of the default product you want display in selection pages at the beginning of a configuration session. The quantity can be zero.

- **Domain Type**. The domain type specifies the domain of the items you add to the relationship:
  - **Product**. The relationship contains a single product. The product can be a customizable product.
  - **Class**. You can define the relationship domain to be all or part of a product class and its subclasses. You must manually select which products to include in the relationship. If you add or remove products from the product class or its subclasses, these changes are not propagated to the relationship. You must manually edit the relationship domain to update it with changes to the product class.
  - **Dynamic Class**. The relationship contains all of the products in a product class and its subclasses. If you add or remove products from the product class or its subclasses, the changes are propagated to the relationship when you refresh the customizable product work space. This domain type provides the best performance and is the recommended one to use.

When you expand the Relationship Name folder, each product in the relationship displays as a separate record with the following fields.

- **Relationship Name**. Products in a relationship display beneath the relationship to which they belong. The name you enter for the product in this field is displayed in selection pages during a configuration session.
- **Product**. The name of the product in the product table. This field is a hyperlink. When you click it, the product table entry for the product appears.
- **Sequence Number**. Enter a number to set the order of display of relationship items within the relationship in selection pages. The item with a sequence number of 1 displays first, and so on. If you do not enter sequence numbers, the items display in selection pages in the order shown in the Product Designer.

  For relationships with domain type dynamic class, entering a sequence number is not useful. The sequence number information is lost when you refresh the customizable product work space. This is because the current contents of the product class are copied into the customizable product.

Understanding Cardinality

When you define a relationship, you can specify a minimum, maximum, and default cardinality. Cardinality refers to the quantity of the products the user can select from a relationship. For example, you define a relationship called Hard Drives. It contains a 20 GB drive and a 30 GB drive. If you set the minimum cardinality to 2, the user must pick 2 items from this relationship. The user can do this in any of the following ways:

- Pick one 20 GB drive and one 30 GB drive
- Pick two 20 GB drives
- Pick two 30 GB drives

The three types of cardinality you can define for a relationship are as follows:

- **Minimum Cardinality**. Governs whether or not selecting items from this relationship is optional or is required. If you set the minimum cardinality to 0, selecting items is optional. If you set the minimum cardinality to greater than 0, the user must select that number of items from the relationship.
- **Maximum Cardinality**. Sets the maximum number of items that the user can select from a relationship. If you set the minimum cardinality to greater than 0, you must set the maximum cardinality to a number at least as large If you do not enter a maximum cardinality, the default is 999. To revise this default, refer to "Revising the System Default Cardinalities" on page 465.
- **Default Cardinality**. Specifies what quantity of the default product is automatically added to the initial solution that the user sees. Default cardinality must be equal to or greater than the minimum cardinality and must be less than or equal to the maximum cardinality.

  If you specify a default cardinality and do not specify a default product, the system uses the first product that displays when you expand the relationship folder in the Product Designer.

Table 17 describes several combinations for setting cardinality. The table shows what the user will see in the initial solution and what actions that the user can take. In the table, N is the quantity of the default product in the initial solution. In all the cases where the Min Card is greater than 0, the user can substitute other products for the default product.

Table 17. Combinations of Cardinality

Min Card	Default Card	Max Card	System Adds Default Product?	User Pick Reqd?	Initial Solution	User Actions Allowed
= 0	= Min Card	> Default Card	No	No	No items	Increase item quantities to Max Card.
= 0	> Min Card	= Default Card	No	No	N = Max Card	Decrease Item quantities to 0 but cannot increase them.
= 0	> Min Card	> Default Card	No	No	N = Default Card	Increase item quantities to Max Card or decrease them to 0.
> 0	= Min Card	= Default Card	Yes	Yes	N = Min, Default, Max	Cannot increase or decrease item quantities.
> 0	= Min Card	> Default Card	Yes	Yes	N = Min	Can increase item quantities to Max Card but cannot decrease them.
> 0	> Min Card	= Default Card	Yes	Yes	N = Default	Can decrease item quantities to Min Card but cannot increase them.
> 0	> Min Card	> Default Card	Yes	Yes	N = Default	Can decrease item quantities to Min Card or increase them to Max Card.

Creating a Customizable Product Work Space

A work space is what distinguishes a customizable product from a simple product. The work space is a context or container to which you add the component products, rules, resources, links, scripts, and user interface definition for a customizable product.

When you have tested the customizable product and are ready to publish it, you release a new version of the product. Releasing a version copies the current work space, assigns a version number to it, and makes the new version available to users.

The most recently released version is what users see when they configure the customizable product. When you release the first version of a customizable product, the system places a check-mark in the Customizable Product check box in the product record.

Once you have created a work space you must lock it before you can make changes to a customizable product.

To create a customizable product work space

1. Navigate to Product Administration.
2. In Products create a new product or select the desired product.
3. Click the Customizable Product tab.

   The Customizable Product > Product Versions view appears. The Versions tab contains no records.
4. In Lock/Unlock Product, open the menu and choose Create Work Space.

   A new record appears in Lock/Unlock Product. A work space record appears in the Versions tab.
5. Click in the Locked Flag field.

   A check mark appears in the Locked Flag field. Your login ID appears in the Locked By field. The system date and time appear in the Locked Date field.

   You can now begin building the new customizable product.

Refreshing the Work Space

If a customizable product contains relationships of type Dynamic Class, refreshing the work space copies a new instance of these product classes into the relationships. This means a fresh copy of all the products in the class become part of the customizable product instance.

For example, the number of products in a class has changed. You have defined a relationship of type Dynamic Class that specifies this product class. When you refresh the work space, the revised product class is copied to the relationship from the product table. When you view the relationship in the Product UI Designer or in Validate mode, the new products display.

Relationships of domain type Class and Product are not updated from the product table when you refresh the work space.

Refreshing the work space updates the products or attributes in a customizable product. The configuration rules, resource definitions, link definitions, and scripts that are part of the customizable product are not updated to reflect changes. You must manually make these updates.

To refresh the work space

1. Open the Customizable Product Show menu and choose Product Versions.

   The Product Versions view appears. You can also refresh the work space by opening the Product Designer menu.

2. Open the Lock/Unlock Product menu and choose Refresh Work Space.

Selecting and Locking a Customizable Product

Before you can build or edit a customizable product, you must lock its work space. This prevents others from modifying the work space. You cannot lock versions that have already been released.

To select and lock a customizable product

1. Navigate to Product Administration.
2. In Products, select the desired customizable product.
3. Click the Customizable Product tab.

   The Customizable Product > Product Versions view appears.
4. In Lock/Unlock Product, click in the Locked Flag field.

   A check mark appears in the Locked Flag field. Your login ID appears in the Locked By field. The system date and time appear in the Locked Date field.

   To unlock the product, click in the Locked Flag field.

Adding a Single Product

Use this procedure to create a relationship that contains a single product.

The product you select must be orderable. To make a product orderable, place a check mark in the Orderable check box in the product record.

When you are finished adding products, you can verify your work by validating the customizable product. Validating a customizable product displays the selection pages a user sees during a configuration session. To validate the customizable product, open the Products Designer menu and choose Validate.

To add a single-product relationship

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. Open the Customizable Product Show menu and choose Product Designer.

   The Product Designer view appears.
4. Add a new record.
5. Enter a relationship name.
6. Click in the Product field to expose the select button. Click the select button.

   A dialog box appears that displays all the available products.
7. In the dialog box, select the desired product.

   The record appears in the Product Designer.
8. Set the Min, Max and Default Cardinalities.

Adding Products by Using the Class Domain

This method of adding products does not maintain a connection to the class system. When you refresh the customizable product work space, relationships are not updated. For example, if you assign a new product to a class, this product is not added to the relationship containing this class when you refresh the work space or release a new version of the customizable product. Use this method when you want to keep the relationship contents static or when you want to add only some of the products in a product class.

The products you select must be orderable. To make a product orderable, place a check mark in the Orderable check box in the product record.

When you are finished adding products, you can verify your work by validating the customizable product. Validating a customizable product displays the selection pages a user sees during a configuration session. To validate the customizable product, open the Products Designer menu and choose Validate.

NOTE: Adding a small number of products to a relationship from a large product class requires that the entire class be searched each time the customizable product is instantiated. This can adversely affect performance. Consider defining customizable products to avoid this.

To add products by using the Class domain

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. Open the Customizable Product Show menu and choose Product Designer.

   The Product Designer view appears.
4. Add a new record.
5. Enter a Relationship name in the Relationship Name field.
6. Open the drop-down menu in the Domain Type field and choose Class.
7. Click in the Class Name field and then click the select button.

   A dialog box appears that contains one record for each class and for each subclass in the class system. Selecting a class selects all of its subclasses.

8 In the dialog box, click the select button to select a class.

9 Click in the Define Domain field and then click the select button.

The Define Relationship Domain dialog box appears and displays all the products in the class.

10 Use the buttons and fields in the dialog box to select the products you want to add to the relationship:

- **Add column**. Click the word Add in the record to add the product to the relationship. A check mark displays in the "Is in domain" field.
- **Query button**. Queries for the desired products in the class.
- **Close button**. Closes the dialog box.
- **Add All button**. Adds all the products in the class to the relationship.
- **Set as Default button**. Adds the product to the relationship and makes it the default product. In the Product Designer, the product name displays in the Default Product field at the relationship level.
- **Clear Default button**. Removes the product from the relationship's Default Product field. Does not remove the product from the relationship.
- **Delete button**. Removes the product from those you have selected to be in the relationship. Removes the check mark from the "Is in domain" field. Does not remove the product from the product class.
- **Delete All button**. Removes all the products from the relationship. No products display a check mark in the "Is in domain" field. Does not remove the products from the product class.

11 When you have finished adding products, click Close in the dialog box.

In the Product Designer, the relationship icon displays as a folder.

12 Enter the Min, Default, and Max Cardinalities for the relationship as needed.

13 Click the folder to display the products you added.

Verify that the relationship is defined properly, that the default product is correct, and that all the products you want to add are present.

14 Enter the Min, Default, and Max Cardinalities for each item in the relationship as needed.

15 Remove the check mark from the Forecastable field for items as needed.

Removing the check mark means the item will not be included in product forecasts when the opportunity is updated for quotes, orders, and so on contained the customizable product.

16 For each product in the relationship, enter a sequence number in the Sequence Number Field.

The item with sequence number 1 displays first within the relationship in selection pages. If your display is not wide enough to show the Sequence Number field, manually adjust column widths to bring the Sequence Number field into view.

Adding Products Using the Dynamic Class Domain

This method of adding products maintains a connection to the class system. When the work space is refreshed, Dynamic Class relationships are updated from the class system. For example, if you add a new product to a class in the class system, this product is added to the relationship containing this class when you refresh the work space or release a new version of the customizable product.

When you refresh the work space to update the contents of the relationship, you must reenter the sequence numbers in the relationship definition.

The products you select must be orderable. To make a product orderable, place a check mark in the Orderable check box in the product record.

When you are finished adding products, you can verify your work by validating the customizable product. Validating a customizable product displays the selection pages a user sees during a configuration session. To validate the customizable product, open the Products Designer menu and choose Validate.

To add products using the Dynamic Class domain

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. Open the Customizable Product Show menu and choose Product Designer.

   The Product Designer view appears.
4. Add a new record.
5. Enter a Relationship name in the Relationship Name field.
6. Open the drop-down menu in the Domain Type field and choose Dynamic Class.
7. Click in the Class Name field and then click the select button.

   A dialog box appears that contains one record for each class and for each subclass in the class system. Selecting a class selects all of its subclasses.
8. In the dialog box, select a class.

   The class displays in the Class Name field.

9 Click in the Define Domain field and then click the select button.

A dialog box appears that displays all the products in the class.

10 In the dialog box, click Add All.

A check mark displays in the "Is in domain" field for all the products in the class.

11 Use the following buttons to select a default product:

a **Set as Default button**. Adds the product to the relationship and makes it the default product. In the Product Designer, the product name displays in the Default Product field at the relationship level.

b **Clear Default button**. Removes the product from the relationship's Default Product field. Does not remove the product from the relationship.

12 When you have finished adding products, click Close in the dialog box.

In the Product Designer, the relationship icon displays as a folder.

13 Enter the Min, Default, and Max Cardinalities for the relationship as needed.

14 Click the folder to display the products you added.

15 Enter the Min, Default, and Max Cardinalities for each item in the relationship as needed.

16 Remove the check mark from the Forecastable field for items as needed.

Removing the check mark means the item will not be included in product forecasts when the opportunity is updated for quotes, orders, and so on contained the customizable product.

17 For each product in the relationship, enter a sequence number in the Sequence Number Field.

The item with sequence number 1 displays first within the relationship in selection pages. If your display is not wide enough to show the Sequence Number field, manually adjust column widths to bring the Sequence Number field into view.

Adding a Group of Products from Different Classes

The products you add to a relationship do not have to be from the same class. You can group products from several classes or products not assigned to a class into one relationship.

You do this by creating a relationship of domain type Class but without specifying a class. This allows you to select products from anywhere in the class system.

You can do anything with this relationship that you can do with other class-type relationships such as creating resources, configuration rules, and links.

The products you select must be orderable. To make a product orderable, place a check mark in the Orderable check box in the product record.

When you are finished adding products, you can verify your work by validating the customizable product. Validating a customizable product displays the selection pages a user sees during a configuration session. To validate the customizable product, open the Products Designer menu and choose Validate.

This method of defining a relationship domain requires a search throughout the class system each time the customizable product is instantiated. This can have an adverse impact on performance. Avoid using this method, if possible.

Adding groups of products from different classes

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. Open the Customizable Product Show menu and choose Product Designer.

   The Product Designer view appears.
4. Add a New Record.
5. Enter a Relationship name in the Relationship Name field.
6. Open the drop-down menu in the Domain Type field and choose Class.

7. Click in the Define Domain field and open the dialog box.

   The Define Relationship Domain dialog box appears. Because you have not specified a class, the dialog box displays all products from all classes and subclasses.

8. Use the buttons and fields in the dialog box to select the products you want to add to the relationship:

   a. **Add column**. Click the word Add in the record to add the product to the relationship. A check mark displays in the "Is in domain" field.

   b. **Query button**. Queries for the desired products in the class.

   c. **Close button**. Closes the dialog box.

   d. **Add All button**. Adds all the products in the class to the relationship.

   e. **Set as Default button**. Adds the product to the relationship and makes it the default product. In the Product Designer, the product name displays in the Default Product field at the relationship level.

   f. **Clear Default button**. Removes the product from the relationship's Default Product field. Does not remove the product from the relationship.

   g. **Delete button**. Removes the product from those you have selected to be in the relationship. Removes the check mark from the "Is in domain" field. Does not remove the product from the product class.

   h. **Delete All button**. Removes all the products from the relationship. No products display a check mark in the "Is in domain" field. Does not remove the products from the product class.

9. When you have finished adding products, click Close in the dialog box.

   In the Product Designer, the relationship icon displays as a folder.

10. Enter the Min, Default, and Max Cardinalities for the relationship as needed.

11. Click the folder to display the products you added.

12. Enter the Min, Default, and Max Cardinalities for each item in the relationship as needed.

13 Remove the check mark from the Forecastable field for items as needed.

Removing the check mark means the item will not be included in product forecasts when the opportunity is updated for quotes, orders, and so on contained the customizable product.

14 For each product in the relationship, enter a sequence number in the Sequence Number Field.

The item with sequence number 1 displays first within the relationship in selection pages. If your display is not wide enough to show the Sequence Number field, manually adjust column widths to bring the Sequence Number field into view.

Adding a Customizable Product

You can add customizable products as components of other customizable products. This means you can create customizable products that are sub-assemblies and then include them as components in the final product. For example, you sell a configurable power supply and a configurable gearbox as part of an industrial lathe. You can create one customizable product for configuring the power supply and one for configuring the gearbox. You can then add both of these component customizable products to the industrial lathe customizable product.

In the Product Designer, when you add a customizable product to a relationship, its configurable parts do not display. Instead, the customizable product displays as a single product.

When you edit a customizable product and release it, the changes propagate to all customizable products containing it.

Use the following procedures to add a customizable product:

- To add a customizable product that is not assigned to a class, use the procedure for adding a single product.
- To add a customizable product that is a member of a class, use the procedure for adding a class.

Editing a Relationship Definition

You can only edit the current work space of a customizable product. You cannot edit a version that has already been released. All the fields in a relationship definition can be edited except the relationship name. Changes are not propagated to other parts of the customizable product.

When you are finished editing, you can verify your work by validating the customizable product. Validating a customizable product displays the selection pages a user sees during a configuration session. To validate the customizable product, open the Products Designer menu and choose Validate.

To edit a relationship definition

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. Open the Customizable Product Show menu and choose Product Designer.

   The Product Designer view appears.
4. Select the desired relationship.
5. Edit the fields in the relationship record as desired.
6. Save the relationship record.
7. Revise configuration rules, resource definitions, link definitions, and scripts as needed to reflect the changes.

Updating Product Information in Relationships

You add items to a customizable product by defining relationships and then adding products to the relationships. Relationships have several domain types, which are updated with product class changes as follows when you refresh the work space.

- **Product**. New and revised attribute definitions are added.
- **Class**. New and revised attribute definitions are added. New products are not added. Products that have been removed from the class are not removed from the relationship.
- **Dynamic Class**. New and revised attribute definitions are added. New products are added. Products that have been removed from the product class are removed from the relationship.

The updates appear in the current work space. To make them available to users, you must release the customizable product.

To update product information by refreshing the work space

1. Modify class attributes as needed.
2. Add or remove products in the class system as needed.
3. Navigate to Product Administration.
4. Select and lock the desired customizable product.
5. Open the Lock/Unlock menu and choose Refresh Workspace.

Deleting Products

You can only delete products from the current work space of a customizable product. You cannot delete products from a released version. You can delete relationships or products included within a relationship.

Changes are not propagated to other parts of the customizable product. For example, if you delete a product from a relationship, configuration rules for that product are not deleted.

If you delete a product from a relationship that has domain type Dynamic Class, the product will be added back to the relationship when you refresh the work space or release the product. This is because the product still exists in the product class. When you refresh the work space or release the product, the relationship is updated so that it contains all the products in the product class and the current attribute definitions.

To avoid this, you can change the relationship domain type to Class. This breaks the connection to the product class system and prevents any further updates of the relationship. You can also leave the domain type unchanged and remove the product from the product class.

When you are finished deleting products, you can verify your work by validating the customizable product. Validating a customizable product displays the selection pages a user sees during a configuration session. To validate the customizable product, open the Products Designer menu and choose Validate.

To delete products

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. Open the Customizable Product Show menu and choose Product Designer.

   The Product Designer view appears.
4. To delete a relationship, select the desired relationship record. To delete a product within relationship, expand the relationship and select the product record.
5. Open the Product Designer menu and choose Delete Record.
6. Revise configuration rules, link definitions, and resource definitions as needed.

Deleting a Customizable Product's Structure

When you delete a customizable product's structure, all previously released versions of the customizable product are deleted. Its rules, user interface definition, resources, links, and scripts are also deleted.

Deleting a customizable product's structure affects all existing quotes containing the product. Remove the customizable product from all such quotes first.

To delete a customizable product's structure

1. Remove the customizable product from all existing quotes.
2. Navigate to Product Administration.
3. Select and lock the desired customizable product.
4. Open the Lock/Unlock Product menu and choose Delete Record.

   The current work space and all released versions are deleted. All other aspects of the customizable product, including the UI design, links, resources, and scripts are also deleted.

   In the product record, the check mark in the Customizable check box is removed.

Release and Manage Customizable Products 11

This chapter describes how to manage customizable products. This includes setting up a work space.

Understanding Bundles as Customizable Products

A bundle is a group of items sold as one product and is a special form of customizable product that has the following characteristics:

- Bundles created in Bundle Administration also display in Customizable Products > Versions and in Product Designer.
- A bundle is made up of one or more relationships that have a Product domain. Each relationship adds only one product to a bundle.
- Bundles do not include a UI definition, configuration rules, links, resources, or scripts. Bundles do not include selection pages. Users do not configure a bundle by starting a configuration session.
- The quantity of a product in a bundle is determined by the Default Cardinality of the product.
- Bundles cannot be designated as class-products.

If you add items to a bundle that are not allowed, they are ignored. For example, if you define configuration rules for a bundle, they are ignored. When you convert bundles to regular customizable products, ignored items then become effective.

If you convert a customizable product to a bundle, only the items within the scope of a bundle are used. All other items, such as configuration rules, are ignored.

For more information on converting bundles, see "Converting a Bundle to a Regular Customizable Product" on page 210 and "Converting a Regular Customizable Product to a Bundle" on page 211.

Regular customizable products are added to quotes using Siebel eAI. Bundles are added to quotes using internal code.

Understanding Customizable Assets and Delta Quotes

When creating a quote, users can choose to add to or modify customizable products the customer has already purchased. The customer's already-purchased customizable products are called customizable assets. The Quotes user creates a quote and selects a customer's customizable asset. The user then starts a configuration session and modifies the asset by adding or removing items. The user then saves the changes to the quote. The customizable asset, with revisions, displays in Quotes > Line Items.

In Quotes > Line Items > Line Item Detail, each item in the customizable asset displays a status in the Delta Status field:

- **Existing**. All items that were not changed. These items display no price.
- **New**. All items that have been added or for which the quantity increased. If the quantity of an item increased, the original quantity is shown with status Existing. A second entry in the quote shows the increase, has status New, and displays a price.
- **Removed**. All items that were removed or decreased in quantity. If the quantity of an item is reduced, the new quantity is displayed with the status Existing. A second entry in the quote shows the reduction, and has status Removed. These items display no price.
- **Modified**. All items for which attribute settings have changed. These items display a price.

Delta Quotes differ from Favorites in that only new or changed items from the configuration session have a price. Unchanged items are listed at zero price. When you add a Favorite to a quote, all the items in the customizable product have a price. Favorites are new products being sold for the first time. Delta Quote products are items being sold as add-ons or replacement components for products you have already sold.

The eConfigurator engine uses the name of the customizable asset's product root to determine what customizable product to load for the quote configuration session. If a new version of the customizable product has been released, the new version is used to modify the customizable asset. Any configuration conflicts that result are displayed during the configuration session and must be resolved. Item pricing is not maintained during the configuration session and should be ignored. Pricing is computed when the user saves the configuration to the quote.

The user adds a customizable asset to a quote by creating a quote and then clicking Delta Quote in Quotes > Line Items. This displays the Customizable Asset dialog box containing all the customizable product assets that have been configured for the account.

Understanding Auto Match

If a quote, asset, or order contains a customizable product configuration based on an out-of-date version of the product, Auto Match can compare the old version with the current version of the product and make limited changes to bring the quote, asset, or order up to date automatically. The user does not have to configure the product again.

Auto Match works as follows:

- Auto Match is triggered when the system determines that the version in the quote, order, or asset is not the current version.
- Auto Match compares the relationships and their contents in the quote, asset or order to the current version of the product.
- Auto Match identifies items in the old version that are not in the same relationship as items in the new version. An item can be a product or product class. These items in the old version are misclassified items. The relationship containing them is the old relationship. The relationship in the current product that contains the items is the new relationship.
- If the old relationship and the new relationship have a common parent, typically the product root, Auto Match will automatically move the items to the new relationship in the quote, order or asset. Auto Match does this by either changing the relationship name or by adding a new relationship.
- If the old relationship and new relationship do not have the same parent, the user receives and error message and must configure the product again.
- If the current version has a lower max cardinality for a relationship, this cardinality is enforced in the version in the quote, order, or asset. For example, the cardinality of Relationship A has been reduced from 10 to 8 in the current version. In a quote, Relationship A contains 10 items. Auto Match will remove two items from the quote.

- If a relationship has been removed from the current version but is included in a quote, order, or asset, Auto Match will attempt to move its items to a relationship at the same level. For example, Relationship A, containing 10 items, has been removed from the current version. A quote has the previous version of the customizable product, including Relationship A with 10 items. Auto Match will try to move all 10 items to other relationships at the same level, while observing maximum cardinality restrictions. Any excess items are removed.
- Auto Match only compares the physical structure of the product's current version to that in the quote, asset, or order. It does not consider configuration rules. For example, if the old version contains Product A, and Product A would be excluded in the new version, Auto Match does not detect this.

Auto Match is implemented as a business service and is not enabled by default. To turn Auto Match on, refer to "Enabling Auto Match" on page 460.

Understanding Finish It!

"Finish It!" is a button that appears in configuration session selection pages. This button is active under the following conditions:

- The configuration session contains a relationship that has a minimum cardinality greater than zero.
- No default product has been defined for this relationship.
- The user has not selected the number of products from this relationship required by the minimum cardinality.

These relationships are called unsatisfied quantity relationships. In selection pages, a red asterisk displays next to the relationship name and next to the item in the relationship, indicating that the user must make a selection.

When the user clicks Finish It!, the eConfigurator engine adds items to the solution from all unsatisfied quantity relationships so that minimum cardinalities are met or exceeded. The eConfigurator engine makes arbitrary selections from these relationships. You cannot specify which products will be selected by setting the sequence of the product in the relationship.

For example, you have defined a relationship called Keyboard. This relationship has a minimum cardinality of 1 and no default product. This causes the Finish It! button to become active in configuration sessions. When the user clicks Finish It!, the eConfigurator engine adds a keyboard to the solution.

If you do not want the Finish It! button to be active during configuration sessions, specify default products for all relationships with a minimum cardinality greater than zero.

Testing a Customizable Product (Validation Mode)

You test a customizable product by selecting validation mode. This creates an instance of the customizable product and presents its selection pages. You can test configuration rules, the user interface, and pricing exactly as if you were a user.

Validation mode creates an instance of the customizable product from its current work space, not from a released version of the product. If you want to troubleshoot a problem in the most recently released version, you can do this by reverting the work space to the most recently released version. This overwrites any changes you have made in the current work space.

To test a specific group of configuration rules, set all the rules you don't want to test to inactive and then go to validation mode to test the desired rules. Then activate configuration rules one at a time as needed and return to validation mode to test the result.

If you have purchased ePricer, be sure to fully validate all component-based pricing adjustments and pricing factors. If you are using automatic pricing updates, verify that selection pages redisplay fast enough after each user action. If redisplay is too slow, consider switching to a base theme for the user interface that uses manual price updates. When in validation mode, the system uses the price list assigned to a special quote called ModelValidation.

Verify that user access is set up correctly for all the components of the customizable product. Do this by checking the categories to which the customizable product and all its components are assigned. Then check the access control groups assigned to these categories and associated catalogs. Users who will configure the product must have access permission to the product and all its components. You can check category assignments in Product Administration > Category or in Catalog Administration.

Validate a customizable product at regular intervals while you are developing it. For example, after you enter a block of related configuration rules or after customizing the selection pages, go to validation mode and check your work.

If your customizable products are complex, consider developing written test plans that exercise all the configuration rules and all expected user behaviors. In particular, be sure to test for unexpected or incorrect user behaviors in order to rule out unexpected responses from the eConfigurator engine.

You enter validation mode by clicking the Validate button, which located in the views where you work with customizable products, such as the Product Versions view, and the Rule Designer.

To test a customizable product

1. Make changes to the customizable product.

   For example, add configuration rules in the Rule Designer.

2. Open the list menu and choose Validate.

   For example, open the Rules List menu in the Rule Designer and choose Validate.

If you want to test pricing, you must associate a price list with the ModelValidation quote. This is a special quote provided for validating customizable products.

To test customizable product pricing

1. Navigate to Quotes.
2. Query for the ModelValidation quote.
3. Assign the desired price list to the ModelValidation quote.
4. Navigate to Product Administration > Product Versions.
5. Select and lock the desired customizable product.
6. Click Validate.

   The system creates an instance of the product and displays its selection pages.

Releasing a Customizable Product for Use

To make a customizable product available to users, you must release it. Releasing a customizable product creates a new version of it and makes this version available for use. Releasing a version also refreshes the work space.

When creating a customizable product, discard your errors by reverting to the most recently released version. Do not discard them by releasing the product. If the most recently released version contains errors, then reverting copies the released version, with its errors, into the work space.

When you release a product, the system refreshes it in the current work space first, and then releases the refreshed instance. This means that all released versions contain the most recent information, such as class structure, available at the time of release. You cannot modify or delete a released version of a customizable product.

Before you release a customizable product, you can set an effective date. When you release the product, it becomes available to users on the effective date. This lets you release several versions of a customizable product and have them become available based on dates.

To identify which version of a customizable product to use, the system begins with the most recently released version. If it cannot be used because the effective date has not arrived, the system looks at the previous version. This process continues until the system finds a version it can use (one with no effective date, or an effective date that has passed).

Table 18 shows how effective dates and versions work together. In the examples, the effective dates are in the future unless specified:

Table 18. How Effective Dates and Versions Work Together

Action	Result
Customizable product A has no released versions. You release version 1 without an effective date.	Version 1 is available to users immediately.
Product A has version 1, which does not have an effective date. You release version 2 with an effective date.	Prior to the effective date, users receive version 1. On the effective date, users receive version 2.
Product A has version 1, which has an effective date. You release version 2, which has no effective date.	Version 2 is available to users immediately. Version 1 is superseded and will never be available to users.
Product A has two released versions. Version 1 has an effective date of June 1. Version 2 has an effective date of July 1. The current date is May 1.	On June 1, the user receives version 1. On July 1, the user receives version 2.
Product A has two released versions. Both version 1 and version 2 have the same effective date.	On the effective date, the user receives version 2.
Product A contains customizable product B. Product A has no effective date and no released versions. Product B has version 1, which does not have an effective date and version 2, which does have an effective date. You release product A.	Before the product B effective date, the user receives version 1 of product A, and version 1 of product B. On the product B effective date, the user receives version 1 of product A and version 2 of product B.

These behaviors apply to in-process quotes, existing quotes, and new quotes. When users open an in-process or existing quote, they receive a message if there is a conflict because of a new version of the product.

When remote users synchronize databases, they receive all released versions of a customizable product not already on the local machine, including those versions with effective dates. Effective dates work on the local machine as described above.

Observe the following guidelines when using effective dates for customizable products:

- Carefully coordinate configuration rules that have effective dates with the effective date of the customizable product. If you know that you must release a new version of the product because of time-sensitive configuration rules, consider using a version effective date rather than effective dates on configuration rules.
- If you want to release two versions such that the second version supersedes the first version on an effective date, be sure to fully analyze the possible business impacts of the new version on in-process, and existing quotes.
- The effective date cannot be modified after a version is released. Instead, you can release a new version with a different effective date.

To release a customizable product

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. To specify an effective-date, enter a date in the Required Start Date field of the Work Space record.
4. Save the record.
5. In Lock/Unlock Product, click Release New Version to release a new version of the product.

   A new record appears in the Versions list. Its version number displays in the Version field. The Required State Date field becomes read-only.

Reverting to the Most Recently Released Version

If you release a customizable product and then make changes to its current work space, you can discard all the changes and revert to the most recently released version of the product.

Discard your errors by reverting to the most recently released version. Do not discard them by releasing the product. If the most recently released version contains errors, then reverting copies the released version, with its errors, into the work space.

When you revert, the entire contents of the current work space is discarded. This includes the customizable product's design, user interface, rules, links, resources, and scripts. An instance of the most recently released version is then loaded into the current work space.

To revert to the most recently released version

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. Open the Lock/Unlock Product menu and choose Revert to Released.
4. Save the current work space.

Deleting a Customizable Product Version

You cannot delete a customizable product version. If you no longer want to use the current version, you can release a new version.

Copying a Customizable Product

When you copy a customizable product, all parts of the product are included in the copy. This includes its relationships, links, resources, scripts, rules, and its user interface.

All the parts of the copied product are visible and can be edited in the designers in the Customizable Product and Configuration Designer pages.

To copy a customizable product

1. Navigate to Product Administration.
2. Select the customizable product you want to copy.
3. Open the Products menu and choose Copy Record.

   The Products form appears.
4. Enter a name for the copy in the Product Field.
5. Revise other fields, such as Part # as desired.

   The copied product displays in the Products list.
6. Save the record.

Obtaining a Report on a Product's Structure

You can request a report that lists all the relationships in a customizable product as well as the contents of each relationship. You can request the report once, or schedule the report to run at scheduled times.

The report title is Product Relationship Report. This report must be enabled on the report server before performing the following procedure.

To obtain a report on a product's structure

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. Open the Customizable Products drop-down menu and select Product Designer.

   The Product Designer view appears.
4. Open the application View menu and click Reports.

   A form appears for selecting and running reports.
5. Verify the selected report is the Product Relationship Report.
6. Select the desired language and locale.
7. To run the report now, click Run Now.

   The report window appears, and you can view the report and print the report as desired.
8. To schedule the report to run at a scheduled time, click Schedule.

   A form appears for scheduling the report.

Creating Class-Product Templates

You can create a customizable product and use it as a template for building other customizable products. These templates are called class-products. Use this feature when you have customizable products that include the same group of items. For example, you sell desktop computers. You have seven configurable models that share the same chassis types, keyboards, and mouse. You can create a class-product consisting of these three relationships. You would then use the class-product as the basis for constructing each model.

When you create a customizable product and designate it as a class-product, all products belonging to the same product class and all subclasses inherit its structure. In effect, the class-product becomes an extension of the class definition, and the class-product's structure is inherited in the same fashion as class attributes.

In Figure 8, Template Customizable Product A has been designated as a class-product in Product Class 1. Customizable Product B is in the same product class, so it inherits the structure of Template Customizable Product A. Customizable Product C is in Subclass 2, a subclass of Product Class 1. It also inherits the structure of Template Customizable Product A.

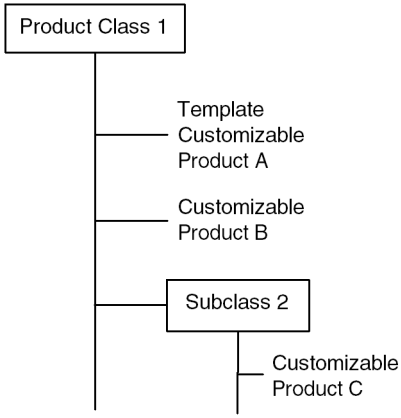

Figure 8. Customizable Product Template

TIP: Assign a customizable product to a class containing a class-product template right after you create its work space. You can then view what parts the class-product contributes. This will help you avoid creating duplicate relationships in the other customizable products assigned to the class.

The following parts of a customizable product are inherited by all class members when you designate it as a class-product:

- Relationships and their contents.
- Configuration rules.
- Resources
- Links
- User interface groups.

- The base theme and product theme are not inherited
- User interface property definitions and scripts are not inherited by class members

When you edit a class-product, the changes propagate to other products in the class when you commit the changes.

When you edit a customizable product that has inherited the structure of a class-product, the class-product portion is viewable in the Product UI Designer, the User Interface Property Designer, and in validation mode. The class-product portion is not visible in the Product Designer, Resource Designer, Link Designer, or Script Designer.

Observe the following guidelines when creating class-products:

- Customizable products that you designate as class-products must not be orderable.
- Assign only customizable products to a class containing a class-product. Assigning simple products to a class containing a class-product can cause unexpected results in the user interface.
- A customizable product must have at least one released version before it can be designated a class-product.

To create a class-product template

1. Navigate to Product Administration.
2. In Products, select the customizable product you want to use as a template.
3. If the customizable product has a check mark in the Orderable field, click in this field to remove the check mark.
4. In Customizable Product, lock the customizable product.
5. Review the structure, user interface groups, resource definitions, link definitions, and scripts defined for the product. Remove any features that you do not want to propagate to other customizable products.
6. Refresh the customizable product's work space.

7. Create a product class for the class-product template as needed and assign the class-product to the class.
8. In the Products tab, select the customizable product then click Set as Class Product/Reset.

   Be careful to click this button only once. Clicking it a second time turns the class product template off.

   A check mark appears in the Class Product field.
9. Add the desired customizable products to the product class.

   Adding a product to the class causes it to inherit the structure of the class-product.
10. For the customizable products you add, assign the desired base and product theme in Customizable Product > Versions.
11. Open the Product UI Designer and verify the class-product structure has been inherited correctly.

Turning Off a Class-Product Template

When you turn off a class-product template, all its inherited features are removed from the members of its product class. The class-product's structure no longer appears in the Product UI Designer or in validation mode for members of the class. Turning off a class-product template is very similar in its effect to deleting an attribute at the class level.

UI groups are not affected when you turn off a class-product template. For members of the class, if you have added items to a UI group from a class-product template, you must manually remove them.

To turn off a class-product template

1. Navigate to Product Administration.
2. Select the desire class-product template.

   Verify that a check mark displays in the Class Product field.
3. Click the Set as Class Product/Reset button.

   Verify that no check mark displays in the Class Product field.
4. For every product in the same product class, open the Product UI designer and remove all the class-product template items from the UI groups.

Converting a Bundle to a Regular Customizable Product

A bundle is a group of items sold as one product and is a special form of customizable product. A bundle has one or more relationships that have a Product domain. Bundles do not include a UI definition, configuration rules, links, resources, or scripts.

When you convert a bundle to a regular customizable product, you can work with the newly converted customizable product in the same way as a regular customizable product.

To convert a bundle to a regular customizable product

1. Navigate to Product Administration and query for the desired bundle.
2. Navigate to Customizable Product > Product Versions.
3. In Lock/Unlock, click in the Locked Flag field.

   A check mark appears in the field.
4. Click Release New Version.

   This converts the bundle to a regular customizable product. A check mark displays in the Customizable Product check box in the Product record. A check mark no longer displays in the Bundle check box.
5. Revise existing quotes and orders as needed to reflect the change.

Converting a Regular Customizable Product to a Bundle

When you convert a regular customizable product to a bundle, any items outside the scope of a bundle, such as configuration rules, stop being effective and are ignored. They are not erased from the definition and become effective again if you convert the bundle back to a customizable product.

Converting has the following effects:

- All previous versions still display in Customizable Products > Versions.
- The current work space is retained, but its contents are altered as described in the following items.
- For relationships that have a Product domain, the product is added to the bundle.
- For relationships that have a Class or Dynamic Class domain, only the product specified in the Default Product field is added to the bundle. If no product is specified, no product is added, even if the Default Cardinality is greater than one.
- The quantity in the Default Cardinality field is used to determine the quantity of the product in the bundle. Other cardinality fields are ignored. If the default cardinality is blank or zero, the quantity of the product in the bundle is blank. When creating a quote, only those products with a quantity greater than or equal to one are displayed in the quote.
- All selection page definitions, UI Property definitions, configuration rules, link definitions, resource definitions, and scripts are ignored and do not become part of the bundle. If you convert the bundle back to a regular customizable product, these again become effective.
- When you convert a customizable product that inherits part of its structure from a class-product, none of the inherited structure becomes part of the bundle.
- You cannot designate a bundle as a class-product.
- If the Forecastable flag is set for a product in the Product Designer and the product becomes part of a bundle, the Forecastable flag remains set for the product in the bundle.

When a user adds a bundle to a quote or order, the bundle and its components display as separate line items. Users cannot start a configuration session.

The customizable product must have at least one released version before you can convert it to a bundle.

To convert a regular customizable product to a bundle

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. Make any desired changes and release a new version.
4. Navigate to Bundle Administration and review the displayed contents.

   Bundle Administration displays the products in the customizable product that will be included in the bundle. If the contents are not correct, revise the customizable product and release a new version.
5. In Bundle Administration, click Modify, then click Done.

   This converts the customizable product to a bundle and releases a new version. A check mark displays in the Bundle field in the product form. The check mark in the Customizable Product field is removed.
6. Revise existing quotes and orders as needed to reflect the change.

Defining a Customizable Asset

You define a customizable asset in Assets > Customizable Assets. This adds the customizable asset to the Customizable Asset dialog box in Quotes. Defining a customizable asset requires two steps:

a Create a customizable asset record.

b Configure the customizable asset.

Once you have defined a customizable asset, you can select it when creating delta quotes.

To create a customizable asset record

1 Navigate to Assets.

2 Add a new record.

The system assigns an asset number and displays it in the Asset # field.

3 Click in the Product field and display the Pick Product Form.

4 In the Pick Product Form, put an X in the Customizable box and click Go.

The Pick Product dialog box displays all the customizable products in the product table.

5 Query for the customizable product on which the customer's customizable asset is based and click OK.

The customizable product name displays in the Product field.

6 In the Assets form Account field click the select button to display the Pick Account dialog box. Query for the desired account and click OK.

This field filters the records displayed in the Customizable Asset dialog box in Quotes. The dialog box displays only the customizable assets that have the same account name as the account name in the Quotes record. If you do not assign an account name, the customizable asset displays every time the Customizable Asset dialog box is opened, regardless of the account.

7 In the Asset Description field, enter a descriptive phrase or name that is meaningful to users creating quotes.

   This field displays in the Customizable Asset dialog box in quotes.

8 Fill out the remaining fields in the Asset record as needed.

   For example, enter a product installed date and status.

The next step is to configure the root customizable product so that it has the same configuration as the product the customer has purchased.

To configure a customizable asset

1 Review the customizable product configuration that the customer has purchased.

   Determine if a new version of the customizable product has been released and what effect this will have on configuring a customizable asset.

2 Highlight the customizable asset record for which you want to create a configuration.

3 Click the More Info tab to display the asset form and then click Customize.

   This starts a configuration session, similar to those users see when configuring customizable products in Quotes. The session displays the selection pages for the customizable product that the user purchased.

4 Configure the customizable product to reflect what the customer purchased and exit the session. This includes configuring component attributes.

   This creates the configured customizable asset.

5 In the Assets list, verify that the desired customizable asset is highlighted.

6 Click the Attribute tab and set the value of attributes defined for the customizable asset as a whole and then click Save.

Controlling How Products and Bundles Are Taxed

The components of customizable products and bundles can have different tax rates.

For example, a company may sell a customizable product called Concrete Services. As its components, this product may have the cost of concrete, the cost of using a truck to pour concrete, the cost of labor, and the cost of engineering services. In some jurisdictions, these components may be taxed at different rates.

You can control how tax is computed by setting the Tax Subcomponent flag. To tax the components individually, set the Tax Subcomponent flag on the root of the customizable product or bundle. If you do not set the Tax Subcomponent flag, the tax is computed on the total price of the customizable product or bundle.

If one of the components is itself a customizable product, you can set the Tax Subcomponent flag on the component. This causes the tax for that component to be the sum of the tax computations on the components of that component.

Note the following points about taxing components:

- You cannot use a base price for the parent product if the components are taxed individually. The parent product price must be the sum of the prices of all the component products. You will get inaccurate results if you give the parent product a price, and then do delta pricing on components and compute tax at the subcomponent level.
- When you set the Tax Subcomponent flag for the parent product, you can either set this flag or not set it for each component product that has subcomponents. If you don't set this flag on a component, the tax will be calculated for the component product. If you do set the flag on a component, the tax for the component will be based on the tax of its components.
- When you set the Tax Subcomponent flag for a component of a product, you must set this flag for the parent product as well. If you do not, you will tax the entire product at the parent level, and you will also tax the component product. This double counting will cause an inaccurate tax calculation.

The tax will be calculated accurately if you apply volume discounts or bundling discounts to the product. These discounts are applied at the component level, so the tax on each subcomponent will be adjusted to reflect the discount.

CAUTION: The tax will not be calculated accurately if you use a single-factor pricing model, which applies the discount to the total value of a customizable product, not to each component. You also cannot use simple bundles for pricing when tax rates of components are different.

To tax the components of a customizable product or bundle

1. Navigate to the Product Administration screen.
2. In the Products list, select the desired customizable product or bundle.
3. Expand the product form and put a check mark in the Tax Subcomponent Flag check box.

Controlling How Customizable Products are Forecast

When you add a component to a customizable product in the Product Designer, a check mark displays in the Forecastable field. This means the component is added to product forecasts when the customizable product is included in a quote and the user updates the related opportunity.

To prevent components from being added to product forecasts, remove the check mark from the component's Forecastable field in the Product Designer.

A Forecastable check box is also available in Quotes > Line Items. This allows you to add or remove a product or component from product forecasts within individual quotes.

Customizable Product User Interface 12

This chapter describes how to use the Product UI Designer to create a user interface for configuring a customizable product. You must create a work space for a product before you can use the Product UI Designer.

Understanding the Role of the Product UI Designer

Using the Product UI Designer, you can define the page or series of pages that display during a configuration session. These pages display when the user configures a customizable product as part of creating a quote in the Quotes screen. They also display when a user selects a customizable product in an eSales Web page. The same pages display when an administrator validates a customizable product.

The pages that display during the configuration process are called selection pages. The user makes selections from these pages to configure the customizable product.

The interface you design is stored with the customizable product. Use the following process to design the selection pages for a customizable product:

a **Choosing a basic page layout.** You set the basic look and feel of the customizable product's selection pages by choosing a base theme in the Versions tab. Several base themes are provided. You can also build your own base themes.

b **Choosing a method of presenting products.** There are several ways to present your products. You can present them all on one page, you can set up several tabbed pages, or you can set up a wizard to guide the user through pages sequentially. You select a product theme in the Versions tab. You can also build your own product themes.

c **Choosing what items a page will contain.** You can choose which items to display on a page using a grouping mechanism provided in the Product UI Designer. You can add to a group the items in a relationship, the attributes of the customizable product, its links, or its resources. Depending on the product theme, each group displays on a separate page.

d **Choosing how items will be selected.** When you add an item to a group in the Product UI Designer, you can choose among several user interface control types for it such as combo box, check box, radio button, quantity box, and text box. These UI control types determine how the user goes about selecting an item.

The user interface you design is part of the customizable product's current work space. When you release a customizable product, the user interface is stored as part of the released version.

Understanding Base Themes

When you create a work space for a customizable product, you can select a base theme and product theme. These control the basic look and feel of the pages a user sees when they configure the product.

A base theme defines the home page for the customizable product. It is the container within which the product theme pages display. The base theme has a thread bar you can use to navigate between product theme pages. Two types of base theme are provided:

- **Base Theme with Auto Pricing**. When the user selects an item, its price is updated immediately.
- **Base Theme with Manual Pricing**. The pricing of items is not updated until the user clicks Update Price.

Upgrade users: If you do not select a base or product theme, default themes are used. If you do not specify groups or controls, intelligent defaults are used. These defaults replace Configuration Assistant in release 6.x and lower.

Understanding Product Themes

Product themes specify the style used to group items together on selection pages. You define which items appear on a page by defining groups in the Product UI Designer and adding products to the groups. Each group is displayed on a separate selection page.

Three basic product theme types are provided.

Tab Product Theme

The items in each group are displayed on a separate page, as shown in Figure 9. The Stereos group and the Options group each have a tab, and the user can move between pages by clicking the tab. A standard group of buttons, Save, Cancel, Done, and Finish It display above the tabs. A Red Star displays next to the title of the Stereos page to indicate that the user is required to choose an item and has not done so.

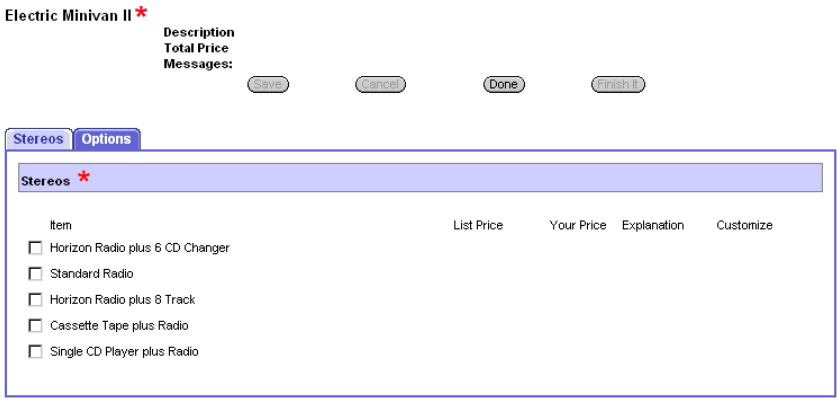

Figure 9. Tab Product Theme

Wizard Product Theme

The items in each group are presented on separate pages, as shown in Figure 10. The Stereos group is displayed by itself as the first page in the sequence. A standard group of buttons, Save, Cancel, Done, and Finish It display above the page. A Red Star displays next to the title of the Stereos page to indicate that the user is required to choose an item and has not done so. Previous and Next buttons, located within the page, allow the user to move between pages.

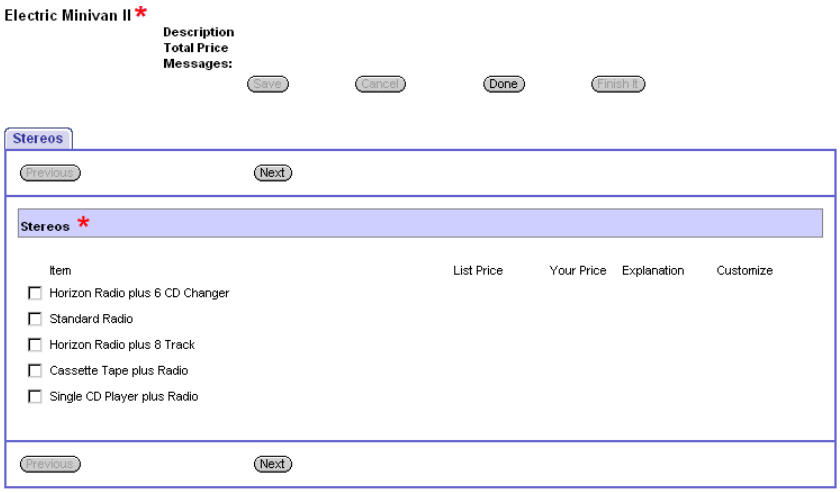

Figure 10. Wizard Product Theme

Single Page Product Theme

All the groups in the customizable product are presented on a single selection page, as shown in Figure 11. The tab pages used in the Tab theme and Wizard theme are stacked vertically one beneath the other to form the selection page. A standard group of buttons, Save, Cancel, Done, and Finish It display at the top of the selection page. A Red Star displays next to the title of the Stereos group to indicate that the user is required to choose an item and has not done so.

The top portion of the page, containing the product name and standard buttons, is not shown in the figure.

Figure 11. Single Page Product Theme

Understanding the Default User Interface

If you do not select a base theme or product theme, the system displays all the configurable items in the customizable product on a single selection page.

If you do not select controls, the system makes intelligent choices for UI controls based on attribute domain type and upon relationship cardinality.

For example, if an attribute has an LOV type domain, the system will display a combo box for setting the attribute. If the attribute has a range type domain, the system will display a text box for entering the range value.

If a relationship has a min and max cardinality of 1, the system will display a combo box without multiple instances. If the max cardinality is greater than 1, a combo box with multiple instances displays.

Understanding the Menu-Based Interface

A special set of themes and UI controls is provided to build a menu-based interface:

- **Base Theme with Menu**. This theme must be selected as the base theme. This theme provides auto-repricing. No base theme is provided to select manual repricing.
- **Menu Product Theme**. This theme must be selected as the product theme. It displays relationships and their contents. It does not displays attributes, resources, or linked items.
- **Standard Menu Group Theme**. This theme must be selected for all groups except a summary page group.
- **Summary Menu Group Theme**. Assign this theme to a group when you want to display a summary page. For more information on summary pages, see "Adding a Summary Page" on page 239.
- **Check Box For Menu Theme**. Select this UI control to display a check box with price.

The menu-based interface displays each group as a menu item. When the user clicks on the group name, a pane opens and displays the items that can be selected in that group. When the user makes a selection, the selection is displayed below the group name. Figure 12 shows a menu-based selection page. Two groups have been defined, Stereos and Options. The user's selection is shown below each group.

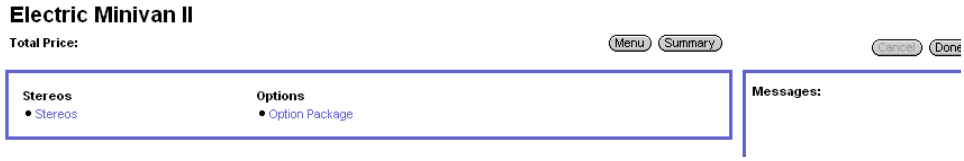

Figure 12. Example of Menu-Based Selection Page

Both group names are hyperlinks. If the user clicks on the Options group, a pane displaying the contents of that group opens as shown in Figure 13. The user can then select items from that group. When the user is finished, they click Menu to close the pane.

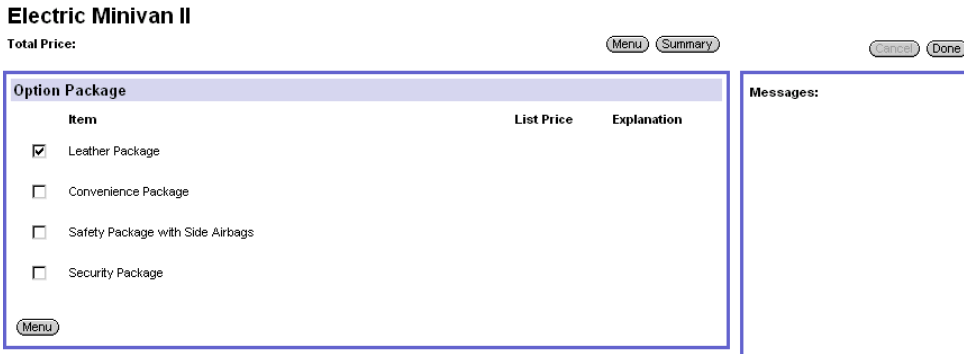

Figure 13. Option Package Group

Observe the following guidelines when using these themes and controls:

- The menu-based themes and controls can be used in conjunction with employee and partner applications. They cannot be used with customer applications, such as Siebel eSales.
- You cannot mix menu-based themes and controls with other types of themes and controls in a product. For example, if you select a menu-based base theme, you must also select a menu-based product theme as well as menu-based UI controls.
- A Customizable product that will be a component of another customizable product cannot use menu-based themes and controls. For example CP2 is a customizable product and a component of product CP1. CP2 cannot use menu-based themes and controls. Note that CP1 is not restricted from using menu-based themes and controls.
- Only one summary group can be defined for products that use menu-based themes and controls. The summary group must be named: .
- Menu-based UI controls for links, resources, and attributes are not supported. Do not add links, resources, or attributes to groups.

Understanding Groups

Groups are the way you define what items appear on a selection page. Depending on the product theme, each group you define causes a separate selection page to be created. For example, you want all the hard disks in a customizable product to appear on the same page. You do this by defining a group in the Product UI Designer and then adding the Hard Disks relationship to this group. If you selected a tab type product theme, the hard disks display as a selectable page tab. When the user clicks on the tab, a selection page displays containing only the hard disks.

The Product UI Designer lets you create groups that contain relationships, attributes, resources, or links.

When you create a group, you can choose a group style. The group style defines the details of how a group will display.

A group definition contains the following fields:

- **Group Name**. Required. The name you give a group displays on the selection page. Use the product theme to guide how you choose group names. For example, if you use a tab theme, name the group according to what you want to appear on the page tabs.
- **Group Theme**. Required. The group theme lets you specify the layout of the group within the base theme. In most cases, you will select the Standard Group Theme.
- **Sequence**. The Sequence field governs the order in which pages display. The group with sequence = 1 displays first in the page series.
- **Description.** Use the description field to keep notes about the group definition. The contents of this field do not display on the selection page.

After defining a group, you select items from the customizable product to add to it. An item record for a group contains the following fields:

- **Name**. For Relationship items, this field contains the product display name. For Attribute, Resource and Linked Items, this field contains the item name. This field displays in the selection page.
- **Type**. This field displays whether the item is from a relationship, is a resource, a linked item, or is one of the customizable product's attributes.
- **Sequence**. The sequence controls the order of display of relationships within a group. The item with Sequence = 1 is displayed first in the group.
- **UI Control**. Click in this field to select a UI control, such as radio button or check box for the item or for the items in a relationship.

Understanding User Interface Controls

A user interface control determines how an object is displayed for selection. For example, a radio button control, displays a list of items with a button for each one. You choose a control as part of adding an item to a group in the Product UI Designer.

The Product UI Designer provides several types of UI controls:

- **Combo Box**. A combo box is a drop-down menu. Items are hidden from view until you click the down-arrow to open the menu and make selections. There are two types of combo box:
  - **Single selection**. The user can select only one value from the drop-down menu.
  - **Multiple selection**. The user can select multiple values from the drop-down menu using an Add Item button.
- **Check Box**. Items display as a list. Each item has a check box next to it. When you select an item, a check mark appears in the check box. You can select more than one item.
- **Radio Button**. Items display as a list. Each item has a button next to it. When you select an item, a dot appears in the button. You can select only one item. If you select another item, the previous item's button clears, and the current item displays a dot in the button.
- **Quantity Box**. A box displays next to the item in which the user enters or edits the quantity. The user then clicks elsewhere in the page to update the quantity.
- **Text Box**. A read-only box displays next to the item. The box contains the value of the displayed item. Use this UI control to display the current value of resources or linked items.
- **Edit Box**. A text box displays next to the item. The text box contains the value of the displayed item. You can edit the value. Use this control when you want users to be able to manually enter or edit attribute values.

Keep in mind the following factors when choosing a user interface control:

- For items added from a relationship, what are the cardinalities? If the minimum and maximum cardinality for the relationship is 1, this means only one item can be selected. The radio button or single-selection combo box can be used because it allows only one selection. If the relationship cardinality allows more than one selection, you must choose a UI control that allows multiple selections such as a check box or multiple-selection combo box.
- For attribute items, select a UI control that matches the attribute type. For LOV attributes use a combo box. For a range of values attribute or a single-value attribute, use a quantity box or text box.
- For resource items and for linked items, use a text box.

Table 19 on page 230 describes the specific UI controls available in the Product UI Designer. In the Multiple Items column, Yes means that the UI control allows selection of more than one item from a list. No means you can select only one item.

For all controls, excluded items display unavailable. For example, if you assign a radio button control to a relationship, excluded items display with the radio button grayed out so that it cannot be selected.

You can revise this so that excluded items do not display at all. You do this by assigning the predefined Excluded UI property to an item. You assign the Excluded UI property in the User Interface Property Designer.

Table 19. Product UI Designer UI Controls

UI Control Type	Select Multiple Items?	Description
Combo Box	No	Selected item highlighted.
Combo Box with Add Button	Yes	Selected item highlighted. Unselected items have Add button.
Combo Box with Price	No	Selected item highlighted. Displays price.
Combo Box with Price and Quantity	No	Selected item highlighted. Displays price. Allows user to enter quantity.

Table 19. Product UI Designer UI Controls

UI Control Type	Select Multiple Items?	Description
Combo Box with Add Button and Price	Yes	Selected item highlighted. Unselected items have Add button. Displays price.
Combo Box with Update Quantity button	No	Selected item highlighted. User can specify a quantity for the selected item.
Check Box with Price	Yes	Displays item price.
Check Box without Price	Yes	Price not displayed.
Radio Buttons with Price	No	Displays item price.
Radio Buttons without Price	No	Price not displayed.
Quantity Box	User enters quantity	Update Quantity button displays next to item name.
Quantity Box with Price	User enters quantity	Displays item price. Update Quantity button displays next to item name.
Text Box with Price	No	Read-only. Displays item price.
Text Box	No	Read-only.
Combo Box for Attribute	No	Selected item highlighted.
Edit Box for Attribute	No	Displays value of attribute. Can be edited.
Radio Button for Attribute	No	User can select one attribute value.
Text Box for Attribute	No	Read-only. Displays value of attribute.
Linked Item	No	Read-only. Displays value of linked item.
Resource	No	Read-only. Displays value of resource.

Understanding Pricing Integration

Siebel ePricer works in conjunction with Siebel eConfigurator to provide updated pricing information during a configuration session. There are two methods for obtaining updated pricing information: automatic and manual. You select which method to use when you select a base theme.

Automatic price updates is the default method and is the method used by the default base theme. Base themes that provide manual price updates are labeled as such when displayed in the dialog box where you select the base theme. Unless labeled otherwise, base themes use the automatic price update method.

With automatic price updates, the pricing of the entire customizable product is updated when the session starts, each time a new solution is created during the session, and when the session ends. When the user picks a product, the price of the product displays automatically.

With manual price updates, the pricing of the entire customizable product is updated when the session starts, when the user clicks the Check Price button, and when the session ends. During the session, prices are not updated automatically when the user picks a product. The user must click the Check Price button to obtain the prices of the products they select.

When a price update occurs during a configuration session ePricer pricing elements trigger in the following order:

a Component-based pricing adjustments

b Attribute-based pricing adjustments

c Customizable product pricing factors (single, bundle, matrix, and script-based only). Aggregate and volume discount pricing factors do not trigger.

d Price List pricing factors (single, bundle, matrix, and script-based only). Aggregate and volume discount pricing factors do not trigger.

When the session ends, ePricer pricing elements trigger in the same order. Volume discount and aggregate pricing factors also trigger for Quotes, Orders, and agreements.

If a customizable product contains other customizable products, only the pricing model for the parent customizable product is triggered during a price update. For example, customizable product CP1 contains customizable product CP2. During a configuration session for CP1, the pricing factor models for CP2 are not triggered during a price update.

When building a customizable product, use automatic price updates. Switch to manual price updates only if performance becomes too slow. The sequence of events after the user selects a product is as follows:

- eConfigurator engine computes a new solution
- eConfigurator engine forwards the solution to ePricer
- ePricer returns pricing for all items in the solution
- eConfigurator redisplays the selection page

If you do not need interactive pricing, consider switching to manual price updates.

Selecting the Base and Product Themes

User interface themes are templates that control the basic look and feel of the selection pages that users see when configuring a customizable product. Base themes control basic page design and product themes control the method used to present product choices.

Selecting the base and product themes

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. In Versions, click in the Base Theme field. Then click the select button to open the dialog box.
4. Select the desired base theme.

   The base theme appears in the Base Theme field.
5. Click in the Product Theme field. Then click the select button to open the dialog box.
6. Select the desired product theme.

   The Product theme appears in the Product Theme field
7. Save the record.

Grouping Items onto Pages

Groups are the way you define what items go on which selection pages. Depending on the product theme, all the items in a group display on one page. These pages display when the user selects the item for configuration.

For example, you design a customizable product that includes a computer monitor and a service plan. The user can select from among 4 monitors and 3 service plans. To display monitors and service plans on separate selection pages, you would create one group for monitors and one for service plans.

Setting up groups requires two steps:

a Create a group for each selection page.

b Add items to the groups.

All the relationships, resource definitions, and linked items in a customizable product are listed in the Product UI Designer. Each item has an Add Item to Group button.

The attributes of the customizable product are also listed. These are not the attributes defined for items in relationships. These are the attributes that the whole customizable product inherits from the class to which it belongs in the product table.

After you create a group and add items, you can verify your work by validating the customizable product. Validating a customizable product displays the selection pages a user sees during a configuration session. To validate the customizable product, open the Group Lists menu and choose Validate.

To create a group

1 Navigate to Product Administration.

2 Select and lock the desired customizable product.

3 Open the Customizable Product Show menu and choose Product UI Designer.

The Product UI Designer view appears.

4 Click New in the Group List tab to create a new group, fill out the group definition form, and click Save.

   The new group definition appears in the Product UI Designer.

5 Repeat these steps until you have created all the desired groups.

Each group definition corresponds to one selection page. After you define a group, add the items you want to display on its corresponding page.

To add items to a group

1 Select the desired group.

2 In the display of the customizable product, click the desired item.

3 Click Add Item.

   The item appears in the Group Item list, which lists the members of the group.

4 Enter a positive whole number in the Sequence field.

   This controls the order of display on the page. Item 1 displays first and so on.

5 Click the select button in the UI Control field.

   The Pick UI Control dialog box appears.

6 Select the desired UI control type.

   The UI control type appears in the UI Control field in the Group Item List.

7 Click Save to save the record.

   Records are not saved automatically when you step off the record.

8 Repeat the steps above for each item you want to add to the group.

Editing Item Groups

You edit item groups by selecting a group and then editing the group record or by editing items in the group's Group Item List.

In the group record, you can change the group theme or the order in which the group displays.

In the Group Item List for a group, you can edit records by changing their sequence of display or by changing the type of UI control used to display the item.

If you change the UI control for a relationship, make sure that the new control allows the user to exercise the full range of the cardinalities you have defined for the relationship.

After you edit a group, you can verify your work by validating the customizable product. Validating a customizable product displays the selection pages a user sees during a configuration session. To validate the customizable product, open the Group Lists menu and choose Validate.

To edit an item group

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. Open the Customizable Product Show menu and choose Product UI Designer.

   The Product UI Designer view appears.
4. In the Group List, select the group you want to edit.
5. To change the group theme, click the group's Group Theme select button and choose a new group theme.
6. To change the group's sequence of display, edit the Sequence field.

   The group with Sequence = 1 displays first. Changing the group sequence changes the order in which the selection pages display.
7. To change the sequence of display of relationships in the group, select an item in Group Item List and edit the Sequence field.

   The item with Sequence = 1 displays first. Changing the sequence changes the order in which the items display on the page.

8 To change the display control associated with an item, select an item, click the UI control select button, and select a new control.

9 Click Save to save the record.

Records are not saved automatically when you step off them.

Deleting Item Groups

Item groups are the mechanism you use to group items onto selection pages. Depending on the product theme, each item group displays on a separate selection page. Deleting a group removes the page and all its items from the collection of pages used to configure a customizable product.

After you delete a group, you can verify your work by validating the customizable product. Validating a customizable product displays the selection pages a user sees during a configuration session. To validate the customizable product, open the Group Lists menu and choose Validate.

Deleting an item group

1 Navigate to Product Administration.

2 Select and lock the desired customizable product.

3 Open the Customizable Product Show menu and choose Product UI Designer.

The Product UI Designer view appears.

4 In the Group List, select the desired group.

5 Open the Group List menu and choose Delete Record.

6 Click OK when asked to confirm you want to delete the record.

Adding a Summary Page

You can add a summary page that shows the user how they have configured a product. This page displays the relationships from which the user has made selections along with attribute values. It also displays all the items the user has chosen.

Each relationship is a hyperlink. When the user clicks on the relationship name, the selection page containing that relationship displays, and the user can revise their selections.

Figure 14 shows an example of a summary page. The top portion of the page lists the relationships and the attribute values that have been selected. The bottom portion of the page shows the items that the user has selected from each relationship.

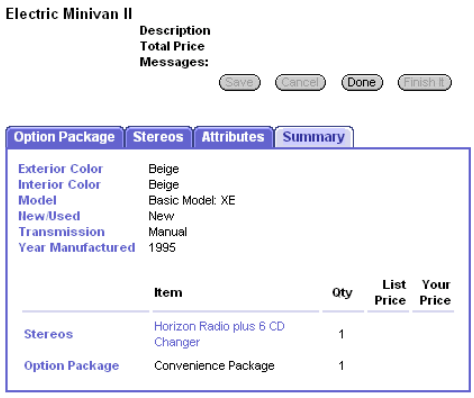

Figure 14. Example of a Summary Page

After you create a summary page, you can verify your work by validating the customizable product. Validating a customizable product displays the selection pages a user sees during a configuration session. To validate the customizable product, open the Group Lists menu and choose Validate.

To add a summary page

1. In the product UI designer, create a group.
2. In the Group Name field, enter the name that will be the tab or page title name.

   For example, enter Summary.
3. In the Group Theme field, click the select button and choose the Summary Group Theme.

   If you are using the Menu templates, choose the Summary Menu Group Theme.
4. In the Sequence field, enter a number to set the sequence of the summary page.

   To make the summary page the last page in the series, enter a sequence number that is higher than the other groups.
5. Click Save to save the record.

   Records are not saved automatically when you step off the record. The system generates the summary page automatically. Do not add records to the group.

Customizable Product UI Properties 13

This chapter explains how to use the User Interface Property Designer to modify the display of selection pages for customizable products. You must define a work space and have used the Product UI Designer to define selection-page layout before using the User Interface Property Designer.

Understanding UI Properties

A UI property is a named variable and its value. UI properties modify the display of an item in a customizable product. You define a UI property by selecting the desired item in a customizable product and then defining one or more UI properties for it.

There are two types of UI properties: predefined and user-defined. Predefined UI properties, such as Hide, are Siebel-provided UI properties that perform special functions. User-defined UI properties are those that you define and then insert into a Web template to control the display characteristics of an item.

The User Interface Property Designer displays all the items in the customizable product:

- **Customizable Product Name**. Define a UI property for the product name to change the product name header on every selection page.
- **Attribute**. The Attribute section displays the attributes that the customizable product as a whole inherits from the class to which it belongs. For LOV type attributes, each menu choice is listed. For range of values attributes, the default value is listed.
- **Relationship**. This sections displays both the relationship name and the contents of each relationship in the customizable product.
- **Group**. The Group section lists the group names you defined in the Product UI Designer.
- **Linked Item and Resource**. These sections display the linked items and resources you have defined for the customizable product.

To use the User Interface Property Designer, you select an item in the customizable product and then define a UI property for the item in Item Display Properties. Each record in Display Properties has two fields. The first contains the name of the UI Property variable. The second field contains the value you want to assign to the variable. This value is what displays in the configuration session.

Observe the following guidelines when defining UI properties:

- The relationship between an item and a UI variable name is one-to-one. You cannot define multiple UI properties for an item, each with the same variable name.
- An item can have multiple UI property definitions, each for a different UI variable. An example of this would be an item that appears in multiple locations within a customizable product.

Understanding Predefined UI Properties

Table 20 shows the predefined UI properties that you can use. These UI properties provide commonly desired ways to modify the display of items. You do not need to insert a variable name for these properties into a customizable product Web template. You only need to assign them to the desired item in the User Interface Property Designer.

For external images, the image must be stored in the Siebel installation directory in \Public\enu\Images\ < filename > , where < filename > is the name of the file.

Table 20. Predefined UI Properties

Property Name	Value	Description
Excluded	Y	Can only be defined on a relationship or on the product root. Cannot be defined on products within a relationship When defined on a relationship, prevents all excluded items in the relationship from displaying. If an item in a relationship is a customizable product, does not prevent display of excluded products within that customizable product. When defined on the product root, prevents excluded items from displaying throughout the product.
Hide	Y or N	When set to Y, causes item to be omitted from selection pages. Can be defined on any part of a customizable product that displays in the UI Properties Designer.
Description	Enter a text string.	Define on relationships only. Enter the text exactly as it will display to the user.
Image	Images/ < filename >	Define on relationships only. The image displays on the right side of relationship header. The image is displayed full size.

Table 20. Predefined UI Properties

Property Name	Value	Description
LearnMore	Enter the full URL to the desired location	Use with relationships only. Do not use with component products, resources, attributes, or links. The words “Learn More” are displayed adjacent to the item and are a hyperlink to the URL you enter.
ProductHeaderImage	Images/ < filename >	Define on product root only. Displays an image of the root product on every selection page. Image displays beneath item header, to the left of item labels. The default image area is 120x120 pixels square. Can only be defined on product root.
FullComputation	Y	Define on attributes with LOV domains only. When user makes an attribute selection, eConfigurator engine updates the selection state of all the attribute values so that only selectable values are displayed. For example, if one of the values is excluded, it displays uavailable. Can cause performance reduction. Default behavior: the eConfigurator engine does not update the selection state of the other attribute values and displays all the values as selectable. For example, if one of the values is excluded, it does not display unavailable. If user selects an excluded value, they receive a conflict message. Use this UI property when display of the selection state of attribute values is required.

Defining a UI Property

A UI property is a named variable and its value. UI properties modify the display of an item in a customizable product. You define a UI property by selecting the desired item in a customizable product and then defining one or more UI properties for it.

There are two types of UI properties: predefined and user-defined. Predefined UI properties, such as Hide, are Siebel-provided UI properties that perform special functions. User-defined UI properties are those that you define and then insert into a Web template to control the display characteristics of an item.

After you define a UI property, you can verify your work by validating the customizable product. Validating a customizable product displays the selection pages a user sees during a configuration session. To validate the customizable product, open the Item Display Properties menu and choose Validate.

To define a UI property for a customizable product

1. Select and lock the desired customizable product.
2. Open the Customizable Product Show menu and choose User Interface Property Designer.

   The User Interface Property Designer view appears.
3. In the box displaying the contents of the customizable product, click the name of the item for which you want to define a UI property.
4. In Item Display Properties, click New.

   A new record appears.
5. Fill out the record.

   - **Name**. Enter the name of the UI property variable. The variable can be predefined or user-defined.
   - **Value**. Enter a value for the variable. The value can be predefined or user-defined.
6. Click Save to save the new record.

   Records are not saved automatically when you step off the record.

Hiding Parts of a Customizable Product

The predefined UI property Hide lets you omit items from display in selection pages. This UI property is very useful with class-products. For child products in the class, you can hide portions of the product structure inherited from the class-product. This allows you to define the class-product structure and then tailor its display for each of the child products that inherit the structure. You can hide any of the items that display in User Interface Property Designer:

- An attribute
- An attribute value
- A relationship
- An item in a relationship
- A group
- A linked item
- A resource

You define the Hide property on an item in the same fashion as other UI properties.

Customizable Product Web Templates 14

This chapter explains how to customize the Web templates that are used to create customizable product selection pages.

Understanding Customizable Product Web Templates

Customizable Product Web templates are files that control all aspects of how selection pages display. You customize the look and feel of selection pages by modifying these templates.

Customizable Product Web templates are stored in the installation subdirectory \webtempl. The templates that control display of selection pages begin with eCfg. There four types of Web templates used by eConfigurator.

- **Base Theme**. The base theme defines an HTML table-based page layout. This layout is used for all selection pages and provides the basic look and feel. The base theme calls the required product theme, which displays in one or more table cells created by the base theme.
- **Product Theme**. The product theme template defines how selection items and options are displayed. The one-page theme displays all the configurable items on one selection page. The tab themes display items on a series of tabbed pages. The wizard theme leads the user from one selection page to the next. Product themes also create table layouts that define how items display within the cells of the base theme. Product themes contain for-each loops that iterate through the customizable product and identify relationships, items, attributes, and so on. Product themes call group themes and control themes.
- **Group Theme**. If you use a tab or wizard theme, you determine what items appear on a page by defining groups. All the items in one group display on one page. A group theme template specifies how a group displays on a page.

- **UI control template**. UI control templates define what type of UI control is used for selecting items. You can choose from several types of check box, radio button, and text box controls. The control template iterates through each group, identifies all the items, and then creates a form that displays the items for selection. The forms display in the table cells created by the product theme.

You select a base theme and product theme template in the Versions tab. You select group themes and UI control templates in the Product UI Designer.

Web templates are not themselves HTML files, but they do contain a combination of HTML table commands, JavaScript, and Siebel Web Engine (swe) commands. The swe commands are in XML format. These commands are used in Web templates as follows:

- **HTML table commands**. These commands are used to define page layout. Control templates create tables and then create forms that display in these tables. Control template tables display in the cells created by product theme templates. Product theme templates displays in cells created by the base theme template.
- **JavaScript**. JavaScript commands are used to create arrays for storing data about the customizable product obtained from the UI service. They are also used in control templates to create forms.
- **swe:include**. This command specifies the name of a template to include. This command links the Web templates together. During iterative processing this command causes the Web Engine to dynamically retrieve, insert, and parse the included Web template.
- **swe:for-each**. This command provides iterative processing. It traverses the customizable product beginning at the product root and identifies specified parts for display. The CfgLoop Type specifies the type of items to be identified, such as group, relationship, or attribute.
- **swe:control**. This XML element defines what control type to display with an item. To define a UI property, you substitute a variable name for the value of the CfgFieldName attribute in this element. Inserting UI property variables in Web templates is how you customize the way an item displays.

Web templates are used by the Frame Code Engine to create HTML selection pages, as shown in Figure 15.

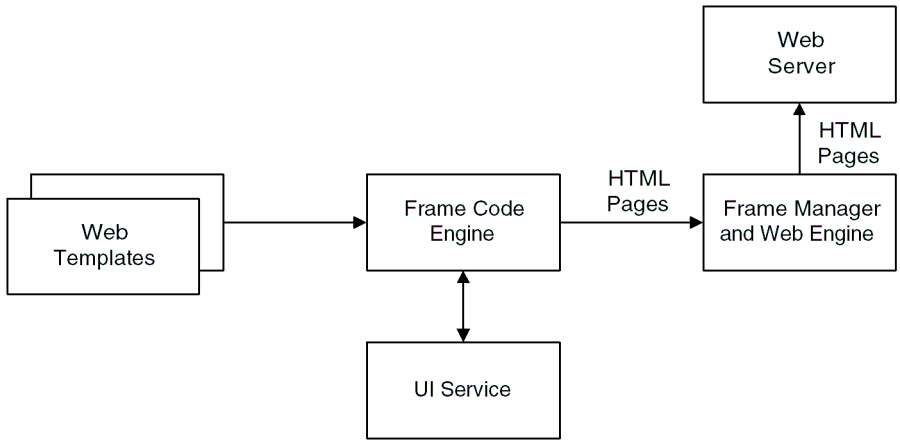

Figure 15. Web Template Processing

When a customizable product is called for display, an instance is created and stored in memory. The Frame Code Engine requests information about the product from the UI service and uses it to provide the values required in the Web templates. The Frame Code Engine then builds selection pages and forwards them to the Web Engine Frame Manager. The pages are then provided to the Web Server.

Understanding UI Properties in Web Templates

The User Interface Property Designer lets you customize the way items display in a configuration session. You do this by defining a UI property for the desired items. The UI property definition is a name-value pair where the variable name is one you have entered in the Web template that controls the display of the item. The value can be a string, HTML commands, XML commands, or JavaScript that defines what you want to display instead of the default item name.

The User Interface Property Designer displays all the items in the customizable product. Define UI properties on these items as follows to change how Web templates display:

- **Customizable Product Name**. Define a UI property for the product name to change the product name header on every selection page.
- **Attribute**. Define a UI property for the attribute name to revise the title of the attribute. To revise the display of the attribute values, define a UI property for the LOV items or default value of the attribute.
- **Relationship**. Like attributes, you can change either how the relationship name displays, or you can change how the items in the relationship display.
- **Group**. Define a UI property for these items when you want to change the name that appears on the product tabs or in the wizard pages of the product theme.
- **Linked Item and Resource**. If you display these items during a configuration session, you can customize them by defining a UI property for them. For example, consider adding a brief explanation along with a linked item name to explain what it does.

The Web Engine uses for-each loops to iterate through each level of a customizable product. At each level, it determines what items occupy that level and what Web template to use for displaying the items. When you modify a Web template and assign it to group member, the template is used to control the display of all the group member's items. For example, if you assign a radio button control template to a relationship containing five items. The control template is used to define how each of the five items displays. Thus, if you have inserted a UI property variable in the radio button control template, then you must define a UI property for all five items.

For example, you have assigned a radio button control to Relationship A in the Product UI Designer. You have inserted the UI property variable ".DisplayChange" into the radio button template. Relationship A contains five items. You want to change the display of Item A to bold. You must define a UI property for all five items:

- The Name in all five UI property definitions is the same: DisplayChange. Do not put a period (.) before the name.
- For Item A the Value of the UI property is < b > Item A < /b >
- For the remaining four items, the Value of the UI property is the item name, without any HTML formatting; for example, Item B.

Understanding UI Property Values

The value you assign to a UI property name for an item can be text, HTML commands, or JavaScript commands. If the value includes HTML or JavaScript commands, it is important to test them for correctness before entering them in the User Interface Property Designer.

If you do not test the commands and they have errors, this can prevent display of the selection pages. If the value is a text string that does not include commands, you do not need to test it.

You test the commands included in the value of the UI property name by inserting them in an HTML file and checking that they display correctly in a Web browser. Observe the following guidelines for including HTML commands or JavaScript in a UI property name value:

- Avoid using tags or tag attributes common only to Internet Explorer or to Netscape.
- Use DHTML commands with caution. Thoroughly test them before using them as the value of a UI property name.
- HTML statements should be self-contained and complete. Use opening and closing tags.
- Use table tags very carefully. Make sure the table you define is sized correctly for the space it will occupy.
- If you insert JavaScript using the < Script > tag, avoid statements that manipulate the document. Also avoid routines that rely on specific page content. If the content is not present, the script may fail and the page may not display.
- Do not use animated images or animated text.

HTML Text Formatting Commands

You can use HTML text formatting commands to enhance the way an item name displays. Here are several examples:

- You can define a UI property value that adds formatting to the item name. For example, you want the item name Lamp to display in boldface. You would assign the following UI property value to the item Lamp: < b > Lamp < /b > .

- You can add a message next to an item. If the message is lengthy consider creating a small, two-cell table. Put the item name in the first cell, and put the explanation in the adjacent cell. The value of the UI property name for the item would then be the HTML table commands, including the item name and message. The base theme and product theme Web templates use tables to layout the Web pages. This means the table you create for the item will be located within a cell of the table that contains the whole Web page. Carefully review the table structure of the base theme and product theme Web templates before creating tables for UI properties.

The following HTML tag types can be used as values for UI property names:

- Text markup tags (< b > , < em > , and so on)
- Table tags
- Content presentation and flow tags (< address > , < nobr > , < plaintext > , and so on)
- Formatted list tags
- Rule, image, and multimedia tags (< img > , < map > , < marquee >)
- Forms tags (< button > , < input type > and so on.). You can use these tags to pass user input to JavaScript routines that are part of the UI property name value.
- Hyperlinks. You must include Target = "" in the link tag (< a >) definition. This causes the link to load in a new browser window. If the link loads in the session browser window, the user will have to click the Back button to return to the session. This can cause the session to lose its context and can cause Web Engine problems.

Do not use the following tag types in UI property name values:

- Header tags (< base > , < basefont > , and so on)
- Skeletal/Layout tags (< frameset > , < body > , and so on)

Images

Use the HTML < img > tag as the UI property value when you want to retrieve and display images. You can shorten the path specification for the src attribute by storing the images in the same directory as other images used by the Web Engine.

The Web Engine stores its images in the following installation subdirectory (Windows path syntax):

```
PUBLIC\<language>\IMAGES
```

The < language > variable is the three-letter language identifier for the language selected during installation. For example, if you selected English during the install, the Web Engine image files are located in the PUBLIC\enu\IMAGES subdirectory.

When you specify the src path in the < img > tag, you only need to specify the IMAGES directory and the file name. For example, you want to retrieve red.gif from the IMAGES directory and use it to replace the attribute name Red. In the User Interface Property Designer, you would assign a UI property name to the Red attribute and specify the following value (Windows path syntax):

```
<img src=”IMAGES\red.gif”>
```

Before validating the UI design, you should test this value to make sure it behaves as expected in the browser. Here is an example of an HTML file for testing image retrieval (English language installation, Windows path syntax):

```
<html>
<head>
<base href="C:\installdir\PUBLIC\enu\">
</head>
<body>
<img src="IMAGES\red.gif">
</body>
</html>
```

Add HEIGHT and WIDTH attributes to the < IMG > tag to make the image the correct size. Consider making the image somewhat smaller than needed and then increasing its size when you validate the UI design. This prevents the image from causing page layout problems when you first validate it.

Creating a New Web Template

The most common reason for creating a new Web Template is to insert UI property variables in the template to change how an item in a customizable product displays. The best way to create a new Web template is to modify an existing template and save it to a new file name.

The process for setting up a new template has two steps:

a Create a new Web template.

b Add the new template to the appropriate dialog box.

To create a new Web template

1 Navigate to Product Administration.

2 Select and lock the desired customizable product.

3 Open the desired dialog box and locate the desired template.

For example, if you wanted to create a new base theme template, navigate to Customizable Products > Product Versions and, click the select button in the Base Theme field to open the dialog box.

4 In the dialog box, write down the filename of the template you want to modify.

The filename is shown in the Template field. The filename ends in .swt.

5 Open the Web template in a text editor, modify it as desired, and save it to a new filename.

The Web templates are located in <install-dir>\webtempl (Windows path syntax), where <install-dir> is the path to the Siebel installation directory. The new filename must end in .swt. By convention, the filename should begin with "eCfg."

In most cases, you will insert a UI property variable name into the template.

The next step is to add the new template to the dialog box.

To add a Web template to the dialog box

1 Open the dialog box again.

This is the dialog box you used to determine the filename of the template.

2 In the dialog box, click New and fill out the form for adding a new template and click OK.

- **Name**: Enter a descriptive name. For example: Modified Base Theme.
- **Template**: Enter the filename of the template you created.
- **Description (Optional)**: Enter a brief description of the template.

This adds the new template to the dialog box. You can now assign the new template to an item in the customizable product. For example, if you created a new base theme template, you can assign this template to the customizable product.

Modifying the Display Name of the Customizable Product

This section explains how to modify the customizable product name in the base theme. The base theme defines the basic page container within which the product theme displays. The base theme includes a header that contains the customizable product name that is stored in the Siebel database. You can define a UI property that changes the name of the customizable product or adds artwork or other formatting.

To change the name of the customizable product in a base theme, you insert a variable in the base theme Web template. This variable tells the Web Engine to get the item name from a defined UI property rather than using the customizable product name.

Here is the process for changing the display name of the customizable product:

1. Create a new base theme template. You do this by saving a copy of the base theme template. Then you insert a variable name in the new template.
2. Assign the new base theme template to the customizable product.
3. Define a UI property for the customizable product. This value is what displays in the selection pages.

Create a New Base Theme Template

You must create a new Web template to customize the display name of the customizable product. You do this by copying an existing base theme template. Then you insert a UI property variable into the copy. Finally, you assign the new Web template as the base theme for the customizable product

To insert a UI property variable into a base theme Web template, you must locate the swe:control element that governs the display of the customizable product name. The base themes have the same basic layout:

- The first < Table > tag creates the page container. Additional < Table > tags stack one on top of another to partition the page vertically.
- There are no swe:for-each loops in the template.
- Near the top of the file, refer to the first < table > tag. Within the table definition, locate the "Product Title" comment. Below the comment locate the first swe:control element.
- One of the attributes of this element is CfgFieldName = "CxObjName".
- Replace CxObjName with the name of the UI Property variable. The name must be preceded with a period (.). For example: ".NewProductName".

To create a new base theme template

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. In the Work Space record, click the select button in the Base Theme field and open the dialog box.
4. In the Pick UI Style dialog box, write down the filename of the base theme.

   The filename is shown in the Template field. The filename ends in .swt.
5. Open the Web template in a text editor and save it to a new filename.

   The Web templates are located in <install-dir>\webtempl (NT path syntax), where <install-dir> is the path to the Siebel installation directory. The new filename must end in .swt.
6. In the new template, locate the swe.control element containing the CfgFieldName = "CxObjName" attribute.
7. Replace "CxObjName" with a UI property variable name and save the file.

   The variable name must begin with a period (.). For example: ".NewProductName". Verify that the variable name is enclosed in quotes.

Assign the New Base Theme Template

To assign the new base theme template to the customizable product, you first add the template to the Pick UI Style dialog box. Then you select it as the base theme.

To assign the new base theme template to the customizable product

1. In the customizable product Work Space record, click the select button in the Base Theme field and open the dialog box.

   The Pick UI Style dialog box appears.

2. In the dialog box, click New and fill out the form for adding a new template and click OK.

   - **Name**: Enter a descriptive name. For example: Modified Base Theme.
   - **Template**: Enter the filename of the template you created.
   - **Description (Optional)**: Enter a brief description of the template.

3. In the Pick UI Style dialog box, click the template you added and click OK.

   This assigns the new base theme template to the customizable product.

Define a UI Property for the Customizable Product

The last step is to define a UI property for all the customizable product.

To define a UI property for the customizable product

1. Open the Customizable Product Show menu and choose User Interface Property Designer.
2. The User Interface Property Designer view appears.
3. In the box displaying the contents of the customizable product, click the name of the customizable product. It is the first item listed.
4. In Item Display Properties, click New.

   A new record appears.
5. Fill out the record.
   - **Name**. Enter the name of the UI property variable you entered in the base theme template. Do not include the period (.) that begins the name.
   - **Value**. Enter the customizable product name you want to display. Include any HTML formatting needed.
6. Save the new record.
7. Open the Item Display Properties menu and click Validate.

   This starts a configuration session. Verify that the customizable product display name is correct.

Modifying the Display Name of a Customizable Product, an Example

You have a customizable product called Premier Workstation. You want to display your company logo to the left of the product name in the base theme. The logo filename is logo1.gif. You have placed the file in the Siebel installation subdirectory \PUBIC\enu\IMAGES (NT path syntax, English language installation).

Use the Product UI Designer to create the Web pages for configuring the customizable product. Validate the customizable product, and verify that the pages display correctly.

To create a new base theme Web template and assign it to the customizable product

1. In the customizable product Work Space record, click the select button in the Base Theme field and open the dialog box.
2. In the Pick UI Style dialog box and write down the filename of the base theme.
3. Open the base theme Web template and save it to a new file name: eCfgNewProductTheme.swt
4. Open the new template and locate the swe.control element. Set CfgFieldName = ".NewProductName". The first character in the variable name must be a period (.). The variable name must be surrounded by quotes. Save the file.
5. In the customizable product Work Space record, click the select button in the Base Theme field and open the dialog box.
6. Click Add and add the new template. Then, select it as the base theme template.

To define a UI property for the customizable product

1 Open the User Interface Property Designer and select the customizable product.

2 Define a UI property for it as follows:

Name: NewProductName. Do not put a period before the name. This is the variable name you inserted in the template file.

Value: < img src = ”\images\logi1.gif” > Premier Workstation.

3 Open the Item Display Properties menu and click Validate.

This starts a configuration session. Verify that the customizable product display name is correct.

Modifying the Display Name of Groups

This section explains how to modify group names that display in the product themes. For example, you can use UI property definitions to define what displays on each tab in a tabbed product theme.

To change the name of a group in a product theme, you insert a variable in the product theme Web template. This variable tells the Web Engine to get the item name from a defined UI property rather than using the group name you defined in the Product UI Designer.

Here is the process for changing the display name of a group:

a Create a new product theme template. You do this by saving a copy of the product theme template. Then you insert a variable name in the new template.

b Assign the new product theme template to the customizable product.

c Define a UI property for all the groups you set up in the Product UI Designer. For each group name, you give the variable the desired value. This value is what displays in the configuration Web pages.

You must define a UI property for all the group names, not just the ones you want to change. This is because the product theme template determines the display name of the all the groups. When you insert a UI property variable in the template, the Web Engine gets all the group names from UI property definitions.

Create a New Product Theme Template

You must create a new Web template to customize the display of group names. You do this by copying an existing product theme template. Then you insert a UI property variable into the copy. Finally, you assign the new Web template as the product theme for the customizable product.

To insert a UI property variable into a product theme Web template, you must locate the swe:control element that governs the display of the group name:

- **Tab Theme**. Locate the first swe:case tag. It is located near the top of the file.
- **Single Page Theme**. Locate the third < table > tag. It is within the for-each loop near the top of the file.

- **Wizard Theme**. Locate the first for-each loop. It is near the top of the file.
- Beneath the location specified above, locate the first swe:control element. It defines the group data record. One of the attributes of this element is CfgFieldName = "CxGroupName".
- Replace CxGroupName with the name of the UI Property variable. The name must be preceded with a period (.). For example: ".NewGroupName".

To create a new product theme template

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. In the Work Space record, click the select button in the Product Theme field and open the dialog box.
4. In the Pick UI Style dialog box, write down the filename of the product theme.

   The filename is shown in the Template field. The filename ends in .swt.
5. Open the Web template and save it to a new filename.

   The Web templates are located in <install-dir>\webtempl (NT path syntax), where <install-dir> is the path to the Siebel installation directory. The new filename must end in .swt.
6. In the new template, locate the swe.control element containing the CfgFieldName = "CxGroupName" attribute.
7. Replace "CxGroupName" with a UI property variable name and save the file.

   The variable name must begin with a period (.). For example: ".NewGroupName". Verify that the variable name is enclosed in quotes.

Assign the New Product Theme Template

To assign the new product theme template to the customizable product, you first add the template to the Pick UI Style dialog box. Then you select it as the product theme.

To assign the new product theme template to the customizable product

1. In the customizable product Work Space record, click the select button in the Product Theme field and open the dialog box.

   The Pick UI Style dialog box appears.

2. In the dialog box, click New and fill out the form for adding a new template and click OK.

   - **Name**: Enter a descriptive name. For example: Modified Tab Theme.
   - **Template**: Enter the filename of the template you created.
   - **Description (Optional)**: Enter a brief description of the template.

3. In the Pick UI Style dialog box, click the template you added and click OK.

   This assigns the new product theme template to the customizable product.

Define a UI Property for all the Groups

The last step is to define a UI property for all the groups.

To define a UI Property for all the groups

1. Open the Customizable Product Show menu and choose User Interface Property Designer.

   The User Interface Property Designer view appears.

2. In the box displaying the contents of the customizable product, select the first group in the Group section.

3. In Item Display Properties, click New.

   A new record appears.

4 Fill out and save the record.

- **Name**. Enter the name of the UI property variable you entered in the product theme template. Do not include the period (.) that begins the name.
- **Value**. Enter the group name you want to display. Include any HTML formatting needed.

5 Save the new record.

6 Perform these steps for all the groups.

7 Open the Item Display Properties menu and click Validate.

This starts a configuration session. Verify that the group names display correctly.

Modifying the Display Name of Groups, an Example

You have a customizable product that has three groups: Group A, Group B, and Collection C. You want to change the name of Collection C to Group C in the selection pages.

Use the Product UI Designer to create the Web pages for configuring the customizable product. Validate the customizable product, and verify that the pages display correctly.

To create a new product theme template and assign it to the customizable product

1. In the customizable product Work Space record, click the select button in the Product Theme field and open the dialog box.
2. In the Pick UI Style dialog box and write down the filename of the product theme.
3. Open the product theme Web template and save it to a new file name: eCfgModifiedTabTheme.swt
4. Open the new template and locate the swe.control element. Set CfgFieldName = ".NewGroupName". The first character in the variable name must be a period (.). The variable name must be surrounded by quotes. Save the file.
5. In the customizable product Work Space record, click the select button in the Product Theme field and open the dialog box.
6. Click Add and add the new template.
7. Select the template as the product theme template.

To define a UI property for each of the groups

1. Open the User Interface Property Designer and select Collection C in the Group section.
2. Define a UI property for it as follows:
   - **Name**: NewGroupName. Do not put a period before the name. This is the variable name you inserted in the template file.
   - **Value**: Group C.
3. Select Group A.
4. Define the UI property for it as follows:
   - **Name**: NewGroupName. Do not put a period before the name.
   - **Value**: Group A.
5. Select Group B.
6. Define the UI property for it as follows:
   - **Name**: NewGroupName. Do not put a period before the name.
   - **Value**: Group B.
7. Open the Item Display Properties menu and click Validate.

   This starts a configuration session. Verify that the group names display correctly.

Modifying the Display Name of Items

This section explains how to modify the display name of an item, such as an attribute, relationship, linked item, or resource. A common reason for doing this is that you want the item name that the customer sees to be different than the name you have in the product table for a specific version of the customizable product.

For example, a computer component is called 256 MB disk drive in the product table. However, in an upcoming offering, you want to call this item the Standard 256 MB Disk. Rather than change the item name in the product table, you can change it using the User Interface Property Designer. This change is specific to the customizable product and is stored with it when you release the product. You can use the User Interface Property Designer to change item names with each release of a customizable product, without having to change the item name in the product table.

To change the display name of an item, you insert a variable in the UI control Web template that governs display of the item. This variable tells the Web Engine to get the item name from a defined UI property.

Here is the process for changing the display name of a relationship item in the form:

a Create a new UI control template. You do this by saving a copy of the UI control template you selected in the Product UI Designer. Then you insert a variable name in the new template.

b Assign the new template to the item.

c Define a UI Property for the item.

Create a New UI Control Template

You must create a new UI control template to customize the display of an item. You do this by copying an existing template. Then you insert a UI property name variable into the copy. Finally, you assign the new template as the UI control template for the item.

To insert a UI property variable into a Web template, you must locate the swe.control element that governs the display of the item. The UI control Web templates all have the same basic layout:

- A swe.include statement reads in a header file. This file places the relationship name at the top of the form that will contain the items.
- A swe for-each loop iterates through the relationship in the customizable product and loads its items into an array.
- A second swe for-each loop reads the array and constructs the form.
- A variable called DisplayValue near the beginning of the second swe for-each loop defines what item name appears next to each instance of the control in the form.
- The DisplayValue variable is set equal to a swe.control element that contains an attribute CfgFieldName = "CxObjName". Replace CxObjName with the name of the UI property variable. The name must be preceded with a period (.). For example: ".NewName".

To create a new UI control template

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. Open the Customizable Product Show menu and choose Product UI Designer.

   The Product UI Designer view appears.
4. In the Group List, click the group name containing the item whose display you want to modify.

   The items belonging to the group you select display in the Group Item List.

5 In the Group Item List, select the item you want to modify, for example a relationship.

6 In the record you selected, click the select button in the UI control field.

The Pick UI Style dialog box displays. The control you have selected for the group member is highlighted.

7 Write down the filename of the Web template that governs the display of this control.

The filename is shown in the Template field. The filename ends in .swt.

8 Open the Web template in a text editor and save it to a new filename.

The Web templates are located in <install-dir>\webtempl (NT path syntax) where <install-dir> is the path to the Siebel installation directory. The filename should begin with eCfg and end with .swt. For example: eCfgport_modifiedcheckbox.swt

9 In the new template, locate the correct swe.control element containing the CfgFieldName = "CxObjName" attribute.

10 Replace "CxObjName" with a UI property variable name and save the file.

The variable name must begin with a period (.). For example: ".NewName". Verify the variable name is enclosed in quotes.

Assign the New UI Control Template

To assign the new Web template to a group item, you first add the template to the Pick UI Style dialog box. Then you select it as the template for an item in the group.

To assign the new UI control template

1. In the Product UI Designer Group List, click the group name containing the item whose display you want to modify.

   The items belonging to the group you select display in the Group Item List.

2. In the Group Item List, click the item you want to modify.

   This selects the record.

3. In the record you selected, click the select button in the UI control field.

   The Pick UI Style dialog box appears.

4. In the dialog box, click New and fill out the form for adding a new template and click OK.

   - **Name**: Enter a descriptive name. For example: Modified Check Box.
   - **Template**: Enter the filename of the template you created.
   - **Description (Optional)**: Enter a brief description of the template.

5. In the Pick UI Style dialog box, click the template you added and click OK.

   This assigns the new template to the item and closes the dialog box.

6. In Group Item List, save the revised record.

Define a UI Property for the Item

The last step is to assign the variable in the new Web template to the item. Then you enter a value for the variable. The value you enter is what displays in a configuration session.

For example, if you assigned the template to a relationship, you must define a UI property for each item in the relationship.

To define a UI Property for the item

1. Open the Customizable Product Show menu and choose User Interface Property Designer.

   The User Interface Property Designer view appears.

2. In the box displaying the contents of the customizable product, select the item on which you want to define the UI property.

3. In Item Display Properties, click New.

   A new record appears.

4. Fill out the record.

   Name. Enter the name of the UI property variable. Do not include the period (.) that begins the name.

   Value. Enter the value you want to the variable to have for this item.

5. Save the new record.

6. Open the Item Display Properties menu and click Validate.

   This starts a configuration session. Verify that the item names display correctly.

Modifying the Display Name of Items, an Example

You have a customizable product that includes a relationship called Hard Drives. This relationship contains several disk drives. Along with the name of each drive, you want to display a picture of the drive. You want to assign radio buttons as the UI control for selecting the drives.

To prepare for defining the UI Property

1. Use the Product UI Designer to create the selection pages for configuring the customizable product.
2. Validate the customizable product, and verify that the pages display correctly.
3. Create a .gif or .jpg for each hard drive.
4. Test each file by displaying it in a browser. Use the HTML < img > tag to set the exact size of each image. The < img > tag will be the value you assign to the UI Property variable.

To create a new UI control template and assign it to the Hard Drives relationship

1. In the Product UI Designer, select the group containing the Hard Drives relationship.
2. In Group Item List, select the Hard Drives relationship.
3. Open the Pick UI Style dialog box and write down the name of the radio button template assigned to the relationship.
4. Open the radio button template and save it to a new file name: eCfgModifiedRadioButton.swt
5. Open the new template and locate the correct swe.control element.
6. Set CfgFieldName = ".NewName".

   The first character in the variable name must be a period (.). The variable name must be surrounded by quotes.
7. Save the file.
8. In the Product UI Designer Group Item List, select the relationship containing the hard drives.

9 Open the Pick UI Style dialog box and click Add to add the new template.

10 Select the new template as the control template for the Hard Drives relationship.

To define a UI property for each of the Hard Drives

1 Open the User Interface Property Designer and select the first drive in the Hard Drives relationship.

2 Define a UI property for it as follows:

- **Name**. NewName. Do not put a period before the name. This is the variable name you inserted in the template file.
- **Value**. Enter the HTML syntax for displaying the drive name and retrieving the image.

3 Repeat these steps to define a UI property for each hard drive in the relationship.

4 Open the Item Display Properties menu and click Validate.

This starts a configuration session. Verify that the item names display correctly.

Customizable Product Resources 15

This chapter explains how to use the Resource Designer to create configuration variables called resources to keep track of important configuration information when the user configures a customizable product.

Understanding Resources

Resources keep track of configuration variables that increase or decrease as the user configures a customizable product. For example, suppose you are defining a desktop computer customizable product. The product includes several types of chassis. Each chassis has a different number of slots for expansion cards. Allowable configurations also include several types of expansion cards, such as disk controllers, and graphics cards.

You don't know in advance which chassis the customer will select or how many expansion cards. However, you do know that you must keep track of the number of slots during the configuration process to make sure that the customer configures the computer correctly.

Resources are the way you do this:

a First define a resource to keep track of slots, for example slots-resource.

b For the class containing all the chassis', define an attribute, slots-provided, that tells how many slots are in the chassis. Typically, this attribute will have a single-value domain and the data type will be Number.

c For each class containing expansion cards, define an attribute, slots-required, that tells how many slots each card needs, usually 1. Typically, this attribute will have a single-value domain, and the data type will be Number.

d Finally, you write provide and consume rules in the Rules Editor to manage the slots-resource.

When the user selects a chassis, a provide rule adds the amount of the chassis' slots-provided attribute to the slots-resource. When the user selects an expansion card, a consume rule subtracts the amount of the card's slots-required attribute (1) from the slots-resource. In this fashion, the slots-resource keeps track of available slots in the computer chassis.

Once you define a resource, the definition is saved to a picklist and you can add the resource definition to other customizable products. Resources definitions have the data type Number. This means that they can only have numeric, integer, or floating point values.

Resource definitions include the following fields:

- **Name**. This is the resource name. It is how you will refer to the resource in configuration and pricing rules. The resource name does not display to the user. Since a resource name can be used by more than one customizable product, avoid making the name product-specific.
- **Type**. Select Number.
- **Description**. Enter a brief description of the resource. This description does not display to the user.

Creating a Resource

When you create a resource, it is automatically added to a picklist. You can then add the resource to other customizable products by selecting it from the picklist.

You must select and lock a customizable product before creating a resource.

When you create a resource, it is added to a dialog box. You can copy this resource definition to other customizable products and edit the definition as needed. In turn, the edited definition is added to the dialog box. When you remove a resource from a customizable product, it is removed from the dialog box.

To create a resource

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. In the Configuration Designer drop-down menu, select Resource Designer.

   The Resource Designer appears. This list contains all the resources defined for the product.
4. In the Resource Designer, click New.

   A new record appears.
5. To create a new resource, enter a name for the resource in the Name field.

   This name is the one the user sees if you include the resource in selection pages.
6. To use an existing definition, click the select button in the Name field and select the desired resource definition from the Pick Resource dialog box.
7. In the Description field, enter a description of the resource. The description is not displayed to users.
8. Save the record.

Editing a Resource Definition

You must select and lock a customizable product before editing a resource definition. If you change the name of a resource, the name is not changed in configuration rules where it appears.

Editing the name of a resource changes its name in the Pick Resource dialog box.

Deleting a Resource

You delete a resource by deleting the resource record from the Resource Designer. Deleting a resource from a customizable product deletes it from the Pick Resource dialog box.

You must select and lock a customizable product before deleting a resource.

To delete a resource

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. Remove the resource from any rules you have created.
4. In the Configuration Designer drop-down menu, select Resource Designer.

   The Resource Designer appears.
5. Select the desired resource definition.
6. Open the Resource Administration menu and choose Delete Record.
7. Click OK when asked to confirm you want to delete this record.

   The record is removed from the Resource Designer.

Managing Resources Using Configuration Rules

The most common way to manage a resource is to write provide and consume rules that add or subtract the value of an attribute from the resource. For example, you could write a configuration rule that contributes the number of slots in a chassis to a resource called slots available. You could also write configuration rules that consume slots from the resource when the user picks an expansion card.

By convention, the value of a resource must exactly equal the sum of all the contributors to the resource. Rules that consume or reduce the amount of a resource are negative contributors. The value of a resource is a computed value and cannot be directly set by a configuration rule.

For example you define resource R. You then write a configuration rule that sets the value of R to 5:

```
R = =5
```

When you validate the customizable product, this rule will be rejected by the system because it sets the value of R at an arbitrary value rather than allowing the value of R to be computed as the sum of all its contributors.

Customizable Product Links 16

This chapter describes how to use the Link Designer to define and manage links. Links allow you to extract information from Siebel business components and from system variables and use it to write rules.

Understanding Links

Links provide a way to use Siebel data in rules that you write for a customizable product. For example, if you have clients outside the U.S., you could create a link that stores the account location. You could then write a rule that uses the account location to determine what kind of power supply and plug types to include with a computer configuration.

Links can store two types of information. Business component links store the value of a field in a Siebel business component. System variable links store the value of a specific system variable.

The value of a link is determined when the user starts a configuration session and is not dynamically updated during the session.

Business Component Links

Business component links map a Siebel business component data field to a link name. The link name can then be used when writing rules for a customizable product.

To create a business component link, you must have a thorough understanding of Siebel business components and be able to use Siebel Tools to identify business objects, business components, and field names.

When you define a business component link, the goal is to retrieve only one record. Several fields are provided in the link definition to help you do this. If more than one record is retrieved by the query, the link data is extracted from the first record in the group. If no records are retrieved by the query, the value entered in the default value field in the link definition is used.

You have the option to extract information from the current instance of a business component or from a new instance. For example, you select an account as part of creating a quote. You have defined a link for a complex product that extracts information from the business component that displays this record. When the user begins configuring the product, the link information will be extracted from the account record being used in the quote. The link uses the current instance of the business component.

You can also define the link on a new instance of the business component. The information will be extracted from the first record returned by the business component. You can control which record is returned by specifying search and sort parameters.

A business component link definition contains the following fields:

- **Name**. This is the name of the link. Use it to refer to the link when you write rules. This field is required.
- **BusObj Name**. This field specifies the business object in which the business component resides. This field is required for business component links.
- **BusComp Name**. This field specifies the name of the business component from which you want to extract information. This field is required.
- **BusComp Field Name**. This field specifies the name of the field in the business component that contains the data you want to extract. This field is required.
- **Expression**. This field contains an XML expression that is automatically generated by your entries in the other fields. No entry is required in this field.
- **Needs Execution**. Put a check mark in this field if you want to retrieve the link information from a new instance of the business component. Leaving this field blank allows you to retrieve information about the current instance of the business component.
- **Search Spec**. Enter a Siebel query-by-example expression to narrow the search to one record. This field is evaluated only if you put a check mark in Needs Execution. An entry in this field is highly recommended.
- **Sort Spec**. Enter a sort specification so that the desired record appears first if more than one record is retrieved. This field is evaluated only if you put a check mark in Needs Execution. An entry in this field is highly recommended.

- **Default Value**. Enter the value that you want to assign to the link if the query returns no records. This field is highly recommended if you put a check mark in Needs Execution.
- **Keyword**. Leave this field blank.
- **Description**. Write a brief description of what the link does. This description is not displayed to customers.

System Variable Links

Links can be defined to extract information from two system variables, TODAY and WHO. The TODAY system variable returns today's date. The WHO system variable returns the log-in name of the user who started the configuration session.

You can use the TODAY variable to write time-sensitive rules. For example, you create a link named TodayDate that stores the value of the TODAY system variable. You could then write a rule that says if today's date is later than December 23, 2001, then the product 64 MB RAM is required in computer configurations.

You can use the WHO variable to customize configuration rules based on the user log-in name. For example, you create a link named UserName that stores the value of the WHO system variable. You could write a rule that says if the user's log-in account name is jsmith, then 64 MB RAM is required in computer configurations.

A system variable link definition contains the following fields:

- **Name**. This is the name of the link. Use it to refer to the link when you write rules. This field is required.
- **Description**. Write a brief description of what the link does. This description is not displayed to customers.
- **Keyword**. Click the down-arrow and select either TODAY or WHO. This field is required.

Leave all the other fields blank.

Creating a Business Component Link

A business component link lets you extract information from Siebel business components and use it to write rules.

To create a business component link you must know the business component name and field name containing the information you want to extract.

You must select and lock a customizable product before creating a link. When you create a link, it is automatically added to a picklist. You can then add the link to other customizable products by selecting it from the picklist.

When you create a link, it is added to a dialog box. You can copy this link definition to other customizable products and edit the link as needed. In turn, the edited link is added to the dialog box. When you remove a link from a customizable product, it is removed from the dialog box.

The following fields are mandatory:

- **Name**. You can enter a link name, or pick an already-defined link from a dialog box by clicking in the field. This is the name the user sees in selection pages during a configuration session.
- **BusObj Name**. This field specifies the business object in which the business component resides.
- **BusComp Name**. You can enter a business component name, or pick a business component from a dialog box by clicking in the field.
- **BusComp Field Name**. You can enter a field name, or pick a field from a dialog box by clicking in the field. The dialog box displays the fields of the business component in BusComp Field Name.
- **Needs Execution**. Place a check mark in this field if you do NOT want the retrieved data to be session-related.

The following fields are highly recommended:

- Search Spec
- Sort Spec
- Default Value

Leave the Keyword field blank.

To create a business component link

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. In the Configuration Designer drop-down menu, select Link Designer.

   The Link Designer appears. This list contains all the links defined for the product.
4. In the Link Designer, click New.

   A new record appears.
5. Complete the fields in the record.

   For those fields with a picklist dialog box, you can search a list of available choices and make a selection. When you specify a business object, the BusComp Name dialog box displays only the business object's business components.

   To use an already-existing definition, click in the Name field and select the desired link definition from the dialog box.
6. If you do not want the information retrieved to be session-related, click the Needs Execution check box in the form.
7. Save the record.

Creating a System Variable Link

A system variable link lets you obtain the value of the following system variables and use it to write rules:

- **TODAY**. Provides the system date. The data type is Date and can be used for date computations.
- **WHO**. Provides the user's login name. The data type is Text.

You must select and lock a customizable product before creating a link. When you create a link, it is automatically added to a picklist. You can then add the link to other customizable products by selecting it from the picklist.

When you create a link, it is added to a dialog box. You can copy this link definition to other customizable products and edit the link as needed. In turn, the edited link is added to the dialog box. When you remove a link from a customizable product, it is removed from the dialog box.

The following fields are mandatory:

- Name
- Keyword

The Description field is recommended. Leave all other fields blank.

To create a system variable link

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. In the Configuration Designer drop-down menu, select Link Designer.

   The Link Designer appears. This list contains all the links defined for the product.
4. In the Link Designer, click New.

   A new record appears.
5. In the Key Word drop-down menu, click TODAY or WHO.

6 Enter a name in the Name field and a brief description in the Description field.

To use an already-existing definition, click in the Name field and select the desired link definition from the dialog box.

7 Save the record.

Editing a Link Definition

You must select and lock a customizable product before editing a link definition. If you change the name of a link, the name is not changed in configuration rules where it appears.

Editing the name of a link changes its name in the Pick Linked Item dialog box.

Deleting a Link

You delete a link for a customizable product by deleting the record from the Link Designer. Deleting a link from a customizable product deletes it from the Pick Link dialog box.

You must select and lock a customizable product before deleting a link.

To delete a link

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.
3. Remove the link from any rules you have created.
4. In the Configuration Designer drop-down menu, select Link Designer.

   The Link Designer appears. The list contains all the links defined for the product.
5. In the Link Designer, select the desired link definition.
6. Open the Link Designer menu and click Delete Record.
7. Click Yes when asked to confirm you want to delete the record.

   The record is removed from the Link Designer.

Customizable Product Rule Designer 17

This chapter explains how to use the Rule Designer to create rules. It also explains how to create rule templates.

Understanding the Rule Designer

The Rule Designer is where you create and manage configuration rules for a customizable product. It is also where you create rule templates that can be used for all customizable products.

A configuration rule defines how two items in a customizable product are related. For example, Component A and Component B are mutually exclusive. If the user picks one, then you want to prevent them from picking the other. One way you can do this is by writing a configuration rule: Component A excludes Component B. The Rule Designer provides a rule template to help you write this rule.

Rule templates are rule statements where you replace variables in the statement to create a configuration rule. The Rule Designer provides rule templates for the most common types of configuration rules. You can also create your own rule templates.

In the Rule Designer, you create a configuration rule by first selecting a rule template. Then you pick items from the customizable product and operators or even other rule templates to replace the variables in the rule. Both arithmetic and logical operators are provided by the Rule Designer.

Configuration rules you create apply only to the current customizable product, and are stored as part of it. In contrast, rule templates reside in the Rule Designer and can be used with any customizable product.

To use the Rule Designer, you select a customizable product, then click Configuration Designer and select Rule Designer in the drop-down menu.

The Rule Designer has three parts:

- **Rule listing**. When you go to the Rule Designer, all the rules defined for a customizable product are listed. You can edit, copy, and delete the rules in the listing.
- **Rule template listing**. When you click New Rule or New Template in the rule listing, the rule template listing appears. This listing contains the pre-defined rule templates in the Rule Designer. It also lists any templates you have created.
- **Rule statement**. When you select a rule template and click Continue, the rule template appears in the Rule Statement tab. You then select items from the customizable product, operators, or other rule templates to replace the variables in the rule statement.

Rule Listing

The rule listing displays all the rules defined for the customizable product. You can edit, copy, and delete the rules in the listing. The listing has the following fields:

- **Name**. You specify the rule name when you save the rule. Organize your rule names so you can locate them using the Find button. For example, consider including the rule type (excludes, requires and so on) in the rule name. Searches using only the Name field would then return groups of rules having the same rule type; for example, all the exclude rules.
- **Rule**. This field contains the rule statement and is not editable. To edit the rule, click Edit.
- **Explanation**. You specify the rule explanation when you save the rule. Organize your explanations so that you can locate the rule using the Find button. For example, consider including information that uniquely identifies the rule, such as item names. Searches using Name and Explanation would then return the specific rule.
- **Start Date**. You can specify a start date on which the rule becomes effective. After you release the customizable product, the system does not use the rule to compute solutions until the specified start date. When you specify a start date, you do not have to specify an end-date.

- **End Date**. You can specify a date on which a rule becomes inactive. After you release the customizable product, the system stops using the rule to compute solutions on the specified date. When you specify an end date, you do not have to specify a start date.
- **Inactive**. When you put a check mark in this box, the rule is not used to compute solutions. Use this feature in the current work space to simulate the behavior of rules that will have a start date, end date, or both when you release the product. You can also use this feature to deactivate a rule but retain it in a released version of the product.

Rule Template Listing

The rule template listing displays all the rule templates that come with the Rule Designer plus those you have created. It displays in the "Pick a rule" tab when you click New rule or New template in the rule listing.

The rule templates provide the basic rule types you need for creating configuration rules. For example, there are rule templates for exclude rules, others for require rules, and so on.

The templates in the listing are part of the Rule Designer and can be used to create rules in any customizable product. The templates are not specific to the customizable product you are working on.

Each rule template contains variables that you replace to create a configuration rule. You can replace the variables with items from the customizable product, links, resources, expressions, or other templates.

Rule Statement

When you select a rule in the rule template listing and click Continue, the Rule Statement tab displays. It contains the rule template you selected. You build a configuration rule by replacing the variables in the statement with items from the customizable product, with resources or links, with operators, or with other rules. To move to another variable in the rule statement, click it. The currently selected variable in the rule statement displays with square brackets around it. Variable that are not current but can be selected, display an underline when the cursor is placed on them. When you select an item for a variable, it displays in red.

The items you can replace a variable with are grouped in the “Insert a” tab, located below the Rule Statement. When you move between variables in the rule, the groupings change to reflect your allowable choices.

In some templates when you replace a variable with a value, typically an expression, the Compound button becomes active. The Compound button lets you nest expressions within expressions. For example, you could use the Compound button to add two variables together where the second variable is itself an expression that adds two variables.

Use the following process to build a configuration rule:

a Select the desired rule template in the rule template listing.

b Select the first variable in the rule.

c Click the desired item grouping.

d Pick the product, operator, or rule template you want to insert.

e Move to the next variable and insert the desired item.

f When you are finished, save the rule.

Understanding Class-Product Rule Inheritance

You can designate a customizable product as a class-product and then add it to a product class. When you do this, all products belonging to the class and its subclasses inherit the class-product's structure and its configuration rules. For example you define the following rule in a class-product and name the rule Rule 1:

Any item from Relationship A requires selection of any item of Relationship B

You can control how this rule is inherited by other customizable products you add to the class containing the class-product. You do this by inserting a rule in these products that has the same name as the one in the class-product.

For example, you have a product class containing a class-product. The class-product contains Rule 1, as shown above. The class also contains three customizable products, CP1, CP2, and CP3. Table 21 shows how inheritance of Rule 1 from the class-product works.

Table 21. Rule Inheritance

Customizable Product	Native Rule	Result
CP1	No rule named Rule 1	Inherits rule from the class-product.
CP2	Rule 1: X excludes Y	Native Rule 1 (X excludes Y within CP2) overrides inheritance of Rule 1 from the class-product. If CP2 is selected, Rule 1 in the class product is not enforced in CP2.
CP3	Rule 1: ” “	Blank native Rule 1 overrides inheritance of Rule 1 from the class-product. Rule 1 in the class product is not enforced in CP3.

You can use named rules to control how rules are inherited from class-products.

- If you want a customizable product to inherit a named rule from a class-product, the customizable product must not contain a rule of the same name.
- If you want only some customizable products to inherit a named rule from a class product, define a blank named rule with the same name in the customizable products where you do not want inheritance to occur.
- If a rule in a class-product has no name, it cannot be overridden by a rule in a customizable product that inherits the structure of the class product.

Creating a Configuration Rule

Observe the following guidelines when creating rules:

- Create at least one rule early in the process of building a customizable product. The presence of a configuration rule, even if it is inactive, causes eConfigurator to check the product for errors more rigorously when you go to validate mode.
- Avoid writing rules that use large quantities until you have verified the logic of the rule. For example, write a rule that refers 10 items and check it before changing the rule to refer to 10,000 items. This prevents needless solution searches if the basic logic of the rule is incorrect.
- If the customizable product will be designated as a class-product, consider not giving names to its rules. This prevents inadvertent override of rules by customizable products that inherit the class-product's rules and have native rules of the same name.
- Test each rule after you create it. Consider inactivating rules that are unrelated to the new rule to facilitate troubleshooting. Test rules by starting a configuration session and selecting the affected items. To start a configuration session, open the Rules List menu and choose Validate.

To create a configuration rule

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.

   If you omit this step, the most recently released version of the customizable product is loaded in the Rule Designer, and you cannot create rules.
3. Click the Configuration Designer tab.

   The Rule Designer Rules List appears. It lists all the configuration rules that have been created for this customizable product.
4. Click New Rule.

   The "Pick a rule" tab appears and lists the rule templates available for creating rules. The Rule Statement tab displays the syntax of the currently-selected rule.

5 Select the desired rule template in "Pick a rule", and click Continue.

The Rule Statement and "Insert a" tabs appear. The Rule Statement tab contains the rule template you selected. The "Insert a" tab lists the item groups available for the currently-selected variable in the rule.

To return to the display of all the Web templates, click Back. To exit and return to the Rules List, click Cancel.

6 In Rule Statement, click the first variable you want to work on.

Variables are enclosed in square brackets. When you click a variable, it turns red to indicate it is selected.

7 In the "Insert a" list, select the item grouping containing the item you want to insert. In the dialog box, choose the desired item.

When selecting products, click the product's select button. If you click the product name, the dialog box displays product information.

The variable in the rule template is replaced by the item.

8 Repeat these steps for each variable until you have built the desired rule.

9 Click Save Rule to save the rule.

The Save button becomes active when you have selected values for all the variables in the rule. Clicking the Save button causes the Save Rule form to appear.

10 Fill out the fields in the Save Rule form. All fields are optional. Then click Save.

The rule displays in the Rules List.

11 Open the Rules List menu and click Validate.

This starts a configuration session. Verify that the rule works correctly.

Editing a Rule

In the Rules List, you cannot edit the definition of the rule in the Rule column. To edit the definition, you must display the rule in the Rule Statement tab, make your changes, and save the rule. When you save the rule, you can overwrite the rule with the changes or save the changes as a new rule.

The following procedure explains how to edit a rule definition and overwrite the rule with the changes.

To edit a rule

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.

   If you omit this step, the most recently released version of the customizable product is loaded in the Rule Designer, and you cannot create or edit rules.
3. Click the Configuration Designer tab.

   The Rule Designer Rules List appears. It lists all the configuration rules that have been created for this customizable product.
4. Select the desired rule.
5. Open the Rule List menu and choose Edit Record. The Rule Statement and "Insert a" tabs display. Edit the rule and click Quick Save.

   The edited rule displays in the Rules List.
6. Open the Rules List menu and click Validate.

   This starts a configuration session. Verify that the edited rule works correctly.

Copying a Rule

When you copy a rule, the system creates an exact duplicate of the rule and displays it in the Rule Designer. You can then edit the rule definition as desired.

If you copy a rule and make no changes to the copy, two exactly equivalent constraints are used to compute each solution. This does not cause a problem, and solutions are computed as if there was only one constraint.

Use the copy feature to create groups of rules that are similar. Start by creating the basic rule. Then copy it once for each rule in the group. Finally, edit the copies to create the rules in the group.

To copy a rule

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.

   If you omit this step, the most recently released version of the customizable product is loaded in the Rule Designer, and you cannot create or edit rules.
3. Click the Configuration Designer tab.

   The Rule Designer Rules List appears. It lists all the configuration rules that have been created for this customizable product.
4. Select the rule you want to copy.
5. Open the Rules List menu and choose Copy Rule.

   A copy of the rule appears in the Rules List. Its name begins with “Copy of.”
6. Click in the Name field and edit the rule name as desired.
7. Edit the rule statement, explanation, and start/end dates as desired.
8. Open the Rules List menu and click Validate.

   This starts a configuration session. Verify that the new rule works correctly.

Deleting a Rule

Deleting a rule removes it from the customizable product. The template on which the rule was based is not removed and remains available.

To delete a rule

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.

   If you omit this step, the most recently released version of the customizable product is loaded in the Rule Designer, and you cannot create or edit rules.
3. Click the Configuration Designer tab.

   The Rule Designer Rules List appears. It lists all the configuration rules that have been created for this customizable product.
4. Select the rule you want to delete.
5. Open the Rules List menu and choose Delete Record.

   Click OK when asked if you want to delete the record. The record is removed from the Rules List.
6. Open the Rules List menu and click Validate.

   This starts a configuration session. Verify that the customizable product rules function correctly.

Creating Groups of Related Rules

Related rules are those that have the same basic construction but differ only in content. For example, you need to create a dozen exclude rules of the form Product A excludes Product B. For each rule the products will be different, but the basic construction is the same.

There are several processes you can use to create groups of related rules. In the following example, you are going to create twelve exclude rules. Here are the processes you can use to create them:

a Create each rule using the Exclude template in the "Pick a Rule" tab of the Rule Designer.

b Create the first rule and save it. Then edit the rule so that it becomes the second rule and save the rule. In the Save Rule form, click "Save changes as new rule." This creates the second of the twelve rules. Repeat these steps to create the remaining rules.

c Create the first rule and save it. Then copy the rule 11 times. Edit each of the copies. In the Save Rule form, click Save to overwrite the copy with the changes.

d Create a rule and save it as a template. Then select this template to create the remaining rules.

Setting Effective Dates for Rules

For each rule you create, you can set effective dates that control when the rule is active. You can set both a start date and an end date. On the start date the rule is used to compute all solutions presented to the user when they configure a product. On the end date, the rule is no longer used to when computing solutions.

Specifying start and end dates in combination has the following effects. In the following descriptions, active means the eConfigurator engine uses the rule to compute solutions. Inactive means the rule is not used to compute solutions:

- **Both a start and end date specified**. The rule becomes active on the start date and becomes inactive on the end date.
- **Start date specified**. The rule becomes active on the start date and remains active thereafter.
- **End date specified**. The rule is active when the version is released for use, and becomes inactive on the end date.

The start date is determined using the Siebel server's system clock. The start and end dates work as follows in relation to the date the product is released (release date):

- If the release date is earlier than the start date, the rule becomes active on the start date.
- If the release date is later than the start date, the rule is active when the product is released.
- If the release date is earlier than the end date, the rule becomes inactive on the end date.
- If the release date is later than the end date, the rule is inactive when the product is released, and the rule remains inactive.

When you are validating a product, you can temporarily activate or deactivate rules in the current work space by clicking the rule's Inactive box in the Rules List. This lets you simulate how the rule will behave on the start and end date.

For example, you can test a rule with a start date in Validate mode using the following process:

a Click in the rule's Inactive box to deactivate the rule. Then go to Validate mode and test the product. This simulates what users will see before the start date when the rule is not being used to compute solutions.

b Click in the rule's Inactive box again to activate the rule. Then go to Validate mode and test the product. This simulates what users will see after the start date when the rule is being used compute solutions.

You can specify start and end dates for rules when you create them or by editing rules after they have been created.

Deactivating a Rule

When you create a rule, it is active by default and is used to compute all solutions (unless the rule is not active in advance of a start date).

When you deactivate a rule, it is not used to compute solutions. One reason for deactivating a rule is to help you test rules in Validate mode. You can deactivate a group of rules and then activate them one at a time to see how each affects the product's behavior when it is being configured.

Another reason to deactivate rules, is when you want to release a version of a product that does not require a rule to be active. You can deactivate the rule and then release the product. The rule is inactive in the released version and is not used to compute solutions.

When you deactivate a rule, a check mark displays in the Inactive check box.

To deactivate a rule

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.

   If you omit this step, the most recently released version of the customizable product is loaded in the Rule Designer, and you cannot create or edit rules.
3. Click the Configuration Designer tab.

   The Rule Designer Rules List appears. It lists all the configuration rules that have been created for this customizable product.
4. Select the rule you want to edit and click the Inactive box.

   A check mark appears in the check box indicating that the rule is deactivated.
5. Save the rule.

   When you activate a rule that is inactive, the check mark is removed from the Inactive check box.
6. Open the Rules List menu and click Validate.

   This starts a configuration session. Verify that the customizable product works correctly.

To activate an inactive rule

1. Select the rule.
2. Click in the rule's Inactive check box.

   This removes the check mark and activates the rule.
3. Save the rule.
4. Open the Rules List menu and click Validate.

   This starts a configuration session. Verify that the customizable product works correctly.

Creating a Rule Template

When you create and save a rule, the rule becomes part of the customizable product. The rule is not visible in other customizable products.

When you create and save a rule as a template, it is added to the list of templates. The list of templates is visible in all customizable products. Create templates for those rules that you will use with several customizable products.

Templates that refer to items in one customizable product cannot be used to refer to items in another customizable product. Items include products, relationships, links, links and resources. For example, you write the following rule for customizable product CP1and save the rule as a template called A Requires B:

Product A requires Product B

You also have customizable product CP2, that includes Product A and Product B. You want to write the same rule for CP2.

If you use the template A Requires B in CP2, you will receive a validation error when you validate CP2. This is because each item in a customizable product receives a unique item ID. This item ID is what the system stores as the item name when you create a rule or a rule template in a customizable product. This ID is not transferable to other customizable products.

Since rule templates can be used across all customizable products, rule templates cannot be edited or deleted.

To create a rule template

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.

   If you omit this step, the most recently released version of the customizable product is loaded in the Rule Designer.
3. Click the Configuration Designer tab.

   The Rule Designer Rules List appears. It lists all the configuration rules that have been created for this customizable product.

4 Open the Rules List menu and choose New Template.

The Pick a rule list appears.

5 Create the desired rule that you want to use as a template.

6 Click Save Template and fill out the Custom Template Definition form.

- **Name**. This is the template name that displays in the Pick a rule list.
- **Template Identifier**. This text string uniquely identifies the template. It does not display to users.
- **Spec**. This field describes the template syntax. Tip: The rule syntax displays in the form below this field. Highlight the syntax and copy it into the Spec field.
- **Description**. This field describes what the template does. The description displays in the Pick a rule list.

7 Click Save to save the template.

The new template name appears in the Pick a rule list.

Editing or Deleting a Rule Template

Rule Templates cannot be edited or deleted. This is to prevent unintended problems across multiple customizable products where templates have been used.

Obtaining a Rule Summary Report

You can obtain a report that lists all the configuration rules in a customizable product. For each customizable product, the report shows the following information:

- Rule name
- Rule Statement
- Explanation
- Start date
- End date
- Inactive
- Updated date
- Updated by

The Rule Summary displays in the Siebel Report Viewer. You can print the report or create an email attachment.

This report must be enabled on the report server before performing the following procedure.

TIP: The on-screen display of the report typically lists more products on each page than the Products list. Use the report to scan quickly through the product table.

To obtain a rule summary report

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.

   If you omit this step, the most recently released version of the customizable product is loaded in the Rule Designer.
3. Click the Configuration Designer tab.

   The Rule Designer Rules List appears. It lists all the configuration rules that have been created for this customizable product.
4. In the application View menu, select Reports.
5. In the Reports dialog box, select Rule Summary.
6. To run the report now, click Run Now.

   The report window appears, and you can view the report and print the report as desired.
7. To schedule the report to run at a scheduled time, click Schedule.

   A form appears for scheduling the report.

Configuration Rule Template Reference 18

This chapter explains how the configuration rule templates in the Product Rule Manager work.

Understanding Constraints

The restrictions that define allowable configurations are called constraints. Constraints can take several forms:

- Picking one item requires that another item also be included.
- Picking one item excludes being able to pick another specific item.
- Picking an item from a group is mandatory.
- An item is available to be picked only after a certain date, or only if the user is a certain account.
- Picking an item consumes a configuration-related resource such as the total available slots in a chassis. Picking an item could also provide something to a resource.
- An item is required or excluded based on a defined condition.

Attribute definitions, cardinality, interface design, and configuration rules are the methods you use to create the configuration constraints that enforce business rules and logic.

You test these constraints by going to validation mode. In validation mode, you configure the product as if you were the end user. This lets you verify that the user interface works correctly and that all constraints are functioning properly.

Attribute Definitions

Attributes define the options for an individual component. For example, a component has the attribute Color. The Color attribute defines the set of colors that a user can choose for the component.

Attributes are defined at the class level in the class system and are inherited by all products assigned to the class. This means you can define and manage attributes for large groups of products and components from a single location. You do not have to define these options for each product or component individually.

Cardinality

You add components to a customizable product by defining relationships. A relationship can contain a single component, all or part of a product class, or a group of products from several classes.

When you define a relationship, you can specify a minimum, maximum, and default cardinality. The cardinality defines whether or not the user is required to select items and also how many items they can select.

- **Minimum cardinality**. Setting the minimum cardinality to greater than 0 means the user is required to make a selection from this relationship. This creates a "required selection." Setting the minimum cardinality to 0 means the user is not required to make a selection.
- **Maximum cardinality**. Setting the maximum cardinality constrains the quantity the user can select from the relationship. This creates a "maximum allowable selection." Setting the maximum cardinality to blank, means that the user can select an unlimited number of items from the relationship.
- **Default cardinality**. The default cardinality defines how many items from the relationship will be added by default to the configuration when the user first starts the configuration session. Depending on the settings for minimum and maximum cardinality, the user can remove the items or add more.

Cardinality is defined at the relationship level and applies to the relationship as a whole. It constrains the quantity of items you can select from a relationship. If the relationship contains one item, cardinality constrains the selection of that item. If the relationship contains a group of items, cardinality applies to selection of these items in any combination.

Upgrade users. Defining the cardinality of a relationship in this release is roughly equivalent to writing an enforced quantity rule on a category in previous releases.

User Interface Design

The way you define groups and choose controls for them in the Product UI Designer can create implicit constraints that control item selection. For example, choosing a radio button control for a group constrains the user to choosing only one item from the group, even if maximum cardinality for the relationship is blank. In other words, using a radio button control creates an implicit constraint: picking one item excludes all the others in the group.

By creating groups in the Product UI designer that contain more than one relationship, you can create more complex implicit constraints. For example, you have two relationships, A and B, that each contains three components. All of the items in the two relationships are optional (minimum cardinality = 0 for both relationships). The final configuration has the following constraint: if the user picks any item in either relationship, all the remaining items in both relationships are excluded from the final configuration. You can enforce this constraint by assigning a radio button control to this group that allows selection of only one item.

Upgrade users. In this release, you can define the configuration UI with the Product UI Designer. In previous releases, you had to use Siebel Tools to change the layout of Configuration Assistant, and this change applied to all products. If you do not define a configuration UI for a customizable product, the system provides a default configuration UI that displays all the items on one selection page.

Configuration Rules

The Rule Designer provides rule templates that allow you to create a wide variety of configuration rules. These explicit constraints differ from the implicit constraints you can create using attributes, cardinality, and the Product UI Designer in the following ways:

- Configuration rules can be based on conditions that occur during the configuration session. For example, you can write a configuration rule: if condition A is true, then Product B is required.
- Configuration rules can keep track of important consumables in a configuration session. For example, you can create a resource called slots available, and then write rules that increase or decrease slots available depending the items the user picks.
- Configuration rules can be complex. For example you can write rules that contain multiple levels of conditions or contain nested expressions.

- Configuration rules can override implicit constraints created by the physical construction of the user interface. For example, you have specified radio buttons to display the items in a relationship. This allows the user to pick only one item from the relationship. However, you could also write configuration rules that require additional items from the relationship when the user picks items elsewhere in the configuration.
- In the example regarding relationship A and B in the previous section, the user is constrained from selecting more than one item from the two relationships by the physical construction of the user interface. The radio button control allows only one item to be selected from the group. However, you could write a configuration rule that says that for each Component A that is picked, then Component C from Relationship B must be in the configuration. Each time the user picks Component A, the system will automatically add Component C to the configuration.
- Configuration rules can use Siebel data external to the customizable product. You can write rules conditioned on the date, the account that is configuring the product, or on other business component data. You access external data by defining links.

The configuration rules you write in the Rule Designer apply only to the current customizable product and are stored with it. You can also create rule templates. These are stored with the Rule Designer and can be used with any customizable product.

Upgrade users. The Rule Designer in this release replaces the Basic Rules Designer, Logic Designer, and Advanced Rules Designer in previous releases. The Rule Designer uses natural-language rule templates to let you write rules as simple or as complex as needed. You can also write rules in Advanced Rule Language if desired.

Resource Values

The value of a resource must exactly equal the sum of all its contributors (provides and consume rules). If you set a resource to an initial value, or constrain a resource within a range, there must be items in the solution that contribute to the resource. You do this by writing rules that contain contributors to the defined resources.

Be careful when using the equals operator [=] with these templates. It can cause unexpected results. For example, you write the rule:

Resource R = 10

In addition, the only contributor to Resource R is Product P, which has the following rule:

Product P provides 2 to Resource R

The eConfigurator engine will return a solution that contains 5 Product P. This is because Resource R must equal 10, and the only way to achieve this is to add enough contributors (Product P) to make this true.

If you had set Resource R to 11, instead of 10, you would receive a validation error. This is because there is no way to add enough Product P to make Resource R exactly equal to 10.

Understanding Configuration Rule Processing

Unlike procedural languages like Siebel eScript or C++, the rules you create in the Rule Designer are elements of constraint programming. Constraint programming differs from procedural logic in several important ways.

In a procedural language, you write statements that are executed one after another. Control can be transferred to other parts of a program, but the method of execution remains serial. Procedural logic relies on groups of statements executed in order.

In constraint logic, you write rules, called constraints, that describe what must be true about the solution. The eConfigurator engine organizes these rules into a search plan and then searches systematically, trying different item values, until a solution that satisfies all the constraints is found. Constraints are evaluated in parallel rather than serial fashion. Their order of evaluation is unimportant.

For example, you write a series of configuration rules that define all the ways a desktop computer can be configured. This is called the declarative portion of the customizable product, and the rules are constraints on every solution. The eConfigurator engine requires that all constraints are observed in every solution. This is the key to understanding constraint programming: *all* constraints must be observed in *every* solution.

You then release this customizable product to users. Users interact with it by selecting components to configure a computer. Each item the user selects is treated by the eConfigurator engine as another constraint on the solution. These user-constraints are added to the constraints in the declarative portion of the product and are used to further narrow down the solutions the engine can create. If a user-constraint conflicts with a constraint in the declarative portion of the product, the user receives a message that provides options for resolving the conflict.

The eConfigurator engine must find a configuration that satisfies all the rules (constraints) in the declarative portion of the product, plus all those created by the user's choices (user-constraints). After the user adds or removes an item, the eConfigurator engine searches until it finds a solution that satisfies all the constraints, including the constraint created by the user's action. The eConfigurator engine then presents the solution. In many cases, the only thing that changes is that the item is added or removed.

However, other items may be added or removed depending on constraints in the declarative portion of the product. If the user selects an item and the eConfigurator engine cannot create a solution that satisfies all the constraints, the user is presented with an option to undo the current selection or previous selections so that all constraints can be satisfied.

Note that constraints are created both by the modeler and by the user. In the declarative portion of the product, the product administrator writes configuration rules that define the relationships between items. For example, if item A is picked, item B is required. The user creates constraints by adding items. For example, picking item A creates a constraint that item A must be in the solution. The constraints that drive the solution are thus jointly provided by both the product administrator and the user.

It is important to understand that to produce a solution, the eConfigurator engine is free to do what is necessary to find a valid solution that satisfies all the constraints (declarative portion constraints and user-constraints). For example, the following are the only two constraints on items A and B in a customizable product:

The quantity of A < the quantity of B

The quantity of B != 4

If the user picks one A, the eConfigurator engine will require that there are at least two Bs in the solution. If the user then increases the quantity of A to two, the eConfigurator engine generates a solution in which there are at least three Bs. However, when the user adds the third A, instead of a solution that increases B by one, the new solution will have two more Bs, making the total number of Bs at least five. This occurs because both rules must be satisfied, and because there is no constraint preventing the eConfigurator engine from generating solutions that increment B by more than one.

The following example illustrates that when there are several possible alternatives, the eConfigurator engine may choose any one of them. You have a customizable product with only the following two constraints on items A, B, and C:

Constrain A + B = C to be true
Constrain C = 1 to be true

In this example, either A or B can be zero, but not both. Neither can both be 1. The eConfigurator engine will choose either A or B and will actively exclude the other from the solution. For example, it could choose A and actively exclude B from the solution. It could also choose B and actively exclude A.

It is important when creating rules in the Rule Designer, to consider the domain of solutions the engine could produce. Otherwise, users may encounter unexpected results when they make selections. In the example above, you could write a rule that prints a message to the user that A + B must equal 1 and then let the user choose which one they want.

Understanding Rule Conditions

In the Rule Designer, many of the rule templates contain conditions or expressions. For example:

- Exclude template: [Item or condition] excludes [item or condition]
- Require template: [Item or condition] requires [item or condition]

A condition is an explicit statement about the configuration. Conditions can play several roles in a rule template. First, they can act as a test that determines whether a rule is enforced. For example, you write the following rule:

Item A > 4 excludes Item B

This rule states that when the quantity of Item A is greater than 4 in the solution, then Item B cannot be present (is excluded). In this rule, "Item A > 4" is a condition that, when true, causes Item B to be excluded.

When a condition is used as a test, the eConfigurator engine evaluates the condition and returns true or false. If the condition is true, the rule is enforced.

Secondly, conditions can define a constraint. For example, you write the following rule:

Item B excludes Item A > 4

This rule states that when Item B is present in the solution, then the quantity of Item A in the solution cannot be greater than 4. In this rule, "Item A > 4" is a condition that defines a constraint.

Conditions can take several forms:

- **Quantity comparisons**. The preceding examples are quantity comparisons.
- **Item values**. The value of attributes, linked items, and resources can be used as conditions.
- **Rule Templates**. Rule templates can be used as conditions. When rule templates are used as conditions, the eConfigurator engine does not enforce the rule as a constraint but instead evaluates the template as true or false. This is discussed further below.

Boolean operators (AND, OR, NOT) are provided in the Rule Designer to allow combining conditions together or to negate an expression used in a condition.

A third way to use conditions is when writing require or exclude rules about relationships. In the rule “item A requires Relationship B” the eConfigurator engine has no way to determine which items in Relationship B to add to the solution if the user picks item A. So when the user picks item A, the eConfigurator engine prints a message in the user’s message area stating that a selection from Class B is required.

Attribute Value (Advanced)

The Attribute Value (Advanced) template has the following form:

(for [any] items)

You can toggle between [any] and [all]. The Attribute Value (Advanced) template can be used only within a require rule, and it changes the logic for conditions involving attributes. This template does not display in the Pick a Rule list. Instead, it displays in the list for inserting a condition.

The following require rule contains two attribute conditions:

Attribute C = M in Relationship A requires Attribute D = P in Relationship B

This rule works as follows: If *any* item from Relationship A has Attribute C = M in the solution, then *all* the items from Relationship B must have Attribute D = P in the solution. This is the default behavior of the require template when it contains attribute conditions and is called the Any-All form.

By inserting the Attribute (Advanced) template into the rule, you can create all the other combinations of Any-All logic:

- Attribute C = M in Relationship A requires Attribute D = P (for any items) in Relationship B

  If any item from Relationship A has Attribute C = M in the solution, then there must be at least one item from Relationship B that has Attribute D = P in the solution (Any-Any form).

- Attribute C = M (for all items) in Relationship A requires Attribute D = P in Relationship B

  If all the items from Relationship A have Attribute C = M in the solution, then all the items from Relationship B must have Attribute D = P in the solution (All-All form).

- Attribute C = M (for all items) in Relationship A requires Attribute D = P (for any items) in Relationship B

  If all the items from Relationship A have Attribute C = M in the solution then there must be at least one item from Relationship B that has Attribute D = P in the solution (All-Any form).

To create this logic in other rule types, such as exclude rules, use one of the Advanced Rule Templates to create the rule. Then insert a numAttr condition in the rule.

Conditional Value

The Conditional Value Template has the following form:

([value] when [condition] is true, otherwise [value])

This template allows you constrain a value based on a condition. The [value] can be any number or item in the customizable product.

This template does not display in the Pick a Rule list. Instead, it displays in the lists for inserting arguments in a rule. You can insert the Conditional Value template anywhere you can insert a number. The most common use for this template is with provide and consume rules.

For example, you want product P1 to provide 2 to the resource R when the quantity of product P2 is greater than 10. If the quantity of P2 is not greater than 10, you want product P1 to provide 1 to resource R. You would write this constraint as follows:

P1 provides (2 when P2 > 10, otherwise 1) to R

If you wanted product P1 to provide 2 to resource R only when product P2 is greater than 10, you would write this constraint as follows:

P1 provides (2 when P2 > 10, otherwise 0) to R

Constrain

The Constrain template has the form:

Constrain [an expression] to be true

The constraint template is useful for making simple comparison or quantity expressions into top-level constraints. For example, you want make sure that there are exactly 4 of Item B in every solution:

Constrain [Item B = 4] to be true

You can also use the Constraint template to create exclude and require rules that have only one operand. For example the following rule excludes Item B from the solution:

Constrain [Item B =0] to be true

To require at least 1 Item B in the solution:

Constrain [Item B > = 1] to be true

By using Compound Logic operators, you can yoke conditions together and then allow or disallow the combination. For example, you want to make sure that if the quantity of Item A > 4 in the solution, then Item B must be less than 5, and vice versa:

Constrain [Item A > 4 AND Item B < 5] to be true

Constrain Attribute Conditions

The Constrain Attribute Conditions template has the following form:

An attribute] [=] [a value] [requires or excludes] [an attribute] [=] [a value]

This template allows you to constrain the selectable values for one attribute based on the value the user selects for another attribute. The requires and excludes operators in this template work the same way as those in the Require and the Exclude templates.

Use this template when attribute choices for one item affect allowable attribute choices for another item. For example, you sell shirts in small, medium and large sizes. The user picks the size by selecting a value for the Size attribute. These shirts come in red, green, and blue for small and medium sizes. Large size shirts come in red only. The user picks the color by selecting a value for the Color attribute.

Use this template to write rules that restrict the choices available to the user when they pick a Size or Color. For example, you would write a rule that both blue and green colors exclude the large size. If the user selects Size large, the blue and green attributes cannot be selected for Color. If the user selects Color blue, then large cannot be selected for Size.

Constrain Attribute Value

The Constrain Attribute Value template has the form:

[An attribute] [=] [a value]

The Constrain Attribute Value template sets the value of an attribute so that it cannot be overridden by the user during a configuration session. If you write a rule that sets the attribute equal to a value, this has the same effect as setting the attribute value and saving the record in Product Administration > More Info > Dynamic Attributes. By setting the comparison operator to other than equals (=), you can constrain the allowable ranges for numeric attribute values. For example you could write a rule that constrains an attribute value to > 100. In this fashion you can use the Constrain Attribute Value template to validate user input for range-of-values attribute domains.

Depending on the data type of the attribute domain, the attribute value can be set to one of the LOV choices, to the value of a linked item, the value of another item's attribute, to a string, or to a number.

You can use this template to restrict attribute values based on conditions that occur during a configuration session. For example, you could write a rule that restricts one attribute's value if the user chooses a specified value for another attribute.

This template cannot be used to constrain the attributes of customizable products that are components in a customizable product. For example, customizable product CP1 has as one of its components customizable product CP2. You cannot use this template to constrain the values of attributes in CP2.

If the product administrator has set the value of an attribute in the Dynamic Attributes list, this value cannot be overridden by a configuration rule, or by the eConfigurator engine.

Constrain Conditionally

The Constrain Conditionally template has the form:

> When [condition A is true] [enforce constraint B], otherwise [enforce constraint C]

This template provides if-then-else logic. When the condition is true the first expression is enforced as a constraint. If the condition is false, the second expression is enforced as a constraint.

Another way to view the logic is as a relationship between a condition and two constraints:

- If condition A is true, then B is enforced as a constraint, and C can be either true or false.
- If condition A is false, then C is enforced as a constraint, and C can be either true or false.

The condition can be defined as a quantity comparison of a product, relationship, or resource. It can also be the value of a linked item. Compound logic operators (AND, OR, and so on) are provided to link conditions together.

The constraints can also be quantity comparisons of products, relationships, or resources. The value of a linked item can also be used.

Constrain Product Quantity

The Constrain Product Quantity template has the form:

[The quantity of a product] [=] [a value]

The template has three parts. The first operand names the product.

The operator, [=], defines how the product quantity is related to the third operand, [a value]. The operator is limited to numeric comparisons (=, <, >, and so on).

The [a value] operand can be any of the following:

- The quantity of a product in the solution
- The quantity of items from another relationship in the solution
- The quantity of items in the solution from a class within a relationship
- The value of an attribute (the data type for the attribute must be number or integer)
- A number

The "Insert a" tab provides two sets of arithmetic functions that allow you to combine these items. For example, you could write the following rule:

The quantity of Item A from Items Class < = the quantity of Products-Class items

In this rule, "the quantity of item A from Items Class" is the first operand. It names Item A in the Items class. The second operand is "< =".

The third operand is "the quantity of Products-Class items" and refers to items in the Products class. This class is contained within a relationship in the customizable product.

The rule states that the quantity of Item A in the solution must be less than or equal to the number of items from the Products-Class in the solution.

Use Constrain Product Quantity rules to set limits on the quantity of products that can be in the solution. The limits can be defined as a number or as the quantity of other items, or the value of an attribute.

Constrain Relationship Quantity

The Constrain Relationship Quantity template has the form:

[The quantity of a relationship] [=] [a value]

The quantity of a relationship is the total number of items that have been added to the solution from the relationship. For example, a relationship contains Item A and Item B. If there is one Item A and one Item B in the solution, then the relationship quantity is two. If there are two of Item A in the solution and no Item Bs, the relationship quantity is also two.

The operator, [=], defines how the relationship quantity is related to the third operand, [a value]. The operator is limited to numeric comparisons (=, <, >, and so on).

The [a value] operand can be any of the following:

- The quantity of a product in the solution
- The quantity of items from another relationship in the solution
- The quantity of items in the solution from a class within a relationship
- The value of an attribute (the data type for the attribute must be number or integer)
- A number

The “Insert a” tab provides two sets of arithmetic functions that allow you to combine these items into expressions. For example, you could write the following rule:

The quantity of Items from Items relationship < = (2 * [Training Classes “Intro Course” Items-limit-attribute])

The first operand is the relationship: “The quantity of Items from Items relationship.” The operator is “ < = ”. The second operand is the “Items-limit-attribute.”

This rule states that the quantity of items from the Items relationship must be less than or equal to twice the value of the “Items-limit-attribute” of the Intro Course. The Intro Course is located in the Training Classes relationship.

Use Constrain Relationship Quantity rules to set limits on the quantity of products from a relationship that can be in the solution. The limits can be defined as a number or as the quantity of other items, or the value of an attribute.

Constrain Resource Value

The Constrain Resource Value template has the form:

[A resource] [>] [a value]

The template has three parts. The first operand names the resource.

The operator, [>], defines how the resource is related to the second operand, [a value]. The operator is limited to numeric comparisons (= , < , > , and so on).

The [a value] operand can be any of the following:

- The quantity of a product in the solution
- The quantity of items from another relationship in the solution
- The quantity of items in the solution from a class within a relationship
- The value of an attribute (the data type for the attribute must be number or integer)
- A number

The “Insert a” tab provides two sets of arithmetic functions that allow you to combine these items into expressions.

Use the Constrain Resource Value template to define limits on the value of a resource. For example, you can constrain a resource to be greater than 0 and less than 5.

Display Message

The Display Message template has the following form:

[Item or condition] displays this rule's explanation

The Display Message template is one of two ways to communicate messages to the user. The other is the Display Recommendation template.

In the Display Message template, when the condition is true, the explanation you entered for the rule is displayed to the user in the Message area. The explanation must describe the condition in the rule and can add additional information.

Use the Display Message template to display messages when a defined condition is true.

For example, you write the following Display Message rule:

When [Product A is selected], display this rule's explanation.

You then save the rule and enter the following in the Explanation field:

Product A has been selected. You can purchase only two of these items.

During a configuration session, when the user selects Product A, the rule explanation appears in the message area.

Display Recommendation

The Display Recommendation template has the following form:

> [Item or condition] recommends [item or condition] by displaying this rule's explanation.

The Display Recommendation template displays the rule's explanation based on the truth state of its two item/condition operands, as shown in Table 22.

Table 22. Truth Table for Display Recommendation Template

First Item/Condition	Second Item/Condition	Message Displays?
True	True	No
False	True	No
True	False	Yes
False	False	No

The table shows that the rule explanation displays only when the first item/condition is true and the second is false. The recommendation displays in the user's message area.

The Boolean equivalent of how the Display Recommendation template works is shown below. The expression is true only when A is true and B is false. When the expression is true, the message displays.

> NOT ((NOT A) OR B)

Use Display Recommendation rules to up-sell or cross-sell other configuration options, to inform the user of configuration restrictions, or to offer configuration suggestions.

For example, when the user picks Product A, you want to display a message recommending Product B. If the user then picks Product B, you want to stop displaying the message.

You would write this rule as follows:

Product A recommends Product B by displaying this rule's explanation.

In the rule's Explanation field: When you select Product A, we recommend you also purchase Product B.

Exclude

The Exclude template has the form:

[item or condition] excludes [item or condition]

The exclude rule is mutual. For example, if Item A is present in the solution, Item B cannot be present. Conversely, if Item B is present, Item A cannot be present.

The excludes operator is functionally equivalent to a Boolean NAND (NOT of A AND B). In Table 23, a T (true) means the item is present in the solution. An F (false) means it is not present or is excluded.

Table 23. Truth Table for Exclude

A	B	A AND B	A NAND B
T	F	False	True
F	T	False	True
F	F	False	True
T	T	True	False

The truth table shows that an exclude rule is always true except when both operands are present in the solution.

Use an exclude rule to:

- **Prevent technical configuration errors**. For example, a computer operating system or software application may be incompatible with certain microprocessors.
- **Prevent configurations that are undesirable or ineffective**. For example, in a financial model, an exclude rule could prevent adding a junk bond fund to a retirement portfolio.

Items

An item in an exclude rule can be any of the following:

- A relationship or class within a relationship
- A product within a relationship or class

Here is how the exclude rule works with items:

- **Product A excludes Product B**

  If Product A is present in the solution, then Product B cannot be present. If Product B is present in the solution, then Product A cannot be present.

- **Relationship A excludes Product B**

  If any product in Relationship A is present in the solution, then Product B is excluded. If Product B is present, then no product from Relationship A can be present.

- **Relationship A excludes Relationship B**

  If any Product in Relationship A is present in the solution, then no product in Relationship B can be present. If any product in Relationship B is present in the solution, then no product from Relationship A can be present.

Conditions

Conditions in an exclude rule can take many forms. For example, an attribute condition specifies an attribute value. Here is a summary of how the exclude rule works with conditions in general. How exclude rules work with specific types of conditions is covered in the sections following this one.

- **Product A excludes Condition B**

  If Product A is present in the solution, then Condition B cannot be true. If Condition B is true in the solution, then product A cannot be present.

- **Relationship A excludes Condition B**

  If any product in Relationship A is present, Condition B cannot be true. If Condition B is true in the solution, then no product from Relationship A can be present.

- **Condition A excludes Product B**

  If Condition A is true in the solution, then Product B is excluded. If Product B is present in the solution, then Condition A cannot be true in the solution.

- **Condition A excludes Relationship B**

  If Condition A is true in the solution, then all products in relationship B are excluded. If any product from Relationship B is present in the solution, then Condition A cannot be true in the solution.

- **Condition A excludes Condition B**

  If Condition A is true in the solution, then Condition B cannot be true in the solution. If Condition B is true in the solution, then Condition A cannot be true in the solution.

Attribute Conditions

Attribute conditions are used to exclude specific attribute values for an item or group of items. For example, if the user picks item A, then the "Large" attribute value for item B is excluded.

In the rule examples below, the attributes are defined on items in relationships within the customizable product. You can also define exclude rules on the attributes of the customizable product itself.

In the following rule examples, *excluded* means the user can no longer select the item. If the excluded item is a relationship, the user can no longer select any of the products in the relationship. Excluded also means the eConfigurator engine will not create solutions that contain the excluded item.

- **Product A excludes Attribute C = M in Relationship B**

  If Product A is present in the solution, then the value M will not be selectable for Attribute C in Relationship B.

  If any product with Attribute C = M in Relationship B is present in the solution, then Product A is excluded from the solution.

- **Relationship A excludes Attribute C = M in Relationship B**

  If any product in Relationship A is present in the solution, then the value M will not be selectable for Attribute C in Relationship B.

  If any product with Attribute C = M in Relationship B is present in the solution, then all the products in Relationship A are excluded from the solution.

- **Attribute C = M in Relationship A excludes Product B**

  If any product with Attribute C = M in Relationship A is present in the solution, then Product B is excluded from the solution.

  If Product B is present in the solution, then the value M will not be selectable for Attribute C in Relationship B.

- **Attribute C = M in Relationship A excludes Relationship B**

  If any product with Attribute C = M in Relationship A is present in the solution, then all the products in Relationship B are excluded from the solution.

  If any products from Relationship B are present in the solution, then the value M will not be selectable for Attribute C in Relationship B.

- **Attribute C = M in Relationship A excludes Attribute D = P in Relationship B**

  If any of the products with Attribute C = M in Relationship A are present in the solution, then the value P for Attribute D will not be selectable in Relationship B.

  If any products with Attribute D = P in Relationship B are present in the solution, then the value M will not be selectable for Attribute C in Relationship A.

Quantity Conditions

Quantity conditions compare the quantities of two items. Depending on the session context in which a quantity condition is evaluated, it either returns true/false or is enforced as a constraint.

For example you write the following configuration rule,

(Product A > Product B) excludes Product C

Context A. If the user picks Product A so that its quantity is greater than Product B, then Product C is excluded. In this case, the quantity condition is evaluated as true/false, and Product C is excluded when it is true.

Context B. If the quantity of Product A in the solution is not greater than Product B and the user picks Product C, then the quantity condition is enforced as a constraint. In all further solutions, the eConfigurator engine will require that the quantity of Product A is < = Product B if Product C is present.

Other Item Constraints

Several other types of conditions can also be used in exclude rules. In some cases these conditions do not make sense when used as the second operand in an exclude rule. Table 24 summarizes how to use item constraints.

Table 24. Item Constraint Usage in Exclude Rules

Item Constraint	First Operand	Second Operand
Attribute Value	Yes	Yes
Consumes	No	No
Provides	No	No
Excludes	Yes	Yes
Excludes List	Yes	Yes
Requires	Yes	Yes
Requires List	Yes	Yes
Linked Item Value condition	Yes	Yes
Message, Recommends	No	No
Product Quantity	Yes	Yes
Relationship Quantity	Yes	Yes
Class Quantity	Yes	Yes

Nested Expressions as Conditions

The Exclude Template can itself be used as a condition in other rules. The most common templates used for writing nested rules are the Exclude and Require templates. For example you could write the configuration rule:

Product A excludes (Product B excludes Product C)

The Boolean form of this rule is as follows:

A NAND (B NAND C)

In Table 25, a T (true) means the item is present in the solution. An F (false) means the item is not present or is excluded.

Table 25. A NAND (B NAND C)

Row	A	B	C	(B NAND C)	A NAND (B NAND C)
1	F	F	F	T	T
2	F	F	T	T	T
3	F	T	F	T	T
4	F	T	T	F	T
5	T	F	F	T	F
6	T	F	T	T	F
7	T	T	F	T	F
8	T	T	T	F	T

The truth table lets you analyze what happens when the user picks items. For example, there are no Product A, Product B, or Product C in the solution. The user picks Product B. You evaluate how the eConfigurator engine will respond as follows:

a Picking B means that B is true, so eliminate all rows from the table where B is False. This leaves rows 3, 4, 7, and 8.

b The engine returns solutions in which all constraints are true, so you can eliminate any of the remaining rows where the whole rule is false. This means you can eliminate row 7. This leaves rows 3, 4, and 8.

c Now examine the truth conditions for Product A and Product C in the rows 3. 4, and 8. The table shows that A is false in row 3 and 4 but true in row 8. This means that if the user picks B, then A can be either present or absent. It is not constrained. The table shows that C is false in row 3 but true in rows 4, and 8. This also means that C is not constrained. Thus, the user can pick B without A or C being excluded or required.

If A or C had been true in all three remaining rows, this means A or C is required. If A or C had been false in all three rows, this means A or C is excluded.

Multiple Operands

You can add multiple operands to an exclude rule by clicking the Compound Field button when you create the rule.

For example, you could create the rule:

Item A excludes Item B > 2, Item C < 5

This rule means the following:

- If Item A is present in the solution, Item B cannot be greater than 2
- If item A is present in the solution, Item C cannot be less than 5
- If Item B is greater than 2 in the solution, Item A cannot be present
- If item C is less than 5 in the solution, Item A cannot be present

Using commas to separate expressions is the same as writing two rules:

Item A excludes Item B > 2

Item A excludes Item C < 5

If you want to write a rule where you exclude the *combination* of two conditions you would do it as follows:

Item A excludes (Item B > 2 AND Item C < 5)

This rule means that if Item A is present in the solution, the two conditions cannot be simultaneously true in the same solution. If Item A is present, the quantity of Item B can be greater than 2 as long as the quantity of Item C is not less than 5 and vice versa.

Provide and Consume Templates

The Provide template has the form:

[An item] provides [a value] to [a target]

The Consume template has the form:

[An item] consumes [a value] from [a target]

Provide and consume rules positively or negatively increment the amount of the target operand each time the specified item is added to the solution. Provide rules contribute a positive amount, that is they increase the amount of the target. Consume rules contribute a negative amount, that is they reduce the amount of the target.

Contrast this with the behavior of require rules. For example, Item A requires Item B. The first time the user picks Item A, if no item B is in the solution, the eConfigurator engine will add at least one Item B. The second time the user picks Item A, the engine does not increment Item B because the require rule does not consider the quantity of Item A in the solution, only that Item A is present.

Now consider the rule Item A provides 1 to Item B. Each time the user picks Item A, the eConfigurator engine increments the number of Item B in the solution by 1. This rule ties the quantities of Item A and Item B together so that each Item A requires an Item B. Provide and consume rules work directly with quantities expressed as resource or attribute values, while require rules consider only the presence or absence of an item.

Item

The Item operand can be a product, a relationship, a product class within a relationship, a resource or an attribute. If the item is a relationship or class, the rule applies to all the items in the relationship or class. For example, Relationship A provides 1 to Item B. Each time an item from Relationship A is added to the solution, Item B is added to the solution.

Value

The value operand defines the quantity to be contributed to the target. The Rule Designer provides several methods for determining this quantity:

- You can explicitly state the quantity, for example Item A consumes 1 from Item B. This rule means that each Item A added to the solution decreases the amount of Item B in the solution by 1.
- You can define the value as the quantity of another item, or the value of an attribute, linked item or resource. For example, Item A provides three times the quantity of Item B to Item C. This rule means that for each Item A added to the solution, the quantity of Item C is incremented by three times the quantity of Item B.
- You can define an expression that determines which products in the relationship or class specified in Item will increment the target. For example: Any item of Relationship A provides Relationship A, Attribute Color = Red to Item B. This rule means that for each item in Relationship A for which the attribute Color = Red that is added to the solution, the quantity of Item B is incremented by one.

Target

The target operand is incremented by the amount specified in the value operand. The target can be a product, resource, or product attribute. It cannot be a relationship, a class, or an expression.

Product Target

When the target operand is a product, the quantity of the product is incremented. For example, Product A provides 2 to Product B. This rule means that each Product A added to the solution increases the quantity of Product B (the target) by 2.

The consume rule works the same way. For example, Product A consumes 2 from Product B. This rule means that each Product A added to the solution decreases the quantity of Product B (the target) by 2.

Resource Target

When the target is a resource, the value of the resource is incremented. One of the most common uses of provides and consume rules is to manage resources.

Resources keep track of configuration variables that increase or decrease as the user makes selections. For example, suppose you are creating a customizable product for configuring desktop computers. Your product includes several types of chassis. Each chassis has a different number of slots for expansion cards. The product also includes several types of expansion cards, such as disk controllers, and graphics cards.

You don't know in advance which chassis the customer will select or how many expansion cards. However, you do know you must keep track of the number of slots available in a chassis during the configuration process to verify that the computer is configured correctly.

Using provide and consume rules to increment a resource is the way to handle this:

- First define a resource to keep track of slots, for example Slots Available.
- Then define an attribute called Slots on the chassis class. Create a list of values of data type integer. Create one record for each chassis type. The value for each record is an integer equal to the number of slots in the chassis type. This creates a menu of choices for setting the number of slots in a chassis. Assign the list of values to the Slots attribute definition.
- Display the attributes for each chassis and set the value of the Slots attribute and save the record. This sets the number of slots in the chassis and prevents it from being changed by the user or the eConfigurator engine.
- Add the chassis class to a relationship, Chassis.
- Define an attribute called Slots Required on the expansion card class. Use a range of values domain and set the data type to Integer. Enter: = =1 as the validation expression. Enter 1 as the default value.
- Display the attributes for each expansion card and save the Slots Required record. This sets the number of slots required at 1 and prevents it from being changed by the user or the eConfigurator engine.
- Add the expansion card class to a relationship, Expansion Card.

- Write the following rule: Chassis provides Chassis Slots to Slots Available.
- Write the following rule: Expansion Card consumes Expansion Card Attribute = Slots Required from Slots Available.
- Write rules as needed to determine what happens if Slots Available is 0 or if it becomes negative.

When the user selects a chassis, the provide rule increases the Slots Available resource by the number of slots in the chassis. Each time the user selects an expansion card, the consume rule decreases the Slots Available resource by one. Thus, the Slots Available resource maintains a record of how many slots are available in the chassis during the configuration session.

If a resource has the same name in two different customizable products, the eConfigurator engines treats them as the same resource. You can take advantage of this in cases where one customizable product is contained within another. For example, customizable product CP2 is contained within customizable product CP1. You define resource R1 in both products. Rules in either customizable product that contribute to R1, affect the value of R1 in both products. Use this behavior to allow a parent customizable product to contribute to a resource in a child customizable product.

Attribute Target

When the target is an attribute, the value of the attribute is incremented. Attribute targets are very similar in behavior and use as resource targets. There are several restrictions on using provide and consume rules to manipulate attribute values:

- The data type of the attribute must be numeric (Integer or Number).
- The attribute must be available for manipulation. You must not have set the value of the attribute. You do this by selecting an attribute value and saving it in Product administration > Dynamic Attributes.
- A child customizable product can contribute to attributes defined on the parent. The parent cannot contribute to attributes defined on the child product. For example, customizable product CP2 is contained within customizable product CP1. CP2 can contribute to attributes defined within CP1. CP1 cannot contribute to attributes defined within CP2.

Use attributes as targets instead of defining multiple resources that keep track of similar variables. This ties the variables directly to a class and makes it easier to keep track of the variables' roles.

Provide and Consume, Simple

The Simple Provide template has the form:

Provides [a value] to [a target]

The Simple Consume template has the form:

Consume [a value] from [a target]

The Simple Provide and Simple Consume templates positively or negatively increment the amount of the target operand. These templates are intended for use as the action portion of a conditional rule. If the condition is true, then a value is provided or consumed from the specified target.

The Simple Provide template contributes a positive amount, that is it increases the amount of the target. The Simple Consume template contributes a negative amount, that is it reduces the amount of the target.

Value

The value operand defines the quantity to be contributed to the target. The Rule Designer provides several methods for determining this quantity:

- You can explicitly state a number as the value.
- You can define the value as the quantity of another item, or the value of an attribute, linked item or resource. For example, you can provide three times the quantity of Item B to Item C.
- The value can be an expression that resolves to an amount. This amount is then contributed to the target. For example: When a condition is true, it provides Relationship A, Attribute Color = Red to Item B. This rule means that when the condition is true, then each item in Relationship A for which the attribute Color = Red that is added to the solution, increments the quantity of Item B by one.

Target

The target operand is incremented by the amount specified in the value operand. The target can be a product, resource, or product attribute. It cannot be a relationship, a class, or an expression.

Product Target

When the target operand is a product, the quantity of the product is incremented in the solution.

Resource Target

When the target is a resource, the value of the resource is incremented.

Attribute Target

When the target is an attribute, the value of the attribute is incremented. Attribute targets are similar in behavior and use as resource targets. There are several restrictions on using provide and consume rules to manipulate attribute values:

- The data type of the attribute must be numeric (Integer or Number).
- The attribute must be available for manipulation. You must not have set the value of the attribute. You set the value of an attribute by selecting an attribute value and saving it in Product administration > Dynamic Attributes.
- A child customizable product can contribute to attributes defined on the parent. The parent cannot contribute to attributes defined on the child product. For example, customizable product CP2 is contained within customizable product CP1. CP2 can contribute to attributes defined within CP1. CP1 cannot contribute to attributes defined within CP2.

Relationship Item Constraint

The Relationship Item Constraint template has the form:

For each item [in a relationship], constrain [an expression] to be true

The “in a relationship” operand can be a whole relationship, a subclass of items in a relationship, or a product in a relationship. The “an expression” operand can be any rule template or any rule you construct from templates.

The purpose of the Relationship Item Constraint template is to allow you to write a rule for items in a relationship as if you had written the rule separately for each instance of the items. For example, you define Relationship A that contains the customizable product desktop PC. The desktop PC is a customizable product that includes two relationships: CPU and Hard Drive. You then write the following rule:

For each item in Relationship A, constrain CPU requires Hard Drive to be true

This rule enforces “CPU requires Hard Drive” separately on each instance of desktop PC in Relationship A. All the desktop PCs from Relationship A must have a hard drive if they have a CPU.

A require rule does not do this. Suppose you had written the following rule:

CPU requires Hard Drive

This rule means if any desktop PC has a CPU from the CPU relationship then at least one desktop PC must have a hard drive from the Hard Drive relationship.

This means, for example, that if the user configures three desktop PCs, all with CPUs, then only one of them must have a hard drive. If the user removes the hard drive, the eConfigurator engine would add a hard drive to another desktop PC in the solution or add a new desktop PC that contains only a hard drive. The require rule defines a constraint that is true about the group of desktop PCs in the solution rather than about individual desktop PCs.

Another problem with the require rule is that it does not limit enforcement of the constraint to the items in Relationship A. If, in the require rule example, desktop PCs were also contained in Relationship B, then desktop PCs configured from Relationship B would also be considered when enforcing the require rule for desktop PCs configured from Relationship A.

An important use of this template is to write rules that apply to customizable products only when these products are contained in a relationship within another customizable product.

Require

The Require template has the form:

[item or condition] requires [item or condition]

A require rule is a logical implies. If the first operand is true then the second operand is implied (must be true). For example, Item A requires Item B. This rule means that if Item A is present in the solution, then Item B must be present. Another example: Condition A requires Item B. This rule means that if Condition A is true, then Item B must be present in the solution.

A require rule is not mutual. The rule Item A requires Item B does not imply Item B requires Item A. However, if Item B is excluded then Item A is also excluded. This is because when Item B is excluded, the rule can never be true.

The require operator is functionally equivalent to the following Boolean expression:

(NOT A) OR B

In the first two columns of Table 26, a 1 means the item is present in the solution. A 0 means it is absent or excluded.

Table 26. Truth Table for RequireExpressions

A	B	NOT A	(NOT A) OR B
1	0	0	False
0	1	1	True
0	0	1	True
1	1	0	True

The table shows that a require rule behaves as follows:

- False when Item A is present and Item B is not. (If Item B cannot be present, Item A cannot be present).
- True when Item B is present and Item A is not.
- True when neither is present.
- True when both are present.

Use require rules to:

- To create a requires relationship for items in different relationships. For example, you can write a rule that if Item A in relationship 1 is picked, then Item B in relationship 2 is required.
- To add items to the configuration if a condition is true.
- To create relationships between other items (conditions) when a product is added to the solution.

Items

An item can be any of the following:

- A relationship or class within a relationship. For example, Relationship A requires Item B. This rule means that when any product in relationship A is present in the solution, then Item B must be present.
- A product within a relationship

When items are the operands, require rules are concerned only with presence or absence, not quantity. For example, Item A requires Item B. When the first Item A is added to the solution, the eConfigurator engine will add at least one Item B if none are present. When the second Item A is added, the engine does not add any more Item B, since at least one is already present.

Here is how the require rule works with items:

- **Product A requires Product B**

  If Product A is present in the solution, then Product B is required. If Product B is excluded, then Product A is excluded.

- **Relationship A requires Product B**

  If any product in Relationship A is present in the solution, then Product B is required. If Product B is excluded, then all the products in Relationship A are excluded.

- **Relationship A requires Relationship B**

  If any Product in Relationship A is present in the solution, then at least one product in Relationship B must be present. If all the products in Relationship B are excluded, then all the products in Relationship A are excluded.

Conditions

Conditions can take many forms. Here is a summary of how the require rule works with conditions in general. How require rules work with specific types of conditions is covered in the sections following this one.

- **Product A requires Condition B**

  If Product A is present in the solution, then Condition B is required to be true. If Condition B is false, then Product A is excluded.

- **Relationship A requires Condition B**

  If any product in Relationship A is present in the solution, Condition B is required to be true. If Condition B is false, then all the products in Relationship A are excluded.

- **Condition A requires Product B**

  If Condition A is true in the solution, then Product B is required. If Product B is excluded, then Condition A is required to be false.

- **Condition A requires Relationship B**

  If Condition A is true in the solution, then at least one product in relationship B must be present in the solution. If all the products in Relationship B are excluded, then Condition A is required to be false.

- **Condition A requires Condition B**

  If Condition A is true in the solution, then Condition B must also be true. If Condition B is false, then Condition A is required to be false.

Attribute Conditions

An attribute condition specifies an attribute value and uses it to identify the items in a relationship to which the rule applies. In the rule examples below, the attributes are defined on items in relationships within the customizable product. You can also define require rules on the attributes of the customizable product itself.

In the rules examples below, the equals operator is used in the attribute expressions. You can use all the math operators (<, >, and so on) when writing this type of rule.

- **Product A requires Attribute C = M in Relationship B**

  If Product A is present in the solution, then M is the only value selectable for Attribute C for all items in Relationship B.

- **Relationship A requires Attribute C = M in Relationship B**

  If any product in Relationship A is present in the solution, then M is the only value selectable for Attribute C for all items in Relationship B.

- **Attribute C = M in Relationship A requires Product B**

  If any product with Attribute C = M in Relationship A is present in the solution, then Product B must be present.

- **Attribute C = M in Relationship A requires Relationship B**

  If any product with Attribute C = M in Relationship A is present in the solution, then at least one of the products in Relationship B must be present.

- **Attribute C = M in Relationship A requires Attribute D = P in Relationship B**

  If any of the products with Attribute C = M in Relationship A are present in the solution, then P is the only value selectable for Attribute D in Relationship B.

Quantity Conditions

Quantity conditions compare the quantities of two items. Depending on the session context in which the quantity condition is evaluated, it either returns true/false or is enforced as a constraint.

For example you write the following rule:

(Product A > Product B) requires Product C

If the user picks Product A so that its quantity is greater than Product B, then Product C is required. In this case, the quantity condition is evaluated as true/false, and Product C is required when it is true.

Contrast this with the following rule:

Product C requires (Product A > Product B)

When the user picks Product C, the condition (Product A > Product B) is enforced as a constraint. In all further solutions, the quantity of Product A must be greater than the Product B. Also, if the quantity of Product B is greater than Product A in the solution, Product C is excluded.

Other Item Constraints

Several other types of conditions can also be used in require rules. In some cases these conditions do not make sense when used as the second operand in a require rule. This is because some conditions are always true (message conditions), or the condition cannot be enforced to be true (linked item conditions).

Table 27 summarizes how to use item constraints.

Table 27. Item Constraint Usage in Require Rules

Item Constraint	First Operand	Second Operand
Attribute Value	Yes	Yes
Consume	No	Yes
Provide	No	Yes
Exclude	Yes	Yes
Exclude List	Yes	Yes
Require	Yes	Yes
Require List	Yes	Yes
Linked Item Value	Yes	No
Message, Recommends	No	Yes
Product Quantity	Yes	Yes
Relationship Quantity	Yes	Yes
Class Quantity	Yes	Yes

Nested Expressions as Conditions

The Require template can itself be used as a condition in other rules. The most common templates used for writing nested rules are the Require and Exclude templates. For example, you could write the following configuration rule:

Product A requires (Product B requires Product C)

The Boolean form of this rule is as follows:

(NOT A) OR ((NOT B) OR C)

Table 28 shows the truth table for this rule.

Table 28. (NOT A) OR ((NOT B) OR C)

Row	A	B	C	NOT A	(NOT B) OR C	(NOT A) OR ((NOT B OR C)
1	F	F	F	T	T	T
2	F	F	T	T	T	T
3	F	T	F	T	F	T
4	F	T	T	T	T	T
5	T	F	F	F	T	T
6	T	F	T	F	T	T
7	T	T	F	F	F	F
8	T	T	T	F	T	T

The table lets you analyze what happens when the user picks items. Use the following steps to do this:

a The engine must return solutions in which all constraints are true. Eliminate any rows where the top-level expression is False. In the table above, eliminate row 7.

b To determine what happens when the user picks and item, look at all rows that list true for that item. For example, to analyze what happens when the user picks Product A, you would look at rows 5, 6, and 8 in the table above.

c To determine what happens when a combination of items are picked, look at all the rows that list true for both items at once. For example, to analyze what happens when the user picks both Product A and Product B, you would look at only row 8 in the table above. (Row 7 is not considered since the top-level expression is false.)

d If only one row has the correct truth conditions, this means the eConfigurator engine will return the result shown in that row. For example, look at row 8. This is the only remaining row that lists item A and B as both true. Since this row lists C as true, this means that when both A and B are present, C must be present.

e If several rows have the condition you are analyzing, look at the truth conditions for each unpicked item in the rows. If they are all true, the unpicked item is required. If they are all false, the unpicked item is excluded. If an unpicked item lists true in some rows and false in others, this means the unpicked item is neither excluded nor required and is available.

The table reveals that the rule has the following behavior:

- When none of the products are present, the user can pick any of the three, and the other two will not be required.
- If the user picks Product A and B, then Product C is required.

Thus, the rule's behavior can be summarized this way: when the user picks Product A, the condition "Product B requires Product C" is enforced as a constraint.

Multiple Operands

You can add multiple operands to a require rule by clicking the Compound Field button when you create the rule.

For example, you could create the rule:

Item A requires Item B > 2, Item C < 5

This rule means the following:

- If Item A is present in the solution, the quantity of Item B must be greater than 2
- If item A is present in the solution, the quantity of Item C must be less than 5
- If Item B cannot be greater than 2 in the solution, Item A is excluded
- If item C cannot be less than 5 in the solution, Item A is excluded

This is the same as writing two rules:

Item A requires Item B > 2

Item A requires Item C < 5

Require (Mutual)

The Require (Mutual) template has the following form:

[Item or Condition] mutually requires [Item or Condition]

Use this template when the requires relationship between items or conditions is mutual. For example, Product A requires Product B, AND Product B requires Product A.

When using components as the operands, you can specify global paths for either or both components.

The Boolean equivalent of the Require (Mutual) template is NOT(A XOR B). Table 29 shows the truth table for this template.

Table 29. Truth Table for Require (Mutual)

A	B	A XOR B	NOT(A XOR B)
T	F	True	False
F	T	True	False
F	F	False	True
T	T	False	True

The behavior of the Require (Mutual) template is the same as the Exclude template, except that operands are required instead of excluded.

Set Initial Attribute Value

The Set Initial Attribute Value template has the form:

[An attribute] has an initial value of [a number]

The Set Initial Attribute Value template sets the numeric value of an item's attribute at the beginning of a configuration session. The attribute must be of data type Number or Integer. The attribute can have either an LOV or range of values type domain.

Setting an attribute value in this fashion brings the attribute value under the control of all its contributors. This means that the attribute value must exactly equal the sum of all its contributors. For example, if during the configuration session, the amount contributed by provide and consume rules exactly equals the attribute value, users will not be able to change the attribute value. If not, the user can adjust the value, but only if this also adjusts the amount contributed by the provide and consume rules.

To set the initial value of an attribute that has a non-numeric data type, use the Set Preference template.

This template cannot be used to set the attributes of customizable products that are components in a customizable product. For example, customizable product CP1 has as one of its components customizable product CP2. You cannot use this template to set the values of attributes in CP2.

If the product administrator has set the value of an attribute in the Dynamic Attributes list, this value cannot be overridden by a configuration rule or by the eConfigurator engine.

Set Initial Resource Value

The Set Initial Resource Value template has the form:

[A resource] has an initial value of [a number]

The Set Initial resource Value template functions as a provide rule and sets the numeric value of a resource at the beginning of a configuration session. The resource must be of data type Number or Integer. The resource can have either an LOV or range of values type domain.

Before specifying a resource using this template, items must be present in the solution that contribute (provide or consume) to the resource. During a configuration session, other rules can increase or decrease the value.

Resource values are under the exclusive control of provide and consume rules. Users cannot set the value of a resource by entering a value in a selection page.

Set Preference

The Set Preference template has the form:

> When possible, constrain [an expression] to be true with a [specified priority]

The expression in a preference rule is enforced as a constraint only if it does not conflict with any other rule type or with any user selections. The purpose of the Set Preference template is to allow you to create soft constraints that guide the eConfigurator engine in producing solutions but which the engine can ignore if needed to avoid conflicts or performance problems.

A key use for preference rules is to cause a default selection of Item B based on the selection of Item A. This is called a dynamic default. You can set a default dynamically based on a previous user selection. The user can then override the default if desired by making choosing a different item than the dynamic default.

For example, you could write the following rule:

> When possible, constrain [Item A requires Item B] to be true with a priority of [0].

When the user picks Item A, the engine will attempt to create a solution containing Item B. However, the engine is free not to include Item B in order to avoid conflicts and performance problems.

If the user does not want Item B, they can remove it without creating a conflict. If you had written the rule as "Item A requires Item B", Item B would be added when the user picks Item A. If the user tries to remove Item B, they would receive a conflict message.

Another use for the Set Preference Template is to set or modify the default value for an attribute. To do this, you would write a preference rule where the expression, is Attribute A = value. The attribute would then be displayed with this value unless overridden by another rule.

The priority operand in preference rules determines the order in which multiple preference rules for an item are evaluated. Preference rules with priority 0 that apply to a specific item are evaluated first. Those with priority 1 that apply to that item are evaluated next, and so on.

For example, you have written two preference rules that apply to a specific relationship. PrefRule A has a priority of 0. PrefRule B has a priority of 1. The eConfigurator engine will attempt to add PrefRule A to the solution before attempting to add PrefRule B.

Here's how the eConfigurator engine creates solutions containing preference rules:

- The engine generates a solution that enforces all constraints and user choices but does not include preference rules.
- The engine then adds the preference rules one at a time for each item. It begins by adding the highest priority preference rules for an item first. Preference rules with the same priority are added in arbitrary order.
- If the solution remains valid when a preference rule is added, then the expression in the preference rule is enforced as a constraint and becomes part of the solution.
- If adding a preference rule creates a conflict so that the solution fails, the preference rule is skipped, and its expression is not enforced. The engine then goes on to the next preference rule.
- Preference rules are evaluated after all other rule types and before the engine searches for and sets default attribute values.

Compound Logic and Comparison Operators

Both Compound Logic and Comparison operators test for the truth of their operands. They return a true or false rather than a quantity.

Compound Logic operators, such as AND, NOT, OR, are used to link expressions together when creating a rule. For example: (Condition A AND Condition B) requires Item C. Compound Logic operators are also called Boolean operators.

Table 30 presents the Compound Logic operators you can use with rule templates.

Table 30. Compound Logic Operators

Operator	Example	Description
Not	`NOT A`	Logical negation. True when *A* is false and false when *A* is true. *A* can be an item or sub-expression.
And	`A AND B`	Both *A* and *B*. True only when both *A* and *B* are true. When used as a top-level constraint, means that only solutions where both *A* and *B* are true are allowed. *A* and *B* can be items or sub-expressions.
Or	`A OR B`	Either *A* or *B* or both. False only when both *A* and *B* are false. *A*and *B* can be items or sub-expressions.
Exclusive Or	`A XOR B`	*A* or *B* but not both. *A* and *B* must have opposite truth states. False when *A* and *B* are either both true or both false. *A* and *B* can be items or sub-expressions.
NAND	`NOT (A AND B)`	Converse of A AND B. False only when both A and B are true. When used as a top-level constraint, means that A and B cannot both be present. A and B can be items or sub-expressions.

Comparison operators compare their operands and return a true or false. In the following rule, when the quantity of item A is less than item B (when the comparison is true), then item C is required.

(Item A < Item B) requires C

If you specify an item as an operand in a comparison, the quantity of the item in the solution is used to make the comparison. If you specify an expression as an operand, the expression must resolve to a number.

If you specify an expression that resolves to true or false, then true is assigned the value 1 and false is assigned the value 0.

Table 31 presents the Comparison operators you can use with rule templates.

Table 31. Comparison Operators

Operator	Example	Description
Greater than	`A > B`	*A* is greater than *B*
Not less than	`A >= B`	*A* is greater than or equal to *B*
Equals	`A == B`	*A* equals *B*
Equals (compound)	A = = B = = C = = D	True if A = B AND A = C And A = D
Not equal to	`A <> B`	*A* does not equal *B*
Not greater than	`A <= B`	*A* is less than or equal to *B*
Less than	`A < B`	*A* is less than *B*

When you are building a rule, you can compound the comparison operators. For example, you could build the following expression:

```
(A>B>C>D)
```

This expression is equivalent to the following expression:

```
A>B AND A>C AND A>D
```

Arithmetic Operators

An arithmetic operator allows you to perform an arithmetic operation on two items. If an operand is a product, the value refers to the quantity of the product in the solution.

The operands in the expression can be two items or can be one item and a constant. For example, you can increase the quantity of an item by a constant amount.

If you specify an expression as one of the items, it must resolve to a quantity. If the expression resolves to true or false, then 1 is assigned to true, and 0 to false.

Results of calculations are handled differently for resources values than for product quantities. Calculation results for resources are expressed exactly, including a decimal point if necessary. Because product quantities represent discrete units, results involving them are rounded to the nearest integer.

Table 32 shows the arithmetic operators.

Table 32. Arithmetic Operators

Operator	Example	Description
Addition	`A + B`	Sum of *A* and *B*. *A* and *B* can be items or expressions. Result is floating point if A or B is floating point.
Subtraction	`A - B`	Subtracts *B* from *A*. *A* and *B* can be items or expressions. Result is floating point if *A* or *B* is floating point.
Negation	`-(A)`	Additive inverse of *A*. Uses only one operand. *A* can be an item or expression.*A*
Multiplication	`A * B`	Product of *A* and *B*. Result is floating point if *A* or *B* is floating point. *A* and *B* can be items or expressions.
Division	`A / B`	Quotient of *A* divided by *B*. Truncates ratio to integer if both *A* and *B* are integers. Result is floating point if *A* or *B* is floating point. *A* and *B* can be items or expressions.
Modulo	`%(A, B)`	Remainder of *A* divided by *B*. For example, %(1900, 72) results in 28. If *A* or *B* is floating point, the value is first rounded to the nearest integer; then the remainder is computed as for integers. *A* and *B* can be items or expressions.

Table 32. Arithmetic Operators

Operator	Example	Description
Minimum	`min(A, B)`	Result is the smaller of *A* and *B* and is floating point if *A* or *B* is floating point. *A* and *B* can be items or expressions.
Maximum	`max(A, B)`	Result is the larger of *A* and *B* and is floating point if *A* or *B* is floating point. *A* and *B* can be items or expressions.

Table 33 shows additional arithmetic operators that also take numeric operands and produce numeric results. Use them to control numeric accuracy or change numeric characteristics.

Table 33. Additional Arithmetic Operators

Operator	Example	How Used
Integer	`int(A)`	Truncates *A* down to an integer. For example, if operand is 6.7, returns 6. *A* can be an item or expression. Useful only with properties.
Float	`flo(A)`	Converts *A* to floating point. Same as multiplying operand by 1.0. *A* can be an expression.
Absolute value	`abs(A)`	Returns the absolute value of *A*. *A* can be an item or expression.
Sign test	`sgn(A)`	Returns -1 if the quantity of $A < 0$, 0 if $A = 0$, 1 if $A > 0$. *A* can be an expression.

Configuration Rule Assembly Language 19

This chapter explains how to create rules using Rule Assembly Language in the Rule Designer. You can use Rule Assembly Language to enter rules instead of using rule templates.

Why Use Rule Assembly Language?

Rule Assembly Language (RAL) is intended for those users who are more comfortable working in a programming environment rather than using templates. Those with experience using this language in previous releases can continue to use it in this release.

In many cases, a combination of RAL and rule templates can be effectively employed as follows.

a Use the existing templates to create basic configuration rules.

b Create specialized templates and use them to create rules to handle configuration areas that are similar.

c Use RAL to create highly complex or unusual rules not easily handled by templates.

Understanding Rule Assembly Language

All rules in the Rule Assembly Language (RAL) consist of expressions. Expressions consist of an operator and its operands. The number and type of operands depend on the operator. All expressions have the following form:

```
operator(A,B...)
```

For example, the following expression evaluates to the sum of *A* plus *B*.

```
+(A,B)
```

Most operators allow their operands to be expressions. In the expression above, both *A* and *B* can themselves be expressions.

Spaces, tabs, and new-lines are ignored in expressions.

In Rule Assembly Language, a rule is a list of one or more top-level expressions. A top-level expression is the top-level operator and its associated operands in RAL statements. Rules are constraints.

For example, in the following statement, `+(A,B)` is a sub-expression and is not a top-level expression. The top-level expression is "==" and its operands. So the rule or constraint on all solutions is that the sum of the quantities of A and B must equal the quantity of C in all solutions.

```
==(+(A,B),C)
```

A sub-expression is an expression that functions as an operand. Depending on the operator, a sub-expression returns a quantity or a logical true or false. The sub-expression itself is not a rule or constraint.

Creating Rules Using the Assisted Advanced Rule Template

A special rule template is provided for creating rules using Rule Assembly Language. This template provides a dialog box for picking components, resources, and links. It also provides a list of RAL operators.

When you create a rule and save it using this template, the Rule Designer displays the rule syntax in the Rule field. The Rule Designer capitalizes the first letter of operator names in the rule for display purposes only. Operator names are case-sensitive, and the Rule Designer stores them in the correct format.

To create a rule using the Assisted Advanced Rule template

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.

   If you omit this step, the most recently released version of the customizable product is loaded in the Rule Designer, and you cannot create rules.
3. Click the Configuration Designer tab.

   The Rule Designer Rules List appears. It lists all the configuration rules that have been created for this customizable product.
4. Click New Rule.

   The “Pick a rule” tab appears and lists the rule templates available for creating rules.
5. Select the Assisted Advanced Rule template in “Pick a rule”, and click Continue.

   The Rule Statement and “Insert a” tabs appear.
6. Select operators and arguments from the displayed lists to build a rule.

   The operators list contains all the operators in the Rule Assembly Language. The arguments list changes depending on the operator you select and contains all the items in the customizable product. Use the Compound Field button to create sub-expressions.

7 Click Save Rule to save the rule.

The Save Rule form appears.

8 Fill out the fields in the Save Rule form. Then click Save. All fields are optional.

- **Name**. Enter a name for the rule. Consider using a naming convention that helps you easily identify and sort rules. Rule names must be unique. If you do not enter a name, this field remains blank in the Rule Designer list.
- **Explanation**. Enter a description of the rule. The description must contain the operator names used in the rule. If you leave this field blank, the Rule Statement in the form is displayed in this field in the Rule Designer list.
- **Start Date/End Date**. Click the Calendar select button and choose the date when you want the rule to become effective (Start Date), the date you want the rule to stop being effective (End Date), or both. Review the section on setting effective dates for rules for more information. Use a consistent approach to setting start and end dates. This makes troubleshooting product behavior problems easier.
- **Inactive**. Click the Inactive check box if you want the rule to be inactive. The rule remains part of the product but will not be used to compute solutions. To activate a rule for a product that has been released, you must change the rule's status to active, save the rule, and release the product again. To activate the rule in the current work space, you must change the rule's status to active and save the rule.

After you click Save, the Rules List appears. The list contains the new rule.

9 Open the Rules List menu and click Validate.

This starts a configuration session. Verify that the new rule works correctly.

Creating Rules Using the Advanced Rule Template

The Advanced Rule template is similar to the Assisted Advanced Rule Template. You can create rules using Rule Assembly Language with either template. The Advanced Rule template does not provide a dialog box for picking products, resources, or links. It also does not provide a list of operators.

The Advanced Rule Template is intended primarily for upgrade users who want to edit the rules in models created in release 6.x.

You must manually enter the path to items when using this template. Table 34 provides examples of paths.

When you create a rule and save it using this template, the Rule Designer displays the rule syntax in the Rule field. The Rule Designer capitalizes the first letter of operator names in the rule for display purposes only. Operator names are case-sensitive, and the Rule Designer stores them in the correct, format.

Table 34. Examples of Paths

Path	Explanation
@.[Relationship A]([Product SubClass])	All the products in SubClass in Relationship A.
@.[Relationship A]([Product 1]).[Color]	The color attribute of all the instances of Product 1 in Relationship A.
@.[Relationship A].[Color]	The Color attribute of all the products in Relationship 1.
$.[Resource 1]	Resource 1.
$.[Link 1]	Link 1.

Observe the following guidelines when writing paths:

- The @ sign specifies the instance of the product or group of products on which the rule is defined. Use the @ sign at the beginning of paths that refer to items inside the structure of a customizable product.
- The $ refers to a special object called the basket that is associated with each customizable product. This object maintains a non-hierarchical, flat view of the whole customizable product. Use the $ at the beginning of a path to specify links and resources. Since links and resources are defined for the whole customizable product, they are stored in the basket.

  Upgrade Users. Users upgrading from release 6.x will have rules containing paths that include syntax such as $.[product]. These paths should function normally. However, you should avoid using this syntax to create new rules. This syntax can cause solutions that contain unintended instances of products.
- Use periods (.) to specify the next property in the path. A property can be a relationship name or attribute name. Do not put spaces before or after the period.
- Use parentheses immediately after a relationship name to specify a subset of the items in a relationship. Parentheses act as a filter. The most common use for parentheses is to specify a subclass within the relationship. Do not put a period before the parentheses. For example @.[P]([X]).[Color] refers to the Color attribute of all the products in subclass X within relationship P.
- Enter names exactly as they are displayed in the lists where they were defined. Do not use display names. You can use subclass names to filter products within a relationship even though the subclass names do not display in the Product Designer.
- The path syntax always refers to the actual set of items specified in the path, however many instances are present. For example, @.[X]. [Color] refers to the color attribute of all the instances of products actually present in relationship X in a given configuration. A path only refers to actual instances, not the possible instances as defined by cardinality settings.
- If you create a rule containing a path to location that contains no items, the rule is ignored until items are present.

- All paths you specify in rules must be unique and unambiguous. For example, if you have multiple relationships with the same name, paths to them may be ambiguous. If this occurs, use the Assisted Advanced Rule template to write rules. This template uses underlying, unique identifiers to identify items.
- The maximum length of a rule is 900 characters. The UI allows rules that are longer, but they cannot be stored in the database. Using Siebel Tools, you can revise the UI to enforce the 900 character limit. See "Enforcing the Field Length for Entering Advanced Rules" on page 461.

When you create a rule, a record describing the rule displays in the Rule Manager Rules List. This record has the following fields:

Name. Enter a name for the rule. Consider using a naming convention that helps you easily identify and sort rules. Rule names must be unique. If you do not enter a name, this field remains blank in the Rule Designer list.

Explanation. Enter a description of the rule. The description must contain the operator names used in the rule. If you leave this field blank, the Rule Statement in the form is displayed in this field in the Rule Designer list.

Start Date/End Date. Click the Calendar select button and choose the date when you want the rule to become effective (Start Date), the date you want the rule to stop being effective (End Date), or both. Review the section on setting effective dates for rules for more information. Use a consistent approach to setting start and end dates. This makes troubleshooting product behavior problems easier.

Inactive. Click the Inactive check box if you want the rule to be inactive. The rule remains part of the product but will not be used to compute solutions. To activate a rule for a product that has been released, you must change the rule's status to active, save the rule, and release the product again. To activate the rule in the current work space, you must change the rule's status to active and save the rule.

To create a rule using the Advanced Rule template

1 Navigate to Product Administration.

2 Select and lock the desired customizable product.

If you omit this step, the most recently released version of the customizable product is loaded in the Rule Designer, and you cannot create rules.

3 Click the Configuration Designer tab.

The Rule Designer Rules List appears. It lists all the configuration rules that have been created for this customizable product.

4 Click New Rule.

The "Pick a rule" tab appears and lists the rule templates available for creating rules.

5 Select the Advanced Rule template in "Pick a rule", and click Continue.

The Rule Statement and "Insert a" tabs appear. The Rule Statement tab contains the rule template you selected.

To exit and return to the Rules List, click Cancel.

6 Enter the configuration rule into the template using Rule Assembly Language.

7 Click Save Rule to save the rule.

The Save Rule form appears.

8 Fill out the fields in the Save Rule form. Then click Save. All fields are optional.

After you click Save, the Rules List appears. The list contains a record for the new rule.

9 Open the Rules List menu and click Validate.

This starts a configuration session. Verify that the new rule works correctly.

Managing Rules Written in Rule Assembly Language

Use the same procedures for copying, editing, and deleting rules written in Rule Assembly Language that you use for rules written using rule templates.

You can modify the Rule Designer list to display the RAL version of each rule you create using rule templates. This is a useful way to learn RAL and to understand more fully how rule templates work. To modify the Rules Designer list, see "Displaying RAL in the Rule Designer" on page 462.

Specifying Data

This section describes how to specify numbers, strings, names, data types, and property types in Rule Assembly Language.

Numbers

You can use both integers and floating point numbers. Floating point numbers contain a decimal point.

- Example of integers: `1, 100, -239`
- Example of floating point: `3.14, 1.0, 10.567`

Strings

Enclose strings in double quotes. For example:

```
"Parker Data Systems recommends a DSL modem"
```

White space in a string is treated as a character. Use a back-slash (\) as an escape character to include double quotes or a back-slash in a string. For example:

```
"Install these fonts in C:\\psfonts on your system"
```

If you put quotes around a number or anything else, it is treated as a string.

Links

Links store data extracted from Siebel databases. Links can also store the value of specific system variables. Links can be used only to define conditions. Enclose link names in square brackets.

Understanding Operators

In expressions, operators define operations or relationships between operands. Operator names are case sensitive. For example, `Req(A, B)` is not the same as `req(A, B)` and will result in a syntax error.

Most operator names are entirely lowercase. However, a few contain capital letters and are noted in later sections. The Rule Assembly Language has several types of operators.

- **Siebel data**. These operators support expressions involving data that originates elsewhere in the Siebel system.
- **Boolean**. These operators take logical operands and return logical results. For example: `and(`*`A, B`*`)`.
- **Comparison**. These operators take numeric operands and return logical results. For example: `>(`*`A, B`*`)`.
- **Arithmetic**. These operators take numeric operands and return numeric results. They provide math operations such as addition and subtraction.
- **Conditional**. These operators provide conditional logical and numerical relationships, such as if-then-else.
- **Special**. These operators interpret their operators in a special way. Some provide access to the configuration session: for example, to signal messages or retrieve property values. They also provide binding and iteration services.

Some operators expect logical operands. Others expect numeric operands. When an operand is of a type different than the operator expects, the eConfigurator engine forces the operand to the correct type.

- When integers are used where floating point is expected, integers are converted to their double-precision floating point equivalent.
- When floating point numbers are used where integers are expected, floating point numbers are rounded to their nearest integer value.
- Numbers greater than zero are interpreted by logical operators as true.
- Numbers less than or equal to zero are interpreted by logical operators as false.
- When used as numeric operands, true is 1 and false is 0.

Data Operators

Use the following operators, shown in Table 35, when working with Siebel data types.

Table 35. Siebel Data Operators

Operator	Syntax	Properties
Number	`Number(A)`	Converts the operand to a number. Operand can be an expression.
String	`String(A)`	Converts the operand to a string. Operand can be an expression.
Date	`Date(A)`	Converts string to a date. Operand can be an expression.
Time	`Time(A)`	Converts the operand to a time. Operand can be an expression.
UtcDateTime	`UtcDateTime(A)`	Converts the operand to the UTC date and time. Operand can be an expression.
DateTime	`DateTime(A)`	Converts the operand to a date and time. Operand can be an expression. Not recommended for use. Use UtcDateTime instead.
Currency	`Currency(A)`	Converts the operand to currency. Operand can be an expression.
Phone	`Phone(A)`	Converts the operand to a phone number. Operand cannot be an expression. Can be used only with Equals (= =) and Not Equal To (! =) operators to compare two phone numbers.

Boolean Operators

When you specify this type of operator, the eConfigurator engine interprets it as true when the item is in the solution, and as false if the item is not in the solution. If you specify a resource or an arithmetic sub-expression as an operand, the eConfigurator engine interprets it as true if the expression is greater than zero, and false if the expression is less than or equal to zero.

The requires operator (req()) is an example of this type of operator. The eConfigurator engine interprets the following rule to mean item A requires item B. In other words, if A is in the solution, B must be in the solution.

```
req([A], [B])
```

The eConfigurator engine does not interpret this rule to mean the current quantity of item A requires the same quantity of item B. That rule would be written as follows:

```
==([A], [B])
```

The Boolean operators are shown in Table 36.

Table 36. Boolean Operators

Operator	Syntax	Properties
Not	`!(A)`	Logical negation. True when *A* is false and false when *A* is true. *A* can be an item or sub-expression.
Requires	`req(A, B)`	*A* implies *B*. False only when *A* is true and *B* is false. *B* does not require *A*. However, excluding *B*, excludes *A*. *A* and *B* can be items or sub-expressions.
Excludes	`excl(A, B)`	*A* excludes *B* and *B* excludes *A*. Same as "not both *A* and *B*." False only when *A* and *B* are both true. *A* and *B* can be items or sub-expressions.
And	`and(A, B)`	Both *A* and *B*. True only when both *A* and *B* are true. When used as a top-level constraint, means that only solutions where both *A* and *B* are true are allowed. *A* and *B* can be items or sub-expressions.
Inclusive Or	`or(A, B)`	Either *A* or *B* or both. False only when both *A* and *B* are false. *A* and *B* can be items or sub-expressions.

Table 36. Boolean Operators

Operator	Syntax	Properties
Exclusive Or	`xor(A, B)`	*A* or *B* but not both. *A* and *B* must have opposite truth states. False when *A* and *B* are either both true or both false. *A* and *B* can be items or sub-expressions.
Logically Equivalent	`eqv(A, B)`	*A* requires *B* and *B* requires *A*. True only when *A* and *B* are either both true or both false. *A* and *B* can be items or sub-expressions.

Table 37 provides the truth-state definition of how the Boolean operators work. The first two columns contain the operands A and B. These could be items or arithmetic expressions, in which case A > 0 means that A is in the solution.

Table 37. Truth Table for Boolean Operators

A >0?	B>0?	req(A, B)	excl(A, B)	and(A, B)	or(A, B)	xor(A, B)	eqv(A, B)
T	T	T	F	T	T	F	T
T	F	F	T	F	T	T	F
F	T	T	T	F	T	T	F
F	F	T	T	F	F	F	T

More on the Requires Operator

The requires operator is not an incremental add. For example, you write the rule req(A, B). If the user picks A and there are no B's in the solution, the eConfigurator engine will add at least one B. The next time the user picks A, the engine does not add another B.

If you want to add a B each time an A is added, use the inc() operator.

More on the Logical Equivalence Operator

As a top level rule, the logical-equivalence operator (eqv()) creates a mutually-requires relationship between its operands. The operands can be either items or sub-expressions. For example eqv([A], [B]) means than if item A is in the solution, then at least one item B must be in the solution. Also, if item B is in the solution, then at least one item A must be in the solution. Note that the relationship between [A] and [B] is noncumulative. (Use the inc() operator to create cumulative-requires relationships.)

For example, the following rule states that if the quantity of [A] > 2, then [B] is required.

```
eqv(>([A],2),[B])
```

This expression constrains the solution as follows:

- If there are more than two A's in the solution, there must be at least one B.
- If there cannot be any B's in the solution, there cannot be any more than two A's.
- If B is in the solution, there must be more than two A's.
- If A is limited to two or less, B is excluded.

More on the Excludes Operator

As a top-level rule, the excludes operator (excl(A, B)) creates a mutually exclusive relationship. The expression excl([A], [B]) means that if item A is in the solution, item B cannot be in the solution. It also means that if item B is in the solution, item A cannot be in the solution.

The exclude rule can also be used with sub-expressions. For example, the following rule excludes item B when item A's quantity is greater than 2. It also prevents item A from being greater than 2 when item B is in the solution.

```
excl(>([A],2),[B])
```

Multiple Operands for Require and Exclude Operators

You can use multiple operands in requires and exclude rules. For example, you could write the following rule:

```
excl([A],[B],[C])
```

This syntax is interpreted by the eConfigurator engine as if you had written two rules:

```
excl([A],[B])
```

```
excl([A],[C])
```

In other words, [A] excludes *both* [B] and [C]. Note that [A] is the first operand in both the rules. The eConfigurator takes the first operand and creates expressions between it and each remaining operands.

This works the same way for require rules:

```
req([A],[B],[C])
```

This syntax is interpreted by the eConfigurator engine as if you had written two rules:

```
req([A],[B])
```

```
req([A],[C])
```

In other words, [A] requires *both* [B] and [C]. There is no limitation on the number of operands you can use in this type of expression.

Comparison and Pattern Matching Operators

Comparison operators expect numeric operands. This means when you specify an item or arithmetic expression, the eConfigurator engine uses the quantity of the item or the value of the expression. These operators produce a logical result. This means the operator evaluates and compares the quantities of the operands and returns a true or false result.

The greater-than operator (>) is an example. The eConfigurator engine interprets the following top-level rule to mean the quantity of item A must be larger than the quantity of item B in the solution. The eConfigurator engine enforces this constraint by adjusting the quantity of A or B as needed to make sure that the constraint is always true.

```
>([A],[B])
```

When used as sub-expressions, comparison operators return true or false. For example, if the quantity of item A in the solution is not greater than the quantity of item B, the example above returns false. This is then acted on by the associated top-level expression.

Pattern matching operators compare two strings. You can test whether the strings are a match or a mismatch. Pattern matching operators return true or false.

Comparison operators are shown in Table 38.

Table 38. Comparison and Pattern Matching Operators

Operator	Syntax	Properties
Greater than	>(*A*, *B*)	*A* and *B* can be items or sub-expressions.
Not less than	>=(*A*, *B*)	*A* and *B* can be items or sub-expressions.
Equals	==(*A*, *B*)	*A* and *B* can be items or sub-expressions.
Not equal to	!=(*A*, *B*)	*A* and *B* can be items or sub-expressions.
Not greater than	<=(*A*, *B*)	*A* and *B* can be items or sub-expressions.
Less than	<(*A*, *B*)	*A* and *B* can be items or sub-expressions.
Selected	sel(*A*)	Returns true if *A* is positive, false if *A* is not. Same as >(*A*, 0). If used as top-level expression, means that *A* must be in the solution in any quantity. *A* can be an item or sub-expression.

Table 38. Comparison and Pattern Matching Operators

Operator	Syntax	Properties
Pattern match	`Like(A, B)`	Result is true when string *A* pattern-matches string *B*. Similar to Siebel search specification. For example *B* could be "Pentium*". Note capitalization of operator. *A* and *B* must be constants. For example, *A* and *B* cannot be attribute names.
Pattern mismatch	`NotLike(A, B)`	Result is true when string *A* does not pattern-match string *B*. Similar to Siebel search specification. For example *B* could be "Pentium*". Note capitalization of operator. *A* and *B* must be constants. For example, *A* and *B* cannot be attribute names.

Multiple Operands for Comparison Operators

You can use multiple operands in rules containing comparison operators. For example, you could write the following rule:

```
>([A],[B],[C])
```

This syntax is interpreted by the eConfigurator engine as if you had written two rules:

> ([A],[B])

> ([A],[C])

Note that [A] is the first operand in both the rules. The eConfigurator takes the first operand and creates expressions between it and each remaining operands. There are no limitations on the number of operands you can use in this type of expression.

Note that the expression are interpreted by the eConfigurator engine as pairs that always include the first operand. This type of expression does not create implied constraints between other operands. For example, you write the following rule:

```
!=([A],[B],[C])
```

This rule expression is interpreted as A ! = B and A! = C. It does not imply that B ! = C.

Arithmetic Operators

Arithmetic operators expect numeric operands and produce a numeric result. They are most frequently used in sub-expressions. The following top-level expression means that the quantity of item C in the solution must be the same as the sum of the quantities of items A and B.

```
==(+([A],[B]),[C])
```

Assuming no other constraints on item A, B, or C; if you add A or B to the solution, then C will be added as well to match the sum. If you add a large number of Cs, the eConfigurator engine will add A and B in arbitrary quantities so that their sum equals the amount of C.

When used in sub-expressions, these operators should return a numeric result. If a sub-expression returns a logical result, true is interpreted as a 1, and false is interpreted as a 0. In the example above, if B is an expression that returns the logical result true, then the expression is equivalent to the following:

```
==(+([A],1),[C])
```

Arithmetic operators are shown in Table 39.

Table 39. Arithmetic Operators

Operator	Syntax	Properties
Addition	`+(A, B)`	Sum of *A* and *B*. *A* and *B* can be items or sub-expressions. Result is floating point if A or B is floating point.
Subtraction	`-(A, B)`	Subtracts *B* from *A*. *A* and *B* can be items or sub-expressions. Result is floating point if *A* or *B* is floating point.
Negation	`-(A)`	Additive inverse of *A*. Uses only one operand. *A* can be an item or expression.
Multiplication	`*(A, B)`	Product of *A* and *B*. Result is floating point if *A* or *B* is floating point. *A* and *B* can be items or sub-expressions.
Division	`/(A, B)`	Quotient of *A* divided by *B*. Truncates ratio to integer if both *A* and *B* are integers. Result is floating point if *A* or *B* is floating point. *A* and *B* can be items or sub-expressions.

Table 39. Arithmetic Operators

Operator	Syntax	Properties
Modulo	`%(A, B)`	Remainder of *A* divided by *B*. For example, %(1900, 72) results in 28. If *A* or *B* is floating point, the value is first rounded to the nearest integer; then the remainder is computed as for integers. *A* and *B* can be items or sub-expressions.
Minimum	`min(A, B)`	Result is the smaller of *A* and *B* and is floating point if *A* or *B* is floating point. *A* and *B* can be items or sub-expressions.
Maximum	`max(A, B)`	Result is the larger of *A* and *B* and is floating point if *A* or *B* is floating point. *A* and *B* can be items or sub-expressions.

The following operators, shown in Table 40, also take numeric arguments and produce numeric results. Use them to control numeric accuracy or change numeric characteristics.

Table 40. Additional Arithmetic Operators

Operator	Syntax	Properties
Quantity	`qty(A)`	Result is the quantity of *A* rounded to nearest integer. For example, if *A* is 6.7, returns 7. If *A* is 6.3, returns 6. *A* can be an item or sub-expression. Useful only with resources.
Integer	`int(A)`	Truncates *A* down to an integer. For example, if operand is 6.7, returns 6. *A* can be an item or sub-expression. Useful only with resources.
Float	`flo(A)`	Converts *A* to floating point. Same as multiplying operand by 1.0. *A* can be a sub-expression. Not useful with resources.
Absolute value	`abs(A)`	Returns the absolute value of *A*. *A* can be an item or sub-expression.
Sign test	`sgn(A)`	Returns -1 if the quantity of $A < 0$, 0 if $A = 0$, 1 if $A > 0$. *A* can be a sub-expression.

Attribute Operators

Rule Assembly Language includes special operators for doing comparisons and particular math operations on attribute values. These operators extract information about the attributes of all the products that have been selected in a relationship. For example, you can determine the number of relationship items that have been selected that have an attribute value greater than a specified amount.

Attribute Comparison Operators

These operators return the number of relationship items that have been selected for which the comparison is true. For example, you can use `numAttr>` to find out how many items with Length greater than 5 feet in a relationship have been selected.

The operators count all the items selected from the relationship, not the number of different items. In the preceding example, if the user selects two of the same item and enters a length greater than 5 feet, the `numAttr>` operator will return 2.

The operators take two arguments, A and B. Argument A is the full path from the root of the product to the attribute. Argument B is the comparison value. This value can be of type Integer, Number, Date, or Time. Type DateTime is not supported. Argument B can also be a sub-expression that resolves to one of these data types.

In addition, for the `numAttr==` and `numAttr!=` operators, argument B can be a text string.

Use attribute comparison operators to create subexpressions that form conditions. Attribute comparison operators are shown in Table 41.

Table 41. Attribute Comparison Operators

Operator	Syntax	Properties
Greater than	`numAttr>(A, B)`	Returns the number of items in the relationship for which the value of attribute *A* is greater than *B*.
Not less than	`numAttr>=(A, B)`	Returns the number of items in the relationship for which the value of attribute *A* is greater than or equal to *B*.
Equals	`numAttr==(A, B)`	Returns the number of items in the relationship for which the value of attribute *A* is equal to *B*.

Table 41. Attribute Comparison Operators

Operator	Syntax	Properties
Not equal to	`numAttr!=(A, B)`	Returns the number of items in the relationship for which the value of attribute *A* is not equal to *B*.
Not greater than	`numAttr<=(A, B)`	Returns the number of items in the relationship for which the value of attribute *A* is less than or equal to *B*.
Less than	`numAttr<(A, B)`	Returns the number of items in the relationship for which the value of attribute *A* is less than or equal to *B*.

You can use the numAttr operators to create "any/all" conditions in rules involving attributes:

- You create the condition "any instance of products in R has attribute A = X" as follows:

  > = (numAttr = = (@.[R].[A], X), 1)

- You create the condition "all instances of products in R have attribute A = X" as follows:

  = = (numAttr = = (@.[R].[A], X), @.[R])

For example, you want to write the rule, when all the instances of products in P have attribute A = X then exclude any instance of products in Q that have attribute B = Y. You would write this rule as follows:

```
excl( = = (numAttr = = (@.[P].[A], X), @[P]),
> = (numAttr( = = (@.[Q].[B], Y), 1)))
```

Attribute Arithmetic Operators

The arithmetic operators allow you to determine the maximum and minimum value of an attribute for items that have been selected in a relationship. You can also sum the values of the attributes.

The operators take one argument, which is the path to the attribute. Attributes can be of type Number, Integer, Date, or Time. DateTime and Text are not supported. If the type is Date or Time, `minAttr` returns the time or date closest to the present. The `maxAttr` operator, returns the time or date furthest from the present.

Use attribute arithmetic operators to create subexpressions that form conditions. Attribute arithmetic operators are shown in Table 42.

Table 42. Attribute Arithmetic Operators

Operator	Syntax	Properties
Minimum	`minAttr(`*A*`)`	For items selected from a relationship, returns the smallest value for attribute *A*.
Maximum	`maxAttr(`*A*`)`	For items selected from a relationship, returns the largest value for attribute *A*.
Sum	`sumAttr(A)`	Returns the sum of attribute *A* for all items selected from a relationship.

Conditional Operators

Conditional logic operators are shown in Table 43.

Table 43. Conditional Operators

Operator	Syntax	Properties
Logical conditional	`if (A, B, C)`	If *A* then *B*, else *C*. If *C* is not specified, it defaults to true (1).
Numeric conditional	`?(A, B, C)`	If *A* then *B*, else *C*. If *C* is not specified, it defaults to false (0). This means the expression returns false if *A* is false and *C* is not specified. Rarely used.

Special Operators

Special operators provide functions that manipulate the eConfigurator engine directly, rather than the underlying customizable product. They also provide defined types of access to the components of a customizable product.

The inc() operator is used to implement provide and consume rules. It has two important characteristics:

- It returns a null value.
- When it is evaluated in a subexpression, this causes the eConfigurator engine to contribute the value requested.

This means that writing rules with the inc() operator as a subexpression based on a condition do not work. The contribution will always occur. For example, you write the following rule.

```
req(X, inc(Y, Z))
```

The eConfigurator engine will contribute the value of Y to Z, regardless of whether X is present in the solution. This is because the eConfigurator must evaluate the two arguments to req() before determining what action to take. Evaluating the inc() argument causes the engine to contribute the value of Y to Z.

In addition, regardless of the value of X, the inc() argument always evaluates to null, which makes the rule meaningless.

To write rules that contribute conditionally, use the numeric conditional operator, ?().

Special operators are shown in Table 44 on page 391.

Table 44. Special Operators

Operator	Syntax	Properties
Constraint	`con(A)`	Makes *A* a constraint. Returns no result. *A* can be an item or sub-expression. Use to make sub-expressions into constraints. Redundant for top-level expressions. Rarely used.
Increment	`inc(A, B,)`	Each *A* requires a *B*. *A* can be a number, item, or sub-expression but must resolve to a number. *B* must be an item and cannot be a sub-expression. inc(1, *B*) increments the quantity of *B* in the solution by 1. Use `inc(A,B),` where *A* and *B* are items, to implement cumulative require rules.
Message	`msg(A) "string"`	Causes message signal when *A* is true (selected). User-defined message displays in user message area. Returns no result. *A* can be an item or sub-expression.
Check	chk(*A*) "string"	Causes a message-signal when *A* is false (not selected). User-defined message displays in user message area. Returns no result. *A* can be an item or sub-expression.
Preference	`prefer(expression, priority)`	The *expression* is enforced as a constraint only if it does not conflict with any other rule type or with any user selections. The *priority* determines the order in which multiple preference rules for an item are evaluated. Preference rules with priority 0 that apply to a specific item are evaluated first. Those with priority 1 that apply to that item are evaluated next, and so on.

Table 44. Special Operators

Operator	Syntax	Properties
With Members	`withMembers(`*A*`,`*B*`)`	Enforce the constraint *B* only on instances in *A*. This operator allows you to move the context for enforcement of a constraint from the root of a customizable product to a location within it. For example, *A* can be a path to a relationship within the customizable product. This causes the constraint *B* to be enforced only within that relationship. *A* can be a relationship, a class within a relationship, or a single product. *B* can be any expression or rule. In *B,* the path must be specified relative to the path in *A*. For example, if *A* specifies relationship R1, the paths to items in *B*, must be specified relative to R1, not the product root. When specifying items in *B*, do not include any part of the path specified in *A*. When adding items to *B* using the object picker in the Assisted Advanced Rule template, you must manually edit the item paths in *B* to make them relative to the path specified in *A*.
With Tuples	`withTuples (((A, B, C), (D,E F), ...), ruleA(%1, %2, %3,...),...)`	This operator allows you to specify multiple sets of operands for one or more rules. The first operand in a group is assigned to the variable %1, the second to %2, and so on. You can specify more than one rule.

More on withTuples

The withTuples operator lets you provide more than one set of operands to a rule. It has the following syntax:

```
withTuples (((A, B, C),(D,E F),...), ruleA(%1,%2,%3,...),...)
```

For example, you have the following two rules:

```
req(and([A],[B]),excl([C],[G]))
```

When both A and B are present, C and G exclude each other

```
req(and([D],[E]),excl([F],[G]))
```

When both D and E are present, F and G exclude each other.

The above two rules can be thought of as one require rule that has two sets of operands. The withTuples operator lets you write the rule in this fashion. This makes rule maintenance easier. If the operands change, you can edit them in one location, rather than having to locate all the rules in which they appear. Here is one way to combine the two rules using withTuples:

```
withTuples((([A],[B],[C]),([D],[E],[F])),req(and(%1,%2),
excl(%3,[G]))
```

Notice that each group of operands is enclosed in parentheses. Also notice that the whole section where the operands are specified is itself enclosed in parentheses.

You can also use the withTuples operator to specify the operands for multiple rules. For example, you have the following two rules:

```
req(and([A],[B]),[C])
inc([C], [Resource1])
```

If both A and B are present, C is required, and contribute the value of C to Resource1.

The above two rules are different but they make use of the same operands. You could use the withTuples operator to show that these two rules use the same operands as follows:

```
withTuples((([A],[B],[C])),req(and(%1,%2),%3),inc(%3,[Resource1]))
```

More on withMembers

The withMembers operator shifts the context for application of a rule from the root of the customizable product to a specified location within the product. This means the rule applies only to the items in the specified path, rather than wherever these items appear in the product.

The withMembers operator adds no other functionality and does not alter the function of other operators.

For example, you define Relationship A that contains only the customizable product desktop PC. The desktop PC includes two relationships: CPU and Hard Drive. You want to write the rule CPU requires Hard Drive and enforce the rule for each instance of the desktop PC. If the user adds a CPU and hard drive to a desktop PC but later removes the hard drive, you want the user to receive a configuration error for that PC.

You would write this rule as follows:

```
withMembers(@.Relationship A, req(@.CPU, @.Hard Drive))
```

This rule enforces “CPU requires Hard Drive” separately on each instance of desktop PC in Relationship A. All the desktop PCs from Relationship A must have a hard drive if they have a CPU. The rule is not enforced for PCs that appear elsewhere in the customizable product other than Relationship A.

Customizable Product Access Operators

Customizable Product access operators, shown in Table 45, allow you to obtain information about other areas of the customizable product. For example, you can obtain the name of an item's parent.

Table 45. Customizable Product Access Operators

Operator	Syntax	Properties
Product root	`root()`	Result is root of the customizable product. Takes no arguments.

Rule Examples

This section contains examples of how to use the eConfigurator Rule Assembly Language to create rules. Some of the rules show item names consisting of both the product name and its configuration ID (Cfg ID).

Basic Rules

The following table shows how create basic rules using Rule Assembly Language.

Rule Type	Advanced Rule Language
A requires B noncumulatively	`req(A, B)`
A requires B cumulatively	`inc(A,B)`
A excludes B	`excl(A,B)`
A provides the amount B to C	`inc(*(A,B),C)`
A consumes the amount B from C	`inc(*(A,-(B)),C)`
A's minimum quantity is B, enforced	`>=(A,B)`
A's maximum quantity is B, enforced	`<=(A,B)`
A recommends B	`rec(req(A,B))`

Boolean and Comparison Operators

The following table shows how to create rules using Boolean and comparison operators.

Rule Type	Advanced Rule Language
A AND B = C	`==(and(A,B),C)`
A OR B = C	`==(or(A,B),C)`
NOT (A = B)	`xor(A,B)`
(A < B) requires C	`req(<(A,B),C)`
(A < = B) requires C	`req(<=(A,B),C)`
(A = B) requires C	`req(==(A,B),C)`
(A != B) requires C	`req(!=(A,B),C)`
(A > = B) requires C	`req(>=(A,B),C)`
(A > B) requires C	`req(>(A,B),C)`
(A + B) contributes to C	`inc(+(A,B),C)`
(A - B) contributes to C	`inc(-(A,B),C)`
(A * B) contributes to C	`inc(*(A,B),C)`
(A/B) contributes to C	`inc(/(A,B),C)`
(A MIN B) contributes to C	`inc(min(A,B),C)`
(A MAX B) contributes to C	`inc(max(A,B),C)`

Rule Template Translations

Table 46 shows examples of the rule templates translated into Rule Assembly Language.

Table 46. Rule Template Translations to Rule Assembly Language

Template	RAL Equivalent	Explanation
Constrain	con(> (Item A),1))	Constrain (quantity of Item A > 1) to be true.
Constrain Conditionally	if(> (Item A), 1), > = (Item B), 2), excl(Item C, Item D))	When (quantity of Item A > 1)(quantity of Item B > = 2), otherwise Selection of Item C excludes selection of Item D.
Constrain Product Quantity	= = (Item A, 2)	The quantity of Item A = 2.
Consume	inc(*(Item A), -(1)), Resource B)	Each Item A consumes 1 from Resource B.
Exclude	excl(Item A, Item B)	Selection of Item A excludes selection of Item B.
Exclude List	excl(Item A, Item B, Item C))	Item A excludes (selection of Item B, selection of Item C).
Message	msg(> (Item A, 1))	When (Item A > 1), display this rule's description.
Preference	prefer(Item A, 1)	When possible, constrain Item A is selected to be true with a priority of 1.
Provide	inc(*(Item A), 1), Resource B)	Each Item A provides 1 to Resource B.
Recommends	chk(req(Item A, Item B), "Item B is recommended")	Selection of Item A recommends selection of at least one Item B by displaying "Item B is recommended".
Requires	req(Item A, Item B)	Item A requires Item B.
Requires (Mutual)	eqv(Item A, Item B)	Item A requires Item B AND Item B requires Item A.
Constrain Attribute Value	= = (@.[Class A] ([Item B]).[Day], "Monday")	The attribute Class A “Item B” Attribute C = Monday.

Customizable Product Scripts 20

This chapter explains how to use the Script Designer to enhance the behavior of customizable products. To use scripting events and methods you must be familiar with Siebel Visual Basic or Siebel eScript programming.

Understanding Scripts

A script is a Siebel Visual Basic or Siebel eScript program that runs when a predefined event to which it is assigned occurs during a configuration session. These events occur at specific points when opening or closing a session or processing a user request. You can also place scripts in a declarations area for use by other event-based scripts.

A script can refer to any product that has been added from the product table to a customizable product. These are called component products. Scripts can also be used to set attribute values of component products and of the customizable product. Scripts cannot be used to refer to relationships or links.

Scripts have full access to the Siebel API. You can use scripts to call Siebel business components and to provide information to back-end databases through Siebel business services. For a description of basic API methods, refer to "eConfigurator API" on page 469.

In addition, several special eConfigurator methods are described in this chapter. These can be included in scripts and allow you to obtain information about the current solution. You can also use them to submit requests to the eConfigurator engine. For example, you can write a script and insert it into an event that is called each time an item quantity changes in the current solution. When the user selects or removes an item, this event is called and your script runs. The script can use eConfigurator methods to determine the amount of the item or of other items in the current solution and can submit a request to the eConfigurator engine to modify item amounts. The script could also perform special computations or update an external application based on information obtained from the eConfigurator methods.

The items and rules that comprise the customizable product are called the declarative portion. The declarative portion cannot be modified by scripts. For example, you cannot add or delete rules from a customizable product using scripts.

Scripts are intended to add additional behaviors or features and can be used in several ways:

- **Arithmetic functions.** You can use scripting to perform more complex mathematical operations that cannot feasibly be created using configuration rules. For example, if you are configuring a solution for a chemical manufacturing process, you can use calculations based on viscosity, average ambient temperature, and desired throughput to arrive at the correct configuration of pumps and other equipment.
- **External program calls.** You can use scripts to make calls to external programs. For example, you can pass information about the current session to external programs.
- **Accessing Siebel objects.** You can use all the standard features of Siebel VB and and Siebel eScript, including reading and editing Siebel databases and calling Buscomps, BusObjects, and other Siebel objects.

Understanding Script Processing

Several events control script processing:

- The Cfg_InstInitialize event occurs when the user begins a configuration session and an instance of the customizable product is created. This occurs when the user clicks Customize in the Quote interface or when the user selects a customizable product for configuration in a Web page. This event is called once at the beginning of the session. Scripts associated with this event are processed after the declarative portion of the product is instantiated but before it is displayed to the user.
- The Cfg_ChildItemChanged event occurs each time the user selects or removes an item during the configuration session. Scripts associated with this event are processed after the new solution is created but before it is displayed to the user. If you insert a method to add or remove items in this event, this causes another solution to be generated.
- The Cfg_ItemChanged event occurs each time the user selects or removes an item during the configuration session. Scripts associated with this event are processed after the new solution is created but before it is displayed to the user. If you insert a method to add or remove items in this event, this causes a second solution to be generated. The script associated with this event must be associated with a component customizable product.
- The Cfig_AttributeChanged event occurs each time the user selects or changes an attribute value during the configuration session. Scripts associated with this event are processed after the new solution is created but before it is displayed to the user. If you insert a method to add or remove items in this event, this causes a second solution to be generated.
- The Cfg_OnConflict event occurs when a conflict happens during the processing of a user request. You can resolve the conflict by undoing the last request or by keeping the last request and removing previous requests that conflict with it. Use this event to resolve conflicts without prompting the user for action. This event is called after the new solution is created but before calling Cfg_ChildItemChanged or Cfig_AttributeChanged. If you insert a method to add or remove items in this event, this causes a second solution to be generated.

- The Cfg_InstPostSynchronize event occurs when you select Save or Done to end the configuration section. This event is called once at the end of the configuration session.

To write a script, you open the Script Designer and select a customizable product. You then either choose whether to write an event script, or a declarations script. Scripts in the declarations area contain methods that can be called by event-scripts and other declarations-scripts.

A script instance is created at the beginning of the associated event and destroyed at the end of the script execution. Variables defined in the declarations section of the script are meaningful only during script execution and do not persist after the script exits. For example, if a script is called because an item has changed, its variable values do not persist. The next time an item changes and the script runs again, the values of variables from the first script execution are not available.

Figure 16 shows when each event occurs during a configuration session. The Cfg_Instinitialize event occurs at the beginning of the configuration session. When the user picks an item, a new solution is generated and new baselines are set. Then the Cfg_On_Conflict event is called if there is conflict. Otherwise the Cfg_ChildItemChanged, Cfg_ItemChanged, and Cfg_AttributeChanged events are called. When the user clicks Save, Done, or updates the quote, the Cfg_InstPostSynchronize event is called.

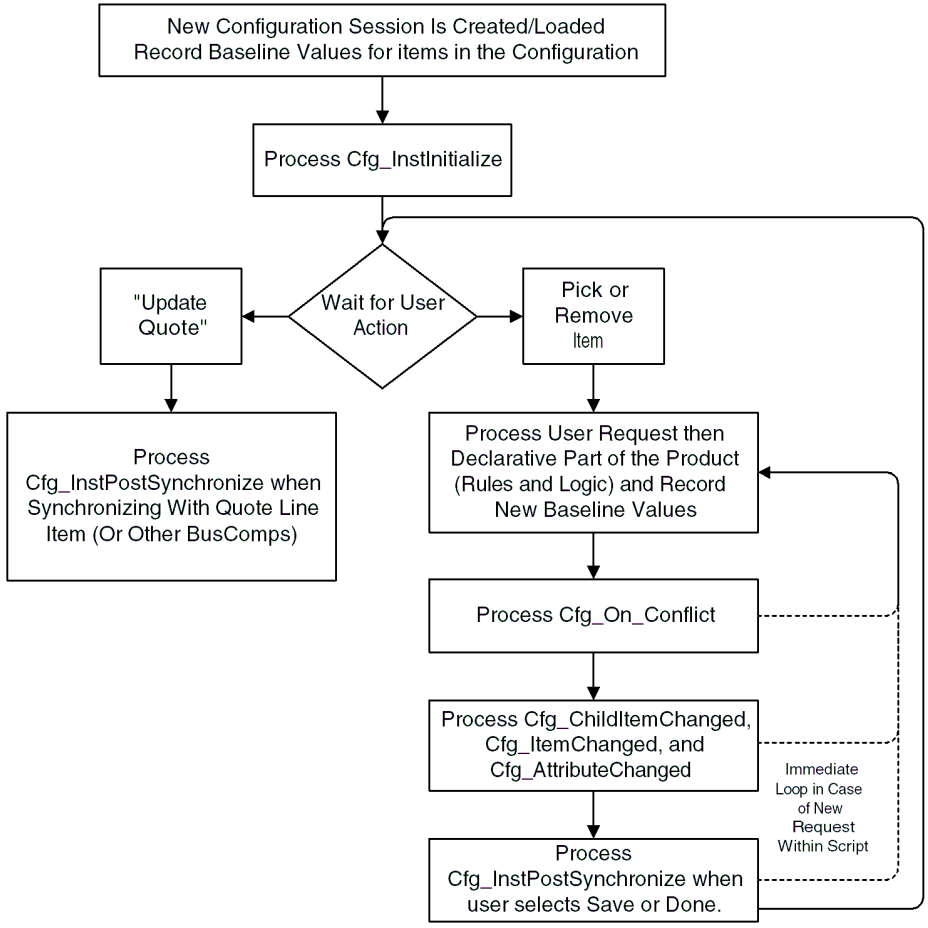

Figure 16. Order of Event Processing

Understanding Product Names

Several scripting methods have product name as an argument. Product name in this context means the name of a component product you have added from the product table to a customizable product. You must specify product names in a way that makes them unique. You do this by specifying the product name, vendor name, vendor location, and attribute values.

Specify product names using the following syntax:

{ProductName; VendorName; VendorLocation}; AttributeName1 = Value1; AttributeName2 = Value2; ...

Observe the following guidelines when specifying product names:

- ProductName is required. All other arguments are optional.
- The order of items in the name is important. ProductName must be followed by VendorName. VendorName must be followed by VendorLocation. You cannot specify ProductName followed by VendorLocation.
- If ProductName is unique, you do not have to include VendorName or VendorLocation, and you do not need to enclose ProductName in braces.

The following are examples of product names:

- {ProductName; VendorName}; AttributeName = Value
- ProductName; AttributeName = Value
- ProductName

Understanding Product Path

The product path is the path from the root of a customizable product to a component product within it. The path is a string that specifies the customizable product root and all relationship names leading to the component product. All or part of the product path are arguments to several scripting methods.

The syntax for product paths is as follows:

$.[Root Product]#1.[Relationship]#[Component Product]

Observe the following guidelines for product paths:

- The $ before .[Root Product] refers to a special configuration object called the basket. The basket contains all the objects in the customizable product.
- The #1 after [Root Product] refers to the first instance of the root product in the basket.
- Use a dot (.) to specify a relationship.
- Use a # to specify a component product within a relationship.
- Enclose relationship names and component product names in square brackets ([ ]). Use product name syntax to specify the name of a component product.
- All paths must end with a product name.

Figure 17 shows the structure of customizable product CP1.

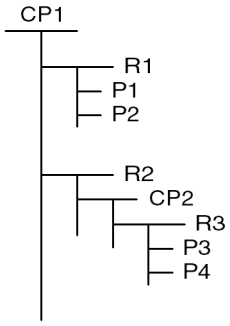

Figure 17. Customizable Product Structure

Customizable product CP1 has two relationships R1 and R2. Relationship R1 contains two component products P1 and P2. Relationship R2 contains the customizable product CP2. CP2 contains one relationship R3, which has two component products P3 and P4.

The product paths for this customizable product are as follows:

- **For P1**. $.[CP1]#1.[R1]#[P1]
- **For P2**. $.[CP1]#1.[R1]#[P2]
- **For CP2**. $.[CP1]#1.[R2]#[CP2]
- **For P3**. $.[CP1]#1.[R2]#[CP2].[R3]#[P3]
- **For P4**. $.[CP1]#1.[R2]#[CP2].[R3]#[P4]

Here are examples of using product paths in script methods:

a To add 1 P1:

```
AddItem("$.[CP1]#1", "R1", "P1", "1")
```

b To add 1 P3:

```
AddItem("$.[CP1]#1.[R2]#[CP2]", "R3", "P3", "1")
```

c To reduce the quantity of P3 to 0:

```
RemoveItem("$.[CP1]#1.[R2]#[CP2].[R3]#[P3]")
```

d To set the attribute Color to Red for P1:

```
SetAttribute("$.[CP1]#1.[R1]#[P1]", "Color", "Red")
```

Cfg_InstInitialize Event

This event is called once per session after the customizable product is instantiated and before any user requests are accepted. The customizable product selection pages do not display until all scripts associated with this event have finished.

Syntax Cfg_InstInitialize (*RootProd* as String)

Argument	Description
RootProd	String. The name of the customizable product.

Returns None.

Usage Use this event to store global variables that will be reused, such as BusObjects and BusComps. This event is also useful for reading information from external sources such as forms or SmartScript.

Cfg_ChildItemChanged Event

After a user request is processed and the eConfigurator engine computes a new solution, this event is called for the product root. The event returns a property set containing all the products whose quantities have changed.

This event is also called if the user changes an item's quantity and the Request Conflict dialog box displays:

- If the user selects OK in the dialog box, this submits a request that reverses the last request. Since this revises the item's baseline quantity, the event is called for the item.
- If the user selects Cancel, previous conflicting requests are removed from the session. Since the current baseline values do not require revision, the event is not called for the item.

This event does not return items for which only attribute values have changed. For example, if the total number of 100 GB disk drives in a solution changes, this event returns a property set containing the 100 GB disk drive because the item quantity changed.

If the user enters or selects 10 feet as the desired value of the length attribute for power supply wiring, the returned property set does not contain power supply wiring since this is an attribute change.

In the selection pages for a customizable product, this event is called when the user selects an item. It is also called when the user increases or decreases the quantity of an item. This event is called before the Cfg_ItemChanged event.

Syntax Cfg_ChildItemChanged (*ChangedItem* as Property Set)

The *ChangedItem* argument is passed as type PropertySet. This is a named XML element:

```
<ChangedItem ObjName = "objname" OldQty = "oldqty"
NewQty = "newqty"/>
```

The properties of this XML element are defined in the following table:

Property	Description
ObjName	String. The item name.
OldQty	String. The item quantity prior to the request.
NewQty	String. The new baseline item quantity.

Several Siebel API-related methods are needed to read data from the property set:

- **GetChildCount()**. Returns the total number of changed items in the property set. Use this method to set the counter for a while-loop that reads the property set.
- **GetChild(*n*)**. Returns the *nth* record in the property set. Use this method within a while-loop to read records from the property set.
- **GetProperty("*argument*")**. Returns the value of *argument*. Allowed arguments are ObjName, OldQty, and NewQty. Arguments must be in quotes. Use this method to read the property values from each record in the property set.

Returns None

Usage Use this event to determine what changes have been made to products in the solution and to submit additional requests as needed. For example, you could track the memory requirements for software the user selects and submit requests to add the correct amount of RAM to a computer configuration.

When you submit a request that changes item quantities, the submission causes the event to be called and the script runs again. Be sure to insert logic in the script that prevents an infinite loop of request submissions.

Example The following Siebel Visual Basic example writes to a file the item name, the old quantity, and the new quantity of all the items whose quantities change in each solution.

```
Sub Cfg_ChildItemChanged (ChangedItem As PropertySet)
   dim psItem as PropertySet
   dim n as Integer
   dim nCnt as Integer
   dim sObjName as String
   dim sOldQty as String
   dim sNewQty as String
   dim sMsg as String
   dim hndl as Long

   hndl = Freefile
   REM use a relative path to open cfgtest.log
   Open "..\cfgtest.log" for append as #hndl
   nCnt = ChangedItem.GetChildCount()
   For n = 0 to (nCnt -1)
      set psItem = ChangedItem.GetChild(n)
      With psItem
         sObjName = .GetProperty("ObjName")
         sOldQty = .GetProperty("OldQty")
         sNewQty = .GetProperty("NewQty")
      End With
      sMsg = "ObjName = " & sObjName
      sMsg = sMsg & "; OldQty = " & sOldQty
```

```
            sMsg = sMsg & "; NewQty = " & sNewQty
            Write #hndl, sMsg
            set psItem = Nothing
        Next
        Close #hndl
    End Sub
```

Cfg_AttributeChanged Event

After a user request is processed and the eConfigurator engine computes a new solution, this event is called for the product root. The event returns a property set containing all the products whose attributes have changed.

This event is also called if the user changes an item's attribute and the Request Conflict dialog box displays:

- If the user selects OK in the dialog box, this submits a request that reverses the last request. Since this revises the item's baseline attribute value, the event is called for the item.
- If the user selects Cancel, previous conflicting requests are removed from the session. Since the current baseline values do not require revision, the event is not called for the item.

In the selection pages for a customizable product, this event is called when the user enters or changes an attribute value.

Syntax Cfg_AttributeChanged (*ChangedAttribute* as Property Set)

The *ChangedAttribute* argument is passed as type PropertySet. This is a named XML element:

```
< Id ObjName = "objname" >

   < AttName = "attribute name" OldVal = "old value"
  NewVal = "newvalue" >

  ...

< /Id >
```

The properties of this XML element are defined in the following table:

Property	Description
ObjName	String. The item name.
AttName	String. The attribute name.
OldVal	String. The attribute value prior to the request.
NewVal	String. The new baseline attribute value.
Id	String. The object ID of the item whose attribute value has changed.

Several Siebel API-related methods are needed to read data from the property set:

- **GetChildCount()**. Returns the total number of changed items in the property set. Use this method to set the counter for a while-loop that reads the property set.
- **GetChild(*n*)**. Returns the *nth* record in the property set. Use this method within a while-loop to read records from the property set.
- **GetProperty("*argument*")**. Returns the value of *argument*. Allowed arguments are ObjName, OldQty, and NewQty. Arguments must be in quotes. Use this method to read the property values from each record in the property set.
- **GetType()**. Retrieves the object ID of the item for which the attribute was changed.

Returns None

Usage Use this event to determine what changes have been made to product attributes in the solution and to submit additional requests as needed. For example, you could track the attributes selected for a product and submit requests based on them.

Example The following example, writes to a file the item name, the old attribute value, and the new attribute value of all the items whose attribute values change in each solution.

```
{
   var item;
   var log = Clib.fopen("c:\\attchgd.log", "a");

   var id = ChangedAttribute.GetType();
   Clib.fputs(id, log);

   var nCnt = ChangedAttribute.GetChildCount();
   Clib.fputs(nCnt, log);

   for ( var i = 0; i<ChangedAttribute.GetChildCount(); i++ )
   {
      item = ChangedAttribute.GetChild(i);
      var attName = item.GetType();
      var oldV = item.GetProperty("OldVal");
      var newV = item.GetProperty("NewVal");
      var s = "AttName = " + attName;
      s = s + "; OldVal = ";
      s = s + oldV;
      s = s + "; NewVal = ";
      s = s + newV;
      Clib.fputs(s, log);
   }
   Clib.fclose(log);
}
```

Cfg_InstPostSynchronize Event

This event is called for the product root after the user clicks Done in the selection pages to end a configuration session. No further processing by the eConfigurator engine occurs in connection with this event. Using this event to adjust item quantities or attribute values is not recommended.

Syntax Cfg_InstPostSynchronize (*RootProd* as String)

Argument	Description
RootProd	String. The name of the customizable product.

Returns None

Usage Use this event to add or modify line item information before it is stored. You can also use this event to modify pricing or do other activities associated with quote line items.

Cfg_ItemChanged Event

After a user request is processed and the eConfigurator engine computes a new solution, this event is called for each component customizable product whose quantity has changed. The script associated with this event must be associated with the component customizable product.

For example, CP1 is a customizable product. One of its component products is customizable product CP2. A script inserted in the Cfg_ItemChanged event in CP2 runs when the quantity of CP2 changes while CP1 is being configured.

To set up a Cfg_ItemChanged script for the component customizable product CP2 you must do the following:

- Select CP2 in the product table and lock its work space.
- Open the script editor and select the Cfg_ItemChanged event.
- Specify CP1 as the root product for the script.
- Write the script, check its syntax, and save the script.
- Release a new version of CP2.

This event provides a simple way to write scripts for a customizable product that run only when that product is a component of another customizable product. This event is called after the Cfg_ChildItemChanged event.

Upgrade users: Use the Cfg_ChildItemChanged event to obtain functionality similar to the Cfg_ItemChanged event in release 6.x.

In the selection pages for a customizable product, this event is called when the user selects the component customizable product. It is also called when the user increases or decreases the quantity of that product.

Syntax Cfg_ItemChanged (*ProdName, OldQty, NewQty*)

Argument	Description
ProdName	String. The name of the component customizable product. Use product name syntax
OldQty	Integer. The component customizable quantity prior to the request.
NewQty	Integer. The new baseline quantity for the component customizable product.

Items Use this event only to write scripts in customizable products that will be components of other customizable products.

Returns Returns the component customizable product name, the quantity of it in the solution prior to the user request, and the quantity after the user request. This event does not return changes in the quantity of the products comprising a component customizable product.

Cfg_OnConflict Event

This event is called for the product root when the eConfigurator engine encounters a conflict while computing a solution. A conflict is when a user action violates a constraint. The constraint can be in the declarative portion of the product or can be a user pick.

When this event is called, if no script is defined, the system's normal conflict resolution messages display. Typically, the user must add or remove items to resolve the conflict. The user can retain the last user-pick and undo a previous user-pick, or they can undo the last user pick.

If a script is defined, you can resolve the conflict in the script. The system does not prompt the user with a conflict message. Submitting a request in a script on this event is not recommended.

Syntax Cfg_OnConflict (*Explanation, Resolution*)

Argument	Description
Explanation	String. Passes in the system message explaining the conflict
Resolution	String. Accepts as output one of the following values: ■ **UndoLastRequest**. This argument causes an undo of the last user request. ■ **RemoveFailedRequests**. This argument causes an undo on all user requests that cause the last request to fail. The last request is retained. If neither of the previous arguments is passed as output, the system presents the user with the normal conflict messages.

Returns Returns the system error message and accepts one of two values as output. The outputs are forwarded to the eConfigurator engine and are used to resolve the conflict.

Usage Use this event to manage conflict resolution in the background. For example, you can could create If-then statements that pass one of the outputs depending on a configuration condition in the model. If the conditions for handling the output do not exist, then passing no value causes the normal conflict resolution message.

GetInstanceId Method

This method returns the system ID of the product root.

Syntax GetInstanceId() as String

Argument	Description
None	

Items Not applicable.

Returns Returns the system ID of the customizable product root.

GetCPInstance Method

This method returns the entire structure of a customizable product as a property set.

Syntax GetCPInstance() as property set

Argument	Description
None	

Items Not applicable.

Returns Returns the structure of the customizable product as a property set. Depending on the structure of the customizable product, the property set can be complex. To learn how to access the property set, use the following JavaScript code to dump the property set to a file. You can then study the property set structure to determine how to access it using a script.

```
/*

Use the PropertySetToFile(PropSet, fileName, title) API in your
script.

   Note that fileName must be double slashed, as demonstrated in the
example below:

     PropertySetToFile(InputsPS, "..\\temp\\testPSexport.txt",
Inputs into " + MethodName);

   This will write the property set to a text file in the Siebel temp
directory.

*/

function PropertySetToFile (PropSet, fileName, title)

{

   var file = Clib.fopen(fileName, "at");

   LogData(("\n----------------------------------------------------
"), file);
```

```
   LogData(("Start Process  " +
Clib.asctime(Clib.gmtime(Clib.time()))), file);
   LogData(title, file);
   LogData("PROVIDED PROPERTY SET", file);
   WritePropertySet(PropSet, file, 0);
   Clib.fclose(file);
   return (CancelOperation);
}
function WritePropertySet(PropSet, file, Level)
{
   if ((Level == "") || (typeof(Level) == "undefined")){
        Level = 0;
   }
   var indent = "";
   for (var x = 0; x < Level; x++){
        indent += "\t";
   }
   var psType = PropSet.GetType();
   var psValue = PropSet.GetValue();
   LogData((indent + "Type: " + psType + " Value: " + psValue), file);

   var propName = PropSet.GetFirstProperty();
   while (propName != ""){
      var propValue = PropSet.GetProperty(propName);
      LogData((indent + propName + " = " + propValue), file);
```

```
      propName = PropSet.GetNextProperty();
   }
   var children = PropSet.GetChildCount();
   for (var x = 0; x < children; x++){
        LogData(( indent + "CHILD PROPERTY SET " + x), file);
        WritePropertySet(PropSet.GetChild(x), file, (Level + 1));
   }
}
function LogData(DataString, file)
{
   try {
      Clib.fputs((DataString + "\n"), file);
      Clib.fflush(file);
   }
   catch (e){
        // no action
   }
}
```

GetObjQuantity Method

This method returns the quantity of the specified component product in the current solution.

Syntax GetObjQuantity (*ProdName*) as Integer

Argument	Description
ProdName	String. The name of the component product. Use product name syntax to identity the component product.

Items Can be used only for component products within the customizable product.

Returns Quantity of the component product in the current solution as an integer. If multiple instances of the component exist, the quantity of all instances are added together and the sum is returned. For example if there are two instances of an item one with quantity five and the other with quantity three, the return is eight.

Example This script fragment obtains the quantity of 10 GB Drive (vendor = Sony) in the current solution and assigns it to the variable iItemQty.

```
Dim iItemQty as Integer

iItemQty = GetObjQuantity(“{10 GB Drive; Sony}”)
```

AddItem Method

This method creates a new instance of an item and adds the specified quantity to the solution. For example, Item A exists in the solution and has quantity three. You use AddItem to add two more Item A to the solution. The new solution will contain two instances of Item A, one with quantity three, and one with quantity two.

Syntax AddItem (*ParentObjPath, RelName, ProdName, Quantity*) as Integer

Argument	Description
ParentObjPath	String. The product path to, but not including, the relationship in which the component product resides. Use product path syntax to specify the path. If the component product is located at the product root, then specify the product root as the *ParentObjPath*.
RelName	String. The name of the relationship containing the component product you want to add. If the component product is located at the product root, then specify the customizable product name as the *RelName*.
ProdName	String. The name of the product you want to add. Use product name syntax to specify the product name.
Quantity	String. The amount of the product you want to add.

Items Can be used only for component products within the customizable product.

Returns Returns 1 if the add was successful. Returns 0 if the add fails.

Example See the section on product paths for an example of using AddItem.

RemoveItem Method

This method reduces the quantity of the specified item to 0 in the current solution. If multiple instances of an item have the same path, the eConfigurator engine randomly picks one of the instances and removes it.

Syntax RemoveItem (*ObjPath*) as Integer

Argument	Description
ObjPath	String. The full path of the component product you want to remove. Use product path syntax to specify the path.

Items Can be used only for component products within the customizable product.

Returns Returns 1 if the item removal was successful. Returns 0 if the removal fails.

Example See the section on product paths for an example of using RemoveItem.

SetAttribute Method

This method sets the value of an attribute for an item in the customizable product. This method can also be used to set attribute values for attributes of the customizable product. If multiple instances of an item have the same path, the eConfigurator engine randomly picks one instance and changes its attribute values.

Syntax SetAttribute (*ObjPath, AttName, AttVal*) as Integer

Argument	Description
ObjPath	String. The full path of the component product. For attributes of the customizable product, specify the product root. Use product path syntax to specify the path.
AttName	String. The name of an attribute of the component product or customizable product.
AttValue	String. The value to which you want to set the attribute.

For LOV domains, the *AttValue* must be one of the values in the list of values. Validation expressions defined for LOV domains are ignored.

For range of value domains, the *AttValue* must be within the domain defined by the validation expression.

Items Can be used only for component products within the customizable product.

Returns Returns 1 if setting the attribute was successful. Returns 0 if setting the attribute failed.

Example See the section on product paths for an example of using SetAttribute.

Creating an Event Script

Event scripts run when defined events are called during a configuration session. You create an event script by selecting an event method and writing the script within it.

Event scripts can call methods and objects defined in the Siebel API. They can also call methods assigned to the declarations area.

To create an event script

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.

   If you omit this step, the most recently released version of the customizable product is loaded in the Script Designer, and you cannot create or edit scripts.
3. Click the Configuration Designer tab.
4. Open the Configuration Designer menu and choose Script Designer.

   The Script Designer, Scripts list appears. It displays the scripts that have been created for this customizable product.
5. Click New to create a new record.

   A form appears.
6. In the Name field, click the down-arrow and select the desired event.

   If this is not the first script for this event, overwrite the event name with a script name. All scripts for a customizable product must have a unique script name.
7. If this is the first script you are creating for this product click the down-arrow in the Program Language field and select Visual Basic or eScript.

   If this is not the first script, click the down-arrow and select the programming language used for previous scripts.
8. Click the Root Product select button and select the current customizable product from the Pick Root Product dialog box.

9 In the Script Designer, click Save.

The form is replaced by a list of the scripts that have been created for this customizable product. The record you just created is highlighted.

10 Enter the script in the area provided in the Script Definition form.

The Script Definition form is located below the list of scripts.

11 When you have finished entering the script, click Check Syntax.

If there are errors, they will display at the top of the Customizable Product Scripts form. Correct any errors before saving the script.

12 In the Script Definition form, click Save.

13 Open the Script Designer menu and click Validate.

This starts a configuration session. Verify that the new script works correctly.

Creating a Declarations Script

Declaration scripts are methods that you want to make available to event scripts or other declaration scripts. Declaration scripts are stored in a common area accessible by event scripts. Declaration scripts can call methods and objects defined in the Siebel API.

Use declaration scripts to write methods that are common to more than one event script. Instead of repeating the method in each event script, you can write one declaration script and call it from within the event scripts that use it.

To create a declarations script

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.

   If you omit this step, the most recently released version of the customizable product is loaded in the Script Designer, and you cannot create or edit scripts.
3. Click the Configuration Designer tab.
4. Open the Configuration Designer menu and choose Script Designer.

   The Script Designer, Scripts list appears. It displays the scripts that have been created for this customizable product.
5. Click New to create a new record.

   A form appears for defining a declaration script record.
6. In the Name field, click the down-arrow and choose "(declarations)."

   If this is not the first declarations script, overwrite "(declarations)" with a script name. All scripts for a customizable product must have a unique script name.
7. If this is the first script you are creating for this product, click the down-arrow in the Program Language field and select Visual Basic or eScript.

   If this is not the first script, click the down-arrow and select the programming language used for previous scripts.

8. Click the Root Product select button and select the current customizable product from the Pick Root Product dialog box.
9. In the Script Designer, click Save.

   The form is replaced by a list of the scripts that have been created for this customizable product. The record you just created is highlighted.
10. In the Script Definition form, enter the script.

    The Script Definition form is located below the list of scripts. If you want to delete your entries and start the script over, click Reset.
11. When you have finished entering the script, click Check Syntax.

    If there are errors, they will display at the top of the Customizable Product Scripts form. Correct any errors before saving the script.
12. In the Script Definition form, click Save.

    The new script displays in the Customizable Product Scripts form when you select the script name in the Script Designer.
13. Click Save in the Scripts list to save the new script.
14. Open the Script Designer menu and click Validate.

    This starts a configuration session. Verify that the new script works correctly.

Editing a Script

You can edit scripts by selecting the script in the Script Designer. If you edit declaration scripts, verify that the changes do not adversely effect event scripts.

To edit a script

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.

   If you omit this step, the most recently released version of the customizable product is loaded in the Script Designer, and you cannot create or edit scripts.
3. Click the Configuration Designer tab.
4. Open the Configuration Designer menu and choose Script Designer.

   The Script Designer, Scripts list appears. It displays the scripts that have been created for this customizable product.
5. Highlight the script you want to edit and click Edit.

   The script displays in Script Definition. The Script Definition form is located below the list of scripts.
6. In Script Definition, edit the script.
7. When you have finished editing the script, click Check Syntax.

   If there are errors, they will display at the top of the Customizable Product Scripts form. Correct any errors before saving the script.
8. In the Script Definition form, click Save.
9. Click Save in the Scripts list to save the edited script.
10. Open the Script Designer menu and click Validate.

    This starts a configuration session. Verify that the edited script works correctly.

Deleting a Script

After deleting a script, test the customizable product in validation mode to verify that the product works correctly.

To delete a script

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.

   If you omit this step, the most recently released version of the customizable product is loaded in the Script Designer, and you cannot create or edit scripts.
3. Click the Configuration Designer tab.
4. Open the Configuration Designer menu and choose Script Designer.

   The Script Designer Scripts list appears. It displays the scripts that have been created for this customizable product.
5. Select the script you want to delete
6. Open the Scripts menu and choose Delete Record.

   Click OK when asked to confirm you want to delete this record. The record no longer displays in the Customizable Product Scripts list.
7. Open the Script Designer list menu and click Validate.

   This starts a configuration session. Verify that the customizable product functions correctly.

Reviewing the Script Log

If a method within a script fails or the script contains syntax errors, you can obtain valuable diagnostic information by reviewing the script log.

To review the script log

1. Navigate to \log\cfgscript.log. It is located in the Siebel installation directory.
2. Open the file with a text editor.
3. Locate the entries for the date and time the script ran.

Multilingual Data 21

You can specify language translations for product-related data the user sees when creating a quote or purchasing a product from an eSales Web site. This chapter describes what product data can be translated and how to specify the translations.

What Can Be Translated?

For both simple and customizable products you can specify language translations for the following data:

- Product description
- Product class display name
- Class display name
- Attribute display name
- Attribute description
- Attribute list of values

In addition, for customizable products with work spaces, you can translate the following data:

- Configuration rule explanation
- Relationship name
- UI group name
- UI property value

How Does Multilingual Data Translation Work?

The process for translating each of the types of product data is the same. The Product Administrator selects the desired item, selects a language, and then enters the translation for the item. This creates a record containing the translation. The Product Administrator can create multiple translation records for an item.

When the user logs in to either Quotes or to an eSales Web page and specifies a language, they see the item translations for that language entered by the Product Administrator.

In some cases, the lists that display items that can be translated include a field called Translate. This field is unrelated to setting up data for multilingual translation and should be ignored.

Translating the Product Description

Use this procedure to translate the product description.

To translate the product description

1. Navigate to Product Administration.
2. Select a product whose description you want to translate.
3. Open the More Info Show menu and choose Translations.

   The languages list appears. If you have not selected any languages, the list is empty.
4. Add a new record.
5. Click in the new record's Code field and select a language code from the Language Name dialog box.

   The record is updated with the language name and language code.
6. Enter the translation of the description in the Description field, and then click Save.

   A new record, containing the translation appears in the languages list.

Translating a Class Display Name

Use this procedure to translate the display name for the product class.

To translate a class display name

1. Navigate to Application Administration > Class Administration.
2. Select the product class display name you want to translate.
3. Click the Class Translations tab.
4. In the Class Translations tab, click New.

   A new record appears.
5. Click in the new record's Code field and select a language code from the Language Name dialog box.

   The record is updated with the language name and language code.
6. Enter the translation of the display name in the Display Name field, and then click Save.

   The record is updated and displays the translation of the class display name.
7. Repeat these steps to create additional language translations for the class display name.

Translating an Attribute Display Name and Description

Use this procedure to translate both the display name for an attribute and its description.

To translate an attribute display name and description

1. Navigate to Application Administration > Class Administration.
2. Select the product class where the attribute you want to translate is defined.
3. In Dynamic Attributes, select the attribute you want to translate.
4. Click the Attribute Translations tab.
5. In the Attribute Translations tab, click New.

   A new record appears.
6. Click in the new record's Code field and select a language code from the Language Name dialog box.

   The record is updated with the language name and language code.
7. Enter the translation of the display name in the Display Name field.
8. Enter the translation of the description in the Description field.
9. Click Save to save the record.

   The record is updated and displays the translations.
10. Repeat these steps to create additional language translations for this attribute's Display Name and Description.

Translating Configuration Rule Explanations

Use this procedure to translate configuration rule explanations for a customizable product.

To translate a configuration rule explanation

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.

   If you omit this step, the most recently released version of the customizable product is loaded in the Rule Designer, and you cannot create rules.
3. Click the Configuration Designer tab.

   The Rule Designer Rules List appears. It lists all the configuration rules that have been created for this customizable product.
4. Select the configuration rule containing the explanation you want to translate.
5. Open the Rules List menu and choose Translate Rule Description.

   A dialog box appears that displays the rule explanation translations you have already created.
6. In the dialog box, click New.

   The dialog box changes to display a language drop-down menu and a field for entering the rule translation.
7. Select a country name abbreviation from the Lang drop-down menu.
8. Enter the rule translation in the Description field and click OK.

   A new record appears in the dialog box showing the country abbreviation and rule translation.
9. Repeat these steps to create additional translations for this rule explanation.

Translating Relationship Names

Use this procedure to translate relationship names in a customizable product.

To translate a relationship name

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.

   If you omit this step, the most recently released version of the customizable product is loaded in the Rule Designer, and you cannot create rules.
3. Open the Customizable Product menu and choose Product Designer.

   The Product Designer view appears.
4. Select the relationship whose name you want to translate.
5. Open the Product Designer menu and choose Translate Relationship.

   A dialog box appears that displays the relationship name translations you have already created.
6. In the dialog box, click New.

   The dialog box changes to display a language drop-down menu and a field for entering the relationship name translation.
7. Select a country name abbreviation from the Lang drop-down menu.
8. Enter the relationship name translation in the Name field and click OK.

   A new record appears in the dialog box showing the country abbreviation and relationship name translation.
9. Repeat these steps to create additional translations for this relationship name.

Translating UI Group Names

Use this procedure translate group names that display in customizable product selection pages.

To translate a UI group name

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.

   If you omit this step, the most recently released version of the customizable product is loaded in the Rule Designer, and you cannot create rules.
3. Open the Customizable Product menu and choose Product UI Designer.

   The Product UI Designer view appears.
4. Select the UI group whose name you want to translate.
5. Open the Group List menu and choose Translate Groups.

   A dialog box appears that displays the UI group name translations you have already created.
6. In the dialog box, click New.

   The dialog box changes to display a language drop-down menu and a field for entering the UI group name translation.
7. Select a country name abbreviation from the Lang drop-down menu.
8. Enter the UI group name translation in the Name field and click OK.

   A new record appears in the dialog box showing the country abbreviation and UI group name translation.
9. Repeat these steps to create additional translations for this UI group name.

Translating UI Property Values

Use this procedure to translate the value of a UI Property. The property type must be type String.

To translate a UI property value

1. Navigate to Product Administration.
2. Select and lock the desired customizable product.

   If you omit this step, the most recently released version of the customizable product is loaded in the Rule Designer, and you cannot create rules.
3. Open the Customizable Product menu and choose User Interface Property Designer.

   The User Interface Property Designer view appears.
4. In Item Display Properties, select the UI property whose value you want to translate.
5. Open the Item Display Properties menu and choose Translate UI Property.

   A dialog box appears that displays the UI property value translations you have already created.
6. In the dialog box, click New.

   The dialog box changes to display a language drop-down menu and a field for entering the UI property value translation.
7. Select a country name abbreviation from the Lang drop-down menu.
8. Enter the UI property value translation in the Value Spec field and click OK.

   A new record appears in the dialog box showing the country abbreviation and UI property value translation.
9. Repeat these steps to create additional language translations for this UI property value attribute's Display Name and Description.

Translating an Attribute List of Values

For attributes with a list of values domain, you can translate the list of values name and the attribute values. You do this by specifying language name and other information when you define the list of values name and the associated attribute value records.

For example, you define a list of values called Color. It has three values, Red, Green, Blue. Your base language is English and you want to translate the list of values to French.

When you define the list of values in the List of Values dialog box, you would place a check mark in the Multilingual field. In the List of Values list, you would first define three records, one for each color, and specify English-American as the language. You would choose Color as the Type. In the Display Value and Language independent Code fields, you would enter the color name in English: Red, Green, and Blue respectively.

Then you would create three additional records, one for each color. You would choose Color as the Type. In the Language Independent Code field, you would enter Red, Green, Blue. In the Language Name field, you would choose French. In the Display Value field, you would enter Rouge, Vert, and Bleu as the Display Value respectively.

For additional information on creating and managing multilingual lists of value (MLOVs), refer to the *Global Deployment Guide*.

Cache Management 22

This chapter explains how to manage the Snapshot Mode cache. This cache improves server performance by storing information about customizable products.

Understanding Snapshot Mode

When you start a configuration session, eConfigurator looks to see if the customizable product is cached in memory. If not, eConfigurator looks in a cache directory (CFGCache) for the product. This directory, located in the Siebel file system, maintains a history of the customizable products that have been loaded into the memory cache. If the customizable product is not in CFGCache, then the customizable product is loaded from the Siebel database. When the product is loaded from the database, it is added to the memory cache and to CFGCache.

Thereafter, when a configuration session starts, the customizable product is loaded from the memory cache or CFGCache. Before loading the customizable product from CFGCache, the system checks the Siebel database to make sure each item in the product is the current version. If it is not, the current version of the item is loaded from the database. This ensures that the most recent version of a customizable product and its contents are loaded.

When the Product Administrator releases a new version of a customizable product, the changes are written to the Siebel database. They are not written to the memory cache or to CFGCache. The CFGCache directory is updated with the changes when the next configuration session is requested for the customizable product.

Snapshot Mode adds an additional memory cache. In Snapshot Mode, when you start a configuration session, customizable product items are loaded from the Snapshot Mode cache. No checking is done to see if the items are the current version. This causes the configuration session to load more quickly and improves performance. If an item is not in the Snapshot Mode cache, the item is retrieved from CFGCache, which verifies the item is the current version. The item is then added to the Snapshot Mode cache.

When Snapshot Mode is On, it works as follows:

- When you go to validate mode, the work space version of the customizable product is used rather than the version in the Snapshot Mode cache.
- When you release a new version of a customizable product, the Snapshot Mode cache is updated with the new version.
- When the Snapshot Mode mode size limit is reached, the oldest or least frequently accessed items are deleted. This means the Snapshot Mode cache contains the most recently requested or most frequently requested customizable product items.

The term *users* refers to Siebel Web Client users, Siebel Dedicated Web Client users and Siebel Mobile Web client users. Siebel Web Client users are those that access the Siebel application through a URL in a Web browser. No application is installed on the client machine. Siebel Dedicated Web Client users are those that have the Siebel application installed on the local machine. This application simulates a Siebel server and Siebel Web engine. These users access a remote Siebel database. Siebel Mobile Web Clients have both the Siebel application and a Siebel database installed on the local machine. These users obtain updates to the local database by synchronizing it with a remote Siebel database.

You can manually refresh the Snapshot Mode cache is one of several ways:

- Selecting the Refresh Cache menu option in Customizable Products > Versions. Refresh Cache erases the entire Snapshot Mode cache on all the servers connected to the database. Starting a configuration session refreshes the Snapshot Mode cache by loading the most recent version of the product from CFGCache. Since the customizable product loads from CFGCache as part of updating the Snapshot Mode cache, performance is slower for this first configuration session. All sessions thereafter load from the Snapshot Mode cache. This applies to Siebel Web Clients, Siebel Dedicated Web Clients.

  For Siebel Mobile Clients, selecting Refresh Cache erases the Snapshot Mode cache on the local machine. If synchronizing has copied a new version of a customizable product to the local database, this version will be loaded at the next configuration session. The local CFGCache and Snapshot Mode cache will also be updated.

- Restarting the Siebel database server erases the Snapshot Mode cache on the server. The next configuration session is loaded as if the user had selected the Refresh Cache menu option. This applies to Siebel Web Clients and Siebel Dedicated Web Clients.

- For Siebel Dedicated Web Clients and Siebel Mobile Web Clients, the Snapshot Mode cache and CFGCache reside on the local machine. Exiting the Siebel application is functionally the same as stopping a Siebel server. This erases the Snapshot Mode cache on the local machine.

- Editing a product record and then selecting Refresh Product Cache. This erases from the Snapshot Mode cache all the customizable products containing the product. The next time a user requests an affected customizable product, the Snapshot Mode cache is refreshed with a new instance of the product.

- Editing a class record and then selecting Refresh Class In Configuration Cache. This erases from the Snapshot Mode cache all the customizable products containing products from the class. The next time the user requests an affected customizable product, the Snapshot Mode cache is refreshed with a new instance of the product.

Snapshot Mode is highly recommended if you have large numbers of Siebel Web Client or Siebel Dedicated Web Client users, and you release new versions of customizable products relatively infrequently.

Observe the following guidelines for using Snapshot Mode.

- If a user selects the Refresh cache in Customizable Products > Versions, the Snapshot Mode cache is erased on *all* servers connected to the same Siebel database. This includes the Snapshot Mode cache on Siebel Dedicated Web Clients. Turn on Snapshot Mode only on servers and Siebel Dedicated Web Clients where it is needed. This limits the number of users that can refresh the Snapshot Mode cache.
- If the customizable product development environment and the production environment are on the same machine, and Snapshot Mode is turned On, the Product Administrator should refresh the Snapshot Mode cache frequently in order to see changes during development. Doing so, also refreshes the Snapshot Mode cache for production users.
- Siebel Dedicated Web Client users should leave Snapshot Mode turned off on their local machines if Snapshot Mode is turned on for the Siebel Server to which they connect.
- When a customizable product contains rules that have start or end dates, the arrival of these dates does not cause the revised declarative portion of the product to be loaded into the Snapshot Mode cache. You must refresh the cache manually on the effective date to load the revised declarative portion of the product.

Setting Up Snapshot Mode on the Siebel Server

You should be familiar with server administration procedures before doing this procedure.

This procedure shows you how to turn on Snapshot Mode on the Siebel Server. In addition, there are several other Snapshot Mode server parameters for managing cache performance. These are described in "Enabling Snapshot Mode" on page 458.

To set up Snapshot Mode on the Siebel server

1. Navigate to Server Administration > Servers.
2. Highlight the desired server.
3. Click the Server Parameters tab.
4. In the Parameter field, query for eprodcfgsnapshotflg.

   The parameter description is: Product Configurator-collect and use the snapshots of the Cfg object.
5. Set the Current Value to True.
6. Set the Value on Restart to True.

Setting Up Snapshot Mode on the Client

The Snapshot Mode cache for the Siebel Dedicated Web Client and the Mobile Web Client resides on the local machine. You enable Snapshot Mode on the local machine by adding parameters to the application configuration file.

To set up Snapshot Mode on the local machine

1. Use a text editor to open the .cfg file for the application you are running, for example siebel.cfg.

   This file is located in the bin subdirectory of the Siebel installation.

2. Add the following entries to the end of the file. If your customizable products are large, consider making the value of `eProdCfgNumOfCachedObjects` larger than 1000:

   ```
   [InfraObjMgr]

   eProdCfgSnapshotFlg = TRUE

   eProdCfgNumOfCachedObjects = 1000
   ```

3. Save the file.

4. Exit the application and restart.

Refreshing the Snapshot Mode Cache

For Siebel Web Client users and Siebel Dedicated Web Client users, refreshing the cache, erases the Snapshot Mode cache on all servers connected to the database.

For Siebel Dedicated Web Client users, refreshing the cache, erases the Snapshot Mode cache the local machine. It also erases the Snapshot Mode cache on all servers connected to the same database as the Siebel Dedicated Web Client.

For Mobile Web Client users, refreshing the cache erases the Snapshot Mode cache on the local machine.

To refresh the Snapshot Mode cache, you must have access to Products > Customizable Products on the Siebel server to which Siebel Web Clients are connected.

To refresh the Snapshot Mode cache from a Siebel Web Client

1. Navigate to Product Administration.
2. Select and lock any customizable product.
3. In Customizable Products > Versions, open the menu and click Refresh Cache.

Refreshing the Cache with Product Changes

When you make changes to a product record, you can use the product record as a filter to selectively update the Snapshot Mode cache. The update removes all customizable products from the cache that contain the product.

The next time the users request the customizable product, they receive a freshly instantiated version reflecting the product change and the cache is refreshed with this version. For example, you could change the product description or part number and then refresh the cache.

This removes from the Snapshot Mode cache all the customizable products containing the product. The next time a user requests the customizable product, the cache will be refreshed with a new instance of the product. The new instance will reflect the product changes.

You cannot propagate changes to class assignment by doing this type of refresh.

To refresh the cache with product changes

1. Navigate to Product Administration.
2. Select and edit any field in the desired product record.
3. Open the Products menu and choose Refresh Product Cache.

Refreshing the Cache with Class Changes

If you have Snapshot Mode turned on and a customizable product that is affected by the class change is in the Snapshot Mode cache, the changes are not propagated to the cached version of the product. The next user that requests the customizable product will receive the cached version, which does not reflect the class changes.

To make sure users receive the class changes immediately, you can use a product class as a filter to selectively delete customizable products from the cache. When you do this, all customizable products containing the specified class are erased from the Snapshot Mode cache. The next time a users request the customizable product, it is retrieved from the database and added to the cache. This new instance will reflect the changes you made to the class.

To refresh the cache with class changes

1. Navigate to Application Administration > Class Administration.
2. Select a product class and modify it or its attribute definitions as needed.
3. Open the Classes menu and choose Refresh Class In Configuration Cache.

   This erases from the Snapshot Mode cache all customizable products containing products from the class. The next time a user requests a customizable product containing products from the class, the cache will be refreshed with a new instance of the customizable product. This new instance will contain the class changes.

Technical Reference 23

This chapter provides technical information of use to server administrators and integrators.

eConfigurator Architecture

The key components of the eConfigurator architecture are as follows:

Object Manager. All services that run within a Siebel application are bound by the Object Manager they are running within. The same applies to all caches as well. Therefore services cannot be shared across object managers and neither can cached objects.

UI Business Service. The UI Business service is used by eConfigurator to render the UI. The UI business service binds the structure of the customizable product to the Web templates and submits them to the Siebel Web Engine for rendering to the client browser. The UI service is the means by which the user interacts with eConfigurator. A unique instance of the UI service is required for each user.

Instance Broker. The Instance Broker is a service that interacts with the UI Business Service. The Instance Broker maintains all the information about the current instance of the customizable product that the user is configuring. The Instance Broker interacts with other services in response to user requests during a configuration session.

Object Broker. The Object Broker is a service that extracts the customizable product definition from the database for use by other eConfigurator services.

Config Services. Config Services consists of factories.

Factory. A factory is a service that translates the customizable product definition retrieved by the Object broker into a format the worker can understand.

Constraint Engine. The Constraint engine is also called the worker. It is also referred to as the eConfigurator engine.

Worker. The worker is a service that computes solutions and enforces all the constraints associated with the configuration. This includes the declarative portion of the customizable product plus constraints added by the user (user picks).

Enabling Snapshot Mode

To use SnapShot Mode, you must turn it on by setting a server parameter. When Snapshot Mode is turned on, the eConfigurator server runs using cached objects, factories, and workers as much as possible. This improves performance.

When turned off, the eConfigurator server creates these objects for each user session. Table 47 lists the server parameters for managing Snapshot Mode.

Table 47. Server Parameters for Managing Snapshot Mode

Parameter Name	Display Name	Data Type	Default Value	Description
eProdCfgSnapshotFlg	Product Configurator-Collect and use snaphots of the Cfg objects	Boolean	FALSE	Enables or disables Snapshot Mode. Set to TRUE to turn on Snapshot Mode.
eProdCfgNumOfCachedObjects	Product Configurator-Number of objects cached in memory	Integer	1000	Sets maximum number of objects a user can have in memory cache.
eProdCfgNumbOfCachedFactories	Product Configurator-Number of factories cached in memory	Integer	10	Sets maximum number of factories that can be in memory cache.
eProdCfgNumbofCachedWorkers	Product Configurator-Number of workers cached in memory	Integer	50	Sets maximum number of workers that can be in memory cache.

Table 47. Server Parameters for Managing Snapshot Mode

Parameter Name	Display Name	Data Type	Default Value	Description
eProdCfgMaxNumbOfWorkerReuses	Product Configurator-Number of worker reuses	Integer	10	Sets maximum number of times a worker can be reused.
eProdCfgNumbOfCachedCatalogs	Product Configurator-Number of cached catalogs	Integer	10	Sets maximum number of catalogs that can be cached. Should be set to same value as eProdCfgMax-NumbOfWorker-Reuses. Catalogs contain the default product structure.

Enabling Auto Match

When a new version of a customizable product is released, Auto Match adjusts the configuration of the product in a quote, asset, or order to reflect the changes. Auto Match is disabled by default.

For Web Client users, you turn Auto Match on by setting its server parameter to TRUE. Table 48 shows the Auto Match server parameter.

Table 48. Server Parameter for Auto Match

Parameter Name	Display Name	Data Type	Default Value	Description
eProdCfgAutoMatchInstance	Product Configurator - auto match quote on reconfigure.	Boolean	FALSE	When set to FALSE, Auto Match is turned off. When set to TRUE, Auto Match is turned on.

For Dedicated Web Client users (also called mobile client users), add the following entries to the configuration file used to start the application, for example Siebel.cfg.

```
;; This section will be read for mobile clients only

[InfraObjMgr]

eProdCfgAutoMatchInstance=TRUE
```

Enforcing the Field Length for Entering Advanced Rules

The Advanced Rule template allows you to enter a rule containing several thousand characters. However, the database can store rules that contain only up to 900 characters.

You can revise the business component associated with the Advanced Rule template so that you cannot enter more than 900 characters. This business component is used for populating several lists. Revising the business component enforces the 900 character limit on all these lists. Use Siebel Tools to determine the other lists that are affected.

To enforce the field length

1. In Siebel Tools, locate the Rule Designer Dummy List VBC business component.

   It is located in the Rule Designer project.

2. Locate the field called 0 (zero).

3. Set the Text Length value to 900.

4. Recompile the desired application and test.

Displaying RAL in the Rule Designer

You can revise the Rule Designer (Rules List) to add a field that displays the Rule Assembly Language (RAL) translation of your template rules. This is a useful way to learn how to use RAL to write configuration rules.

You must use Siebel Tools to add the field to the Rule Designer and then recompile the siebel.srf file. You should be familiar with creating and modifying applets in Siebel Tools before performing this procedure.

The process for revising the Rule Designer has three tasks:

a Locate the Rule Designer applet.

b Modify the Rule Designer applet.

c Recompile the application siebel.srf file.

Locate the Rule Designer Applet

This task selects a target browser and queries for the Cfg SWE Rule Manager Applet.

To locate the Rule Designer applet

1. Save a copy of the siebel.srf file. It is located in the objects subdirectory of your installation directory.
2. Start Siebel Tools.
3. Select View, Toolbars, Configuration Context.
4. In the Target Browser Group drop-down menu, select Target Browser Config.
5. In Available browser groups, select ALL and click the right-arrow to transfer it to “Selected browser groups for layout editing.”
6. Click OK.
7. Click Applet in the Object Explorer.
8. In the Applets list, query for the Cfg SWE Rule Manager Applet.

Modify the Rule Designer Applet

This task adds the Rule Spec field to the Cfg SWE Rule Manager Applet.

To modify the Rule Designer applet

1. With the SWE Rule Manager Applet highlighted, select Tools, Lock Project.
2. Right click the highlighted applet record and select Edit Web Layout.
3. In the window displaying the layout, right-click and select Preview from the pop-up menu.
4. Click the Template icon and select the Applet List (Base/EditList) template. It's the default.
5. In the Mode drop-down menu, select 3: Edit List.
6. In the Controls/Columns window, click Rule Spec, and drag it to the [field] just to the right of End date in the applet display.
7. Click Save. Click OK on the pop-up message that asks if you want to save your changes.

Recompile Siebel.srf

This task recompiles the application's siebel.srf file.

To recompile siebel.srf

1. Click Tools, Compile.
2. Select Locked Projects.
3. Enter the path to the application siebel.srf file. It is located in the objects subdirectory of the installation. Do not enter the path to the siebel.srf file in the Tools installation directory.

Turning Off Default Instance Creation

When you add a customizable product to a quote, order, or agreement, a default product instance is created. This causes the default items in the customizable product to display as line items. When the user clicks Customize, another instance is created for the configuration session. The default instance is not used.

For large customizable products, creating the default instance can significantly increase the time required to add the customizable product to Line Items to the quote or order. To improve performance, you can turn off default instance creation. When you add a customizable product, this causes it to display as a single line item. The default components do not display as line items.

This will not affect performance when the user clicks Customize since this creates a new product instance. Turning off default instance creation applies only to customizable products. It does not apply to bundles.

To turn off default instance creation

1. In Siebel Tools, locate the Quote, Agreements, or Orders business component.
2. Display user properties.
3. Set the Skip Loading Default Cfg Instance user property to Y.
4. Recompile the desired application and test.

Revising the System Default Cardinalities

When a you create a relationship in a customizable product, you can specify a minimum, maximum, and default cardinality. If you do not specify cardinalities, the system uses the following defaults:

- Minimum cardinality = 0
- Default cardinality = 0
- Maximum cardinality = 999

If you do not specify cardinalities this means that users are not required to select any items from the relationship and are limited to selecting a maximum of 999 items.

You can change these defaults as needed. For example, you can set the maximum system default cardinality to a number larger than 999.

To revise the system default cardinalities

1. In Siebel Tools, locate the Complex Product Structure BusComp.
2. Within the business component, locate the desired field: Default Cardinality, Max Cardinality, or Min Cardinality.
3. Display the user properties for the field.
4. Set the Pre Default Value user property to the desired amount.

   The amount should be an integer that is greater than or equal to 0.

Displaying Fields from S_PROD_INT in Selection Pages

You can add the fields from the Product Master tables (S_PROD_INT) to selection pages. The process has the following steps:

a Add the fields to the Cfg CX Buscomp and define user properties. This buscomp is part of the Object Broker and extracts data from S_PROD_INT.

b Add SWE code to the desired Web template. The SWE code retrieves the field from the buscomp and displays it in selection pages. Fields display as text boxes.

c Delete the contents of the CFGCache directory. This forces the system to create a new instance of the customizable product containing the fields.

You can display text fields only for product items or for the product root. This means you can insert the SWE code only in the following places:

- For-each loops that iterate on relationship domains or the children of relationship domains. You cannot insert the code in for-each loops that iterate on attributes or on groups.
- At the root level. The template in which you insert the SWE code must not be called from inside a for-each in any other Web template.

The procedures in this section require you to have a thorough knowledge of Siebel Tools. You must also have a thorough understanding of eConfigurator Web template structure.

Add Fields to the Cfg CX Buscomp

This procedure adds the fields you want to display to the Object Broker and recompiles the application. This makes the fields available for display.

To add fields to the Cfg CX Buscomp

1. Locate the Cfg CX Buscomp in Siebel Tools.
2. Add the desired fields from S_PROD_INT to the buscomp.
3. For each field you add, define a user property called Cfg UI Field. Set the user property value to TRUE.
4. Recompile the repository and copy it to the application installation directory.

Add SWE Code to the Web Template

The following example shows the SWE code you would insert in a Web template to retrieve the Part Number field for display:

```
<swe:control id="swe:101Id+4400" CfgUIControl="CfgLabel"
CfgHtmlType="CfgLabel" property="FormattedHtml"
CfgFieldName="Part Number"/>
```

The "id" must be that specified in the for-each loop iteratorName, and the increment amount must be unique within the for-each loop.

If you want to display a field name next to the field value, insert an swe:control statement that extracts the field name from the repository. This allows you to support localization. You can insert the swe:control wherever needed in the template. It does not have to be inside a for-each loop. Here is an example of an swe:control tag that extracts the field name for Part Number from the repository. The "id" in the tag must be present but is not used for anything. The lblPartNumber value is the name of the label control in the repository.

```
<swe:control id="partnum" CfgUIControl="lblPartNumber"
property="Displayname"/>
```

To add SWE code to a template

1. Copy the desired template and give it a new filename.
2. Insert the SWE code into the new template.
3. Add the new template to the Pick UI Style dialog box.
4. Select the new template as the UI control for a relationship or an item.

Delete Contents of CFGCache Directory

You must delete the contents of this directory. This makes sure that the system loads your changes when generating a customizable product, rather than loading the objects from the cache directory.

To delete the contents of the CFGCache directory

1. Locate the Siebel File System directory.

   To see the directory path or system name for the directory, open the Siebel application Help menu and choose Technical Support.

2. In the Siebel File System directory, locate the CFGCache directory.

3. Delete all the files in the CFGCache directory.

eConfigurator API

This section is intended to summarize the available API to the Siebel eConfigurator, version 7.0.x. It explores in some detail a segment of those API.

This is intended as an introduction to the topic for advanced users. This section assumes knowledge of the eConfigurator and Siebel server architecture. Implementing the API described in this section also requires proficiency in Siebel EAI and Siebel Object Interfaces.

In order to use these API, the user should be familiar with the following:

- Siebel Business Process Designer
- Runtime Events (personalization) if invoked from the UI
- Siebel Object Interfaces
- A Siebel scripting language (Siebel VB or Siebel eScript)
- Recursive programming techniques
- Constraint satisfaction theory
- Underlying behavior of the Siebel eConfigurator
- Siebel product definition data model
- Siebel Property Set representation of data (creation and transformation)
- EAI Transports and Interfaces

Available API

There are three main groups of API for accessing the eConfigurator. These API are supported by the Complex Object Instance Service.

- Group 1: UI
  - CPRUI Service API as Siebel Web Template items.
  - DOM API within the browser inherent in JavaScript and HTML.
- Group 2: Model
  - Scripts that execute in the context of the current session that are implemented as part of the configuration model in the Script Designer View.
- Group 3: Instance
  - API for using the eConfigurator or manipulating the configuration session from other than the Configurator runtime UI.
  - The Remote Complex Object Instance Service. This business service is available for accessing the Instance API.

Explanation of the Instance API

This section presents general concepts:

- The Remote Complex Object Instance Service is a business service. It can be accessed by anything in the Siebel architecture that can use a business service. As a business service, it is used by invoking methods, passing in property sets with input arguments, and getting results from the Outputs property set.
- A session is uniquely identified by two ID values. The Object Id and the Root Id. In the case of a quote, the Object Id is the Quote Id and the Root Id is the Quote Item Id for the top-level parent (the root). In assets, the Object Id and the Root Id are both the root Asset Id.
- A session is unique only within its own user session on a given Object Manager.
- A port is another name for a Relationship.
- A complex product is another name for a customizable product.

- The Port Id is the ID of the relationship as defined in the Complex Product Structure BusComp.
- The Prod Item Id is the ID of the relationship item as defined in the Complex Product Structure BusComp.
- The Path for an item is the Integration ID of the specific item.
- Version arguments are used only when testing a customizable product version that is different from the currently released version.

NOTE: The parameters are property set and, unless indicated, all properties are on the root level property set.

LoadInstance

Loads the complex object in memory. This is the starting point for all configurations.

In:

- ObjId – unique identifier to the complex object header (e.g. Quote ID).
- RootId – unique identifier to the complex object root (e.g. Quote Line Item row id).
- IntObjName – name of the integration object specified in Siebel Tools.
- TriggerEvent – flag that determines if script events are triggered. Should normally be "Y". Set to "N" for special uses of the API where script events are not desired.
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.
- NewRecord (optional) – if set to "Y", the instance will be populated with default values.
- AutoSync (optional) – if set to "Y", the instance will be synchronized to the database immediately after loading.

- SearchSpec (optional) – set this parameter to filter out all other hierarchical instances in the child buscomp. This parameter must have the following format:

  "[Header Buscomp.Id] = 'Id' AND [Item Buscomp.Root Id] = 'Root Id'"

  ex. [Quote.Id] = '10-4FR6D' AND [Quote Item.Root Id] = '10-81DUX'

- ExternalScript (optional) – this parameter must be set to "Y" when running headless configurations (e.g. through Siebel COM Data Server). Anything other than the Cfg Web UI Service is considered headless configuration. The difference is who resolves the linked items.

Out:

The following properties will be returned from the output property set:

- CreateSession – if this property is set to "Y" the method CreateSession must be called after LoadInstance.
- IsConfig (optional) – if this property is present and set to "Y" the configuration model has configuration rules defined.
- Links (optional) – if this property is set to "Y" the model has linked items.
- UnresolvedLinks (optional) – if this property is set to "Y" the model has unresolved linked items that must be calculated by the caller.

- If NewRecord is set to "Y" in the input property set and CreateSession is set to "N" in the output property set, the output will have the instance property set returned as a child of type "CxObj" (see output example below). Example return property set*:*

```
< IsConfig='Y' UnresolvedLinks='Y' CreateSession='Y' Links='Y'>

<Links 1-19D0X='10/19/2001' 1-1Z876='SADMIN' 1Z771='SADMIN'>

</Links>"

   <UnresolvedLinks>"

      <UnresolvedLink DispName='Quote Name'
Definition='<CfgVariableDef BUS_OBJ = ""Quote"" BUS_COMP = ""Quote""
FIELD_NAME = ""Name"" SEARCH_SPEC = """" SORT_SPEC = """" DEFAULT_VAL
= """" EXECUTE = ""N""/>' Description='' DefValue='' Name='Quote
Name' BusObj='Quote' Field='Name' ID='1-1Z875' BusComp='Quote'>

      </UnresolvedLink>

   </UnresolvedLinks>

</>
```

Extracting the instance property set from LoadInstance. The instance property set can be extracted by first getting the child property set of type "CxObj" and then extracting its only child.

Links and what to do with them: A child property set of type Links will be returned if the model has linked items. The Links child property set must then be extracted and passed in to CreateSession's OUTPUT arguments as a child property set. In 7.0.4 this changed to the INPUT property set for CreateSession. It is possible that configuration rules have been defined for these linked items and so the configuration session must know the link values. The linked items are represented as property–value pairs with link IDs as properties and link values as property values.

For example:

```
<Links 1-19D0X='10/19/2001' 1-1Z876='SADMIN' 1-1Z771='SADMIN'>
```

A child property set of type UnresolvedLinks will be returned if the model has linked items which the business service was unable to resolve. The children of this property contain the information necessary to calculate the value of the linked item.

```
      <UnresolvedLinks>"

         <UnresolvedLink DispName='Quote Name'
Definition='<CfgVariableDef BUS_OBJ = ""Quote"" BUS_COMP = ""Quote""
FIELD_NAME = ""Name"" SEARCH_SPEC = """" SORT_SPEC = """" DEFAULT_VAL
= """" EXECUTE = ""N""/>' Description='' DefValue='' Name='Quote
Name' BusObj='Quote' Field='Name' ID='1-1Z875' BusComp='Quote'>

         </UnresolvedLink>

      </UnresolvedLinks>
```

Only links that have the execute flag set or pull system parameters such as TODAY will be resolved by the configurator when used headless. All other links must be resolved by the programmer.

NOTE: The unresolved links must be calculated and their IDs and values must be added to the Links child property set as properties with the link ID as the property and the link value as the property value.

CreateSession

Initializes a configuration session. This is necessary for customizable products that have constraint rules. This is called immediately following LoadInstance where required.

In:

- ObjId – unique identifier to the complex object header (e.g. Quote Id).
- RootId – unique identifier to the complex object root (e.g. Quote Line Item row id).
- IntObjName – name of the integration object specified in Siebel Tools.
- TriggerEvent – flag that determines if script events are triggered. Should normally be “Y”. Set to “N” for special uses of the API where script events are not desired. Should be set the same for LoadInstance and CreateSession.
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.

- NewRecord (optional) – if set to "Y", the instance will be populated with default values.
- AutoSync (optional) – if set to "Y", the instance will be synchronized to the database immediately after loading.
- ExternalScript (optional) – this parameter must be set to "Y" when running headless configurations (e.g. through Siebel COM Data Server).

Out:

If NewRecord is set to "Y" in the input property set the output will have the instance property set returned as a child of type CxObj. This is essentially the same output as what is returned from GetInstance**.**

SetInstance

Creates a configuration session with a supplied property set. This permits configuration without directly writing to the database. The structure of the input property set does not need to correspond to a Siebel object, such as a quote as indicated by the integration object specified.

In:

Same as LoadInstance but also requires the property set indicating the state to load. This property set must have the SiebelMessage object as the only first level child.

Out:

Same as LoadInstance.

SyncInstance

Saves the complex object instance from whence it came.

In:

- ObjId – unique identifier to the complex object header.
- RootId – unique identifier to the complex object root (e.g. Quote Line Item row id).
- IntObjName – name of the integration object specified in Siebel Tools.

- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.

Out:

None

UnloadInstance

Removes the existing configuration session from memory. Should be called after synchronizing the instance at the end of the configuration session.

In:

- ObjId – unique identifier to the complex object header.
- RootId – unique identifier to the complex object root.
- IntObjName – the name of the integration object that was used to load the instance.

Out:

None

GetAllPorts

Retrieves a list of all ports and possibly their contents for a product. This gets all ports for a product but does not drill down to the ports of the child products. This retrieves the basic definition of the product and does not consider any current configuration session state so every possible port will be retrieved.

In:

- Product Id – ID of the product in Internal Product.
- Version (optional) – only used in validate mode.
- GetPortDomain – flag that determines whether or not to also retrieve the domain of each port. Use Y to get the domain, N to not get the domain.

Out:

All ports are returned as children of the output property set of type Port.

```
<Output>
   <Port>
      Port Information here
   </Port>
</Output>
```

EnumObjects

Returns either all immediate objects under an object or all immediate objects under a specified port. This gets the items that are currently in the port, not the items that could be there.

In:

- ObjId – unique identifier to the complex object header (e.g. Quote ID).
- RootId – unique identifier to the complex object root (e.g. Quote Line Item row id).
- IntObjName – name of the integration object specified in Siebel Tools.
- Parent Path – path to the parent object whose child objects we want to enumerate. The path is the object's Integration ID.
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.
- Port Item Id (optional) - ID of a specific port (e.g. ORIG_ID of the port in S_PROD_ITEM table). If specified, only the items in this port are enumerated, otherwise all items in all immediate ports are returned.

Out:

Child item information is returned as child property sets as follows:

```
<Output>

   < Name=”value” Product Id=”value” Path=”value” Sequence
Number=”value” />

   …

</Output>
```

GetAttribute

Retrieves the value of an attribute.

In:

- ObjId – unique identifier to the complex object root.
- RootId – unique identifier to the complex object root.
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.
- Path – path of the item we are retrieving an attribute from.
- Name - attribute name.

Out:

The value is returned as a property of the output property set.

```
<Output Value=”value”>

</Output>
```

GetFieldValues

Retrieves field values for a product that exists in the complex product.

In:

- ObjId – unique identifier to the complex object root (e.g. Quote ID).
- RootId – unique identifier to the complex object root (e.g. Quote Line Item row id).

- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.
- Path – path to the item.

Out:

The output property set returned will have the field names as properties:

```
< Field="value" Field="value" … Field="value"/>
```

GetInstance

Gets the loaded instance as a property set. This returns the full structure of products and attributes.

In:

- ObjId – unique identifier to the complex object root.
- RootId – unique identifier to the complex object root.
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.

Out:

The entire property set output is the complex object instance.

GetParents

Retrieves all the parents of an item.

In:

- ObjId – unique identifier to the complex object root.
- RootId – unique identifier to the complex object root.
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.

- Path – item path.

Out:

The property set returned will have child property sets, each with the following properties:

```
< >

   < Product Id=”value” Name=”value” Sequence Number=”value”
Path=”value”/>

   …

</ >
```

GetPossibleDomain

Retrieves selectable items from the configuration engine for a port

In:

- ObjId – unique identifier to the complex object root (e.g. Quote ID).
- RootId – unique identifier to the complex object root (e.g. Quote Line Item row id).
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.
- Parent Path – item parent path.
- Port Item Id – item port id.

Out:

The property set returned will have the possible domain item product ids as properties, each with the value 0:

```
< ProdId1=”0” ProdId2=”0” … ProdIdn=”0” />
```

GetPossibleValues

Retrieves selectable values from the configuration engine for an attribute.

In:

- ObjId – unique identifier to the complex object root (e.g. Quote ID).
- RootId – unique identifier to the complex object root (e.g. Quote Line Item row id).
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.
- IntObjName – name of the integration object.
- Path – Integration ID of the port at which this attribute is attached.
- XA Id –ID of the attribute for which the values need to be determined.

Out:

The property set returned will have the possible values as the property names.

```
< [PossibleValue1]=”Val1” [PossibleValue2]=”Val2”/>
```

GetProductId

Gets the root Product ID of the complex object instance.

In:

- ObjId – unique identifier to the complex object header (e.g. Quote ID).
- RootId – unique identifier to the complex object root (e.g. Quote Line Item row id).
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.

Out:

The product id is returned as a property of the output property set:

```
< Product Id=”value” />
```

GetRootPath

Returns the path of the complex object instance root.

In:

- ObjId – unique identifier to the complex object header (e.g. Quote ID).
- RootId – unique identifier to the complex object root (e.g. Quote Line Item row id).
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.

Out:

The root path is returned as a property of the output property set:

```
< Path="value" />
```

HasGenerics

Returns generics and children flags for an item. A port has generics if the required cardinality is greater than the current cardinality and no default product is specified.

In:

- ObjId – unique identifier to the complex object root (e.g. Quote ID).
- RootId – unique identifier to the complex object root (e.g. Quote Line Item row id).
- IntObjName – name of the integration object.
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.
- Port Item Id – ID for the port item that will have either children or generics.
- Path – Path to the parent item of interest.

Out:

HasGenerics – Y if it does, not present if it does not.

HasChildren – Y if it does, not present if it does not.

API to Interact with Conflicts and Messages

NOTE: The API in this section only apply to customizable products with constraint rules.

GetDetailedReqExpl

Retrieves conflict messages.

In:

- ObjId – unique identifier to the complex object root (e.g. Quote ID).
- RootId – unique identifier to the complex object root (e.g. Quote Line Item row id).
- IntObjName – name of the integration object.

Out:

Expl# - The explanations for the conflicts. Substitute a number for #, such as Expl0, Expl1, etc.

GetExplanations

Retrieves configuration explanations for an item.

In:

- ObjId – unique identifier to the complex object root (e.g. Quote ID).
- RootId – unique identifier to the complex object root (e.g. Quote Line Item row id).

- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.
- Path – item path.

Out:

The property set returned will have child property sets, each with the property Value as the explanation:

```
<Output>
   <Expl Value="Explanation"/>
   <Expl Value="Explanation"/>
   …
   <Expl Value="Explanation"/>
</Output>
```

GetSignals

Retrieves configuration engine signals.

In:

- ObjId – unique identifier to the complex object root (e.g. Quote ID).
- RootId – unique identifier to the complex object root (e.g. Quote Line Item row id).
- Path – integration ID for the item to get signals at (optional).
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.

Out:

The property set returned will have child property sets as follows:

```
< >
```

```
   <Signal Expl="signal"/>
   <Signal Expl="signal"/>
   …
   <Signal Expl="signal"/>
</ >
```

RemoveFailedRequests

Removes all failed requests sent to the configuration engine.

In:

- ObjId – unique identifier to the complex object root (e.g. Quote ID).
- RootId – unique identifier to the complex object root (e.g. Quote Line Item row id).
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.

Out:

None

UndoLastRequest

Removes the last request sent to the configuration engine.

In:

- ObjId – unique identifier to the complex object root (e.g. Quote ID).
- RootId – unique identifier to the complex object root (e.g. Quote Line Item row id).
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.

Out:

None

API to Set Product and Attribute Values

This section contains API used for setting product and attribute values.

AddItem

Adds an item to a specified port. This will create a new instance of an item. If you want to change the quantity of an existing instance of an item, use SetItemQuantity.

In:

- ObjId – unique identifier to the complex object header.
- RootId – unique identifier to the complex object root.
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.
- AutoResolve (optional) – automatically resolves port cardinality violations.
- Prod Item Id– ID of the item (e.g. ORIG_ID in S_PROD_ITEM table)
- Name - item name.
- Product Id– product id in S_PROD_INT table.
- Port Item Id – ID of the item's port (e.g. ORIG_ID of the port in S_PROD_ITEM table).
- Quantity – item quantity.
- List Price – item list price from Pricing Manager, can be empty.
- Current Price – current price from Pricing Manager, can be empty.
- Parent Path – path of the parent item the port belongs to.

Out:

None.

CopyInstance

Copies an instance.

In:

- ObjId – unique identifier to the complex object header of the source instance.
- RootId – unique identifier to the complex object root of the source instance.
- DestObjId – unique identifier to the complex object header of the destination instance.
- IntObjName – name of the integration object specified in Siebel Tools.

Out:

None.

RemoveItem

Removes an item from the instance.

In:

- ObjId – unique identifier to the complex object header.
- RootId – unique identifier to the complex object root.
- Version (optional) – Version arguments are used only when testing a customizable product version that is different from the currently released version.
- Path – path of the item.

Out:

None.

RepriceInstance

Updates the instance with values from the Pricing Manager service. A call to the Pricing Manager service's CalculatePriceCX method returns a property set that is the input to this method.

In:

- ObjId – unique identifier to the complex object header.

- RootId – unique identifier to the complex object root.
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.

The input property set has child property sets containing the reprice information. The format of the input property set is as follows:

```
< ObjId="value" RootId="value" Version="value" >

   < IntId="integration id" FieldName="value"… FieldName="value" >

   < IntId="integration id" FieldName="value"… FieldName="value" >

    …

< />
```

In this context, the Integration ID is used to retrieve the instance item.

Out:

None.

SetAttribute

Sets the value of an item's attribute.

In:

- ObjId – unique identifier to the complex object root.
- RootId – unique identifier to the complex object root.
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.
- Path – path of the item with attribute to set.
- Value – attribute value.
- Name – attribute name.

- XA Id – extended attribute ID. This is the row ID of the attribute in the XA Attribute business component.
- Property Type Code – attribute type.

Out:

None.

SetItemQuantity

Sets the quantity of an item.

In:

- ObjId – unique identifier to the complex object root.
- RootId – unique identifier to the complex object root.
- Version (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version.
- Path – path of the item.
- Quantity – quantity to set.

Out:

None.

Object Broker Methods

The methods in this section call the Cfg Object Broker business service, which functions as a wrapper for the Object Broker.

GetProdStruct

This method returns the full structure of the customizable product.

In:

- RootId – Unique identifier to the complex object root. If provided, RootName, Vendor, and Org are ignored. If not provided, RootName, Vendor, and Org are used to uniquely identify the product.
- RootName – The root product name. Name can be used together with Vendor and Org as optional to uniquely identify a product.
- Version – (optional) – Version arguments are used ONLY when testing a customizable product version that is different from the currently released version. Specify 0 to return the work space.
- Vendor (optional) – Use with RootName to uniquely identify product.
- Org (optional) – Use with RootName to uniquely identify product.
- Full – Yes returns full product structure. Blank or No returns the first level of the product.

Out:

```
 <ProdStruct> RootId

    <ProdId> Name ClassId

       <Port> Name ClassName ClassId OrigId Type MinCard MaxCard
DefltCard LocalType InternalType

          <Subobject Id/>

          ...

       </port>

       ...

       <Attribute> Name

          <Domain Value />

          ...

       </Attribute>

       ...

    </ProdId>
```

```
    ...
</ProdStruct>
```

DeltaQuote

Performs a recursive tree comparison of two property sets to determine the difference between them based on a supplied criteria. This returns a copy of the destination product instance marked up to indicate changes. Out of the box, this API is called from an external service only in SIS Order Management so using that feature is the only way to see this API in action for the API Discovery.

Example: you start with a computer that has one hard drive and a 900 MHz processor. You upgrade it to add a second hard drive (quantity now = 2) and replace the processor with a 1000 MHz model. The result would be one hard drive existing, one hard drive new (the instance is split), one 900 MHz CPU removed, and one 1000 MHz CPU new.

If the output property set is empty and no error code is thrown, the cause is most likely that the instances were not recognized. Check the RootId parameters and the indenting of your source and destination SiebelMessages.

In:

- SrcRootId – Root ID for the product in the source of the comparison. This is the "before" property set.
- DestRootId – Root ID for the product in the destination of the comparison. This is the "after" property set.
- DeltaSrcField – the Path field in the source instance.
- DeltaDestField – the Path field in the destination instance.
- SrcItemIntComp – name of the integration component for the items in the source instance. For example, "Quote Item".
- DestItemIntComp – name of the integration component for the items in the destination instance. For example, "Quote Item".
- SrcXAIntComp – name of the integration component for the XA in the source instance. For example, "Quote Item XA".

- DestXAIntComp – name of the integration component for the XA in the destination instance. For example, “Quote Item XA”.
- ITEM_MAPPING – This is a property set with type = “ITEM_MAPPING”. It contains a list of those fields that should be copied when creating a new instance of an item in the destination property set. The property names are the fields in the source instance, the property values are the names in the destination instance. Here is an example output from the API sniffers:

```
CHILD PROPERTY SET 3

      Type: ITEM_MAPPING Value:

      Unit Price = Unit Price

      Action Code = Action Code

      Root Id = Root Id

      Port Item Id = Port Item Id

      Integration Id = Integration Id

      Discount Amount = Discount Amount

      Parent Id = Parent Id

      Product Id = Product Id

      Prod Item Id = Prod Item Id

      Quantity = Quantity
```

- XA_MAPPING - This is a property set with type = “XA_MAPPING”. It contains a list of those fields that should be copied for each of the attributes when creating a new instance of an item in the destination property set. The property names are the fields in the source instance; the property values are the names in the destination instance. Here is an example output from the API sniffers:

```
CHILD PROPERTY SET 2

      Type: XA_MAPPING Value:

      Action Code = Action Code

      Value = Value
```

```
Read Only = Read Only

Name = Name

Property Type Code = Property Type Code

XA Id = XA Id
```

- ITEM_COMPARE - This is a property set with type = "ITEM_COMPARE". This defines what constitutes a unique instance of an item in the source and destination instances. It answers for the service the question "How do I know if these two things are the same?". Here is an example output from the API sniffers:

```
CHILD PROPERTY SET 1

Type: ITEM_COMPARE Value:

Port Item Id = Port Item Id

Product Id = Product Id
```

- XA_COMPARE - This is a property set with type = "XA_COMPARE". This defines what constitutes a unique instance of an attribute in the source and destination instances. It answers for the service the question "How do I know if these two things are the same?". Here is an example output from the API sniffers:

```
CHILD PROPERTY SET 0

Type: XA_COMPARE Value:

Value = Value

Name = Name

Property Type Code = Property Type Code

XA Id = XA Id
```

- SrcInst - This is the Source Instance ("before") that will be used in the delta comparison. This is a double-indented SiebelMessage with a type of "SrcInst". The top section of the output from the API sniffers is pasted below for reference. Note the indenting of the SiebelMessage.

```
CHILD PROPERTY SET 4

Type: SrcInst Value:
```

```
CHILD PROPERTY SET 0
   Type:  Value:
   CHILD PROPERTY SET 0
      Type: SiebelMessage Value:
      MessageId = 1-12949
      IntObjectFormat = Siebel Hierarchical
      MessageType = Integration Object
      IntObjectName = CX Product Validation
      CHILD PROPERTY SET 0
         Type: ListOfCX Product Validation Value:
         CHILD PROPERTY SET 0
            Type: Product Header Value:
            Name = 1-12950
            Price List Id = 1-ZEC
            Id = 1-12951
            CHILD PROPERTY SET 0
               Type: ListOfProduct Item Value:
               CHILD PROPERTY SET 0
                  Type: Product Item Value:
                  Action Code = Existing
                  Port Item Id =
                  Integration Id = 1-12744
                  Cfg Type = eConfigurator
                  Name = System Chassis
                  Product Id = 1-1M9Y
```

```
Prod Item Id = null
Quantity = 1.0
Id = 1-12744
```

- DestInst - This is the Destination Instance (“after”) that will be used in the delta comparison. This is a double-indented SiebelMessage with a type of “DestInst”. The top section of the output from the API sniffers is pasted below for reference. Note the indenting of the SiebelMessage.

```
CHILD PROPERTY SET 5
      Type: DestInst Value:
      CHILD PROPERTY SET 0
         Type:  Value:
         CHILD PROPERTY SET 0
            Type: SiebelMessage Value:
            MessageId = 123
            IntObjectFormat = Siebel Hierarchical
            MessageType = Integration Object
            IntObjectName = CX Product Validation
            CHILD PROPERTY SET 0
               Type: ListOfCX Product Validation Value:
               CHILD PROPERTY SET 0
                  Type: Product Header Value:
                  Name = 1-12950
                  Price List Id = 1-ZEC
                  Id = 1-12951
                  CHILD PROPERTY SET 0
                     Type: ListOfProduct Item Value:
```

```
CHILD PROPERTY SET 0
   Type: Product Item Value:
   Has Generics Flag = Y
   Action Code = Existing
   Integration Id = 1-12744
   Port Item Id =
   Sequence Number =
   Name = System Chassis
   Cfg Type = eConfigurator
   Product Id = 1-1M9Y
   Quantity = 1.0
   Prod Item Id = null
   Id = 1-12744
```

Out:

The destination instance is returned as a SiebelMessage, modified and marked up with status information. The status (new, modified, existing, removed) is indicated in the Action Code field of each item and attribute.

Application Integration Network

Siebel provides a library of business services that allow Siebel applications to share information with other applications through integration servers. This library is called the Universal Application Network, and its business services are called Application Service Interfaces (ASI's).

There are two ASI's for managing products: External Simple Product and Siebel Simple Product. These are briefly described below. For a full technical description of these ASI's, see *Application Services Interface Reference: Siebel eBusiness Application Integration Volume VI.*

External Simple Product

This ASI sends information about simple products created in the Siebel system to an external, third-party system. This ASI allows you to create, update, query, and delete a product in the third-party system. This ASI is intended for inclusion in Siebel work flows that automate exporting products. This ASI does not support sending information about customizable products or bundles.

The export feature included in the Siebel Product Administration interface exports basic product information to an XML file and is intended for Siebel-to-Siebel transfers of product records. The External Simple Product ASI exports a much larger set of information about the product.

This information sent includes almost all of the fields in the product record, except those relating to customizable products.

This ASI receives the following information:

- Confirmation
- Error messages
- Status
- Product ID

Siebel Simple Product

This ASI receives information about simple products created in third-party systems This ASI allows users to create, update, query, and delete a products in the Siebel system. This ASI is intended for use in automated business processes in third-party systems that need to synchronize the Siebel system to external product masters. This ASI does not support receiving information about customizable products or bundles. The information accepted by this ASI includes the following:

- Product name
- Product description
- Part number
- Product attributes and attribute values
- Product unit of measure
- Product classification
- Product type
- Substitute product name
- Literature
- Product catalog
- Product category
- Price list

This ASI sends the following information:

- Confirmation
- Error messages
- Status
- Product ID

Index

C

G

H

I

P

Q

R

S

T

U

V

W

X